Windows Server RDMA Deployment Guide

Including WS 2016 – 2022, Native and Converged NIC scenarios, Data Center Bridging, & Guest RDMA

Last Update 11/10/2021

Important: Azure Stack HCI Deployments

Azure Stack HCI is a new operating system that contains the same HCI technologies with Cloud enhancements. Microsoft highly recommends using the Azure Stack HCI operating system which includes a new technology to deploy the RDMA capabilities in this guide call Network ATC.

Network ATC is the recommended deployment mechanism for Azure Stack HCI host networking and eliminates the need for this guide and several others.

For more information on Network ATC, please see: Azure Stack HCI: Simplify host networking with Network ATC

If you are using Windows Server, you will not be able to use Network ATC. Instead, you can use this guide and configure the machine manually.

Introduction to the Deployment Guide

The Instructions below provide the detailed steps to deploy Windows Server 2016 - Windows Server 2022 with RDMA over Converged Ethernet (RoCE). This guide also provides instructions for implementing Guest RDMA deployments, first introduced in Windows Server 2016 v1709. To the degree that we are aware of differences between RDMA technologies and vendor-specific requirements, we have included those differences in this paper.

Before you begin

Please read the RDMA specific documentation found here which outline and compares the various RDMA technologies.

While *Microsoft recommends using an iWARP RDMA* solution as it has proven to be significantly less complex than most RoCE deployments, we are aware that vendors have been actively working to reduce the complexity associated with their RoCE-based solutions. Microsoft cannot directly endorse or validate the efficacy of these efforts.

Instead, we recommend that you work with your vendor or solution integrator to fully test any RDMA workload under full fabric (physical network) stress conditions to ensure that your solution is properly configured and performing as expected. This would include any test that fully saturates the fabric bandwidth while running Storage Spaces Direct with intensive VM workloads and various other scenarios including, but not limited to:

- Patching and rebooting host cluster nodes
- Running backups

How to read this guide

The instructions in this guide apply to two configurations (see **Error! Reference source not found.** and **Error! Reference source not found.**).

- The Instructions in this document marked in GREEN are for BASIC single adapter scenarios only. This is the case where only the minimal set of operations and resources are desired.
- The instructions in this document marked in BLUE are for the recommended Dual-port configuration Datacenter deployment with multiple RDMA Host vNICs for maximum performance and availability.
- The instructions in BLACK apply to both configurations.
- When the output from PowerShell cmdlets is shown, the important parts of the output are highlighted in yellow.

Contents

Figures

RDMA modes of operation

The Network Direct Kernel-mode Provider Interface (NDKPI), which specifies how an RDMA-capable NIC should interface with the Windows Operating System, defines three modes of operation:

- NDKPI Mode 1: Native host to Native host RDMA communication
- NDKPI Mode 2: Virtual Switch RDMA – RDMA exposed on a Host vNIC of a Hyper-V Switch
- NDKPI Mode 3: Guest RDMA – RDMA exposed on a Guest vmNIC through an SR-IOV virtual function

This document covers all three scenarios. Any RDMA interface in a Windows host, no matter which mode of operation it is in, can communicate to any other RDMA interface in any other Windows host as long as both systems support the same RDMA protocol (e.g., iWARP or RoCEv2). The mode applies only to the local RDMA interface. Specifically,

- A Native host RDMA interface (e.g., a File Server) can communicate with another Native host, a Converged NIC instance, or a Guest RDMA instance.
- A Converged NIC instance can communicate with a Native host instance, another Converged NIC instance, or a Guest RDMA instance.
- A Guest RDMA instance can communicate with a Native host instance, a Converged NIC instance, or another Guest RDMA instance.

Terminology

iWARP	RDMA over TCP
pNIC	Physical NIC, the physical hardware that exchanges packets with the TOR
RoCE	RDMA over Converged Ethernet
RoCEv2	2nd generation RoCE using UDP/IP for routability (a.k.a. Routable RoCE)
TOR	Top of Rack switch
vmNIC	Virtual Machine NIC – Virtual NIC from vSwitch exposed in a guest partition
vNIC	Host vNIC – Virtual NIC from vSwitch exposed in the host partition
vSwitch	Hyper-V virtual switch

Intended configurations

For this paper we assume one of the two following configurations are the goal. We further assume the administrator has two hosts with the same physical configuration available, i.e., either two of the single NIC hosts or two of the two-NIC hosts. The paper starts with only the operating system installed and only fully configures one host.

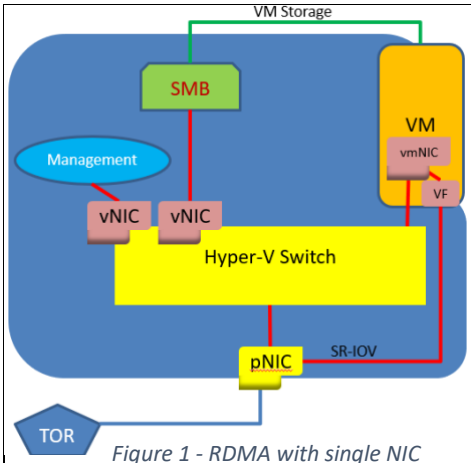

Figure 1 - RDMA with single NIC

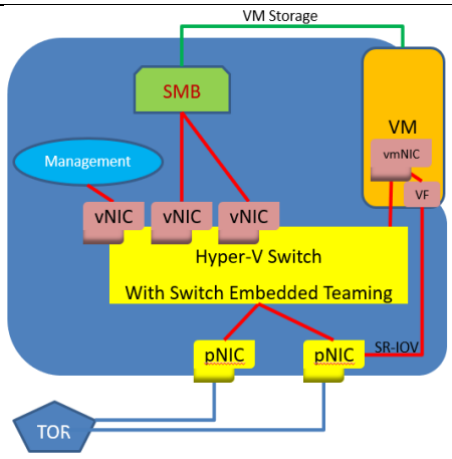

Figure 2 - RDMA with SET Teamed NICs

Step 1: Test Basic Connectivity

Let's start with just the Operating System (Windows Server 1709) installed. We'll add Hyper-V and the other needed components later in this process.

Step 1a – Single pNIC configuration

Single NIC configuration

Rename the pNIC to "NIC1" in each host. This optional step makes it possible to reuse the PowerShell cmdlets below as shown.

```
PS> Rename-NetAdapter Ethernet NIC1
```

Check that the name change took effect.

```
PS> Get-NetAdapter

Name InterfaceDescription    ifIndex Status MacAddress         LinkSpeed
---- --------------------    ------- ------ ----------         ---------
NIC1 Chelsio Network Adapter       3 Up     00-07-43-2D-D6-D8   40 Gbps
```

Host A: Assign IP address 192.168.1.3 to the pNIC

```
PS> New-NetIPAddress -InterfaceAlias NIC1 -IPAddress 192.168.1.3 -PrefixLength 24
```

Confirm the addresses have been assigned:

```
PS> Get-NetIPAddress -InterfaceAlias NIC1 | ft IPAddress
```

The return should show both the default IPv6 address and the assigned IPv4 address.

```
IPAddress
---------
fe80::dcaa:bda9:a33a:c570%9
192.168.1.3
```

Host B: Assign IP address 192.168.1.5 to the pNIC

```
PS> New-NetIPAddress -InterfaceAlias NIC1 -IPAddress 192.168.1.5 -PrefixLength 24
```

Confirm the addresses have been assigned:

```
PS> Get-NetIPAddress -InterfaceAlias NIC1 | ft ipaddress
```

The return should show both the default IPv6 address and the assigned IPv4 address.

```
IPAddress
---------
fe80::d0f1:81d2:22fd:68a8%11
192.168.1.5
```

At the end of this step the configuration of your hosts should resemble:

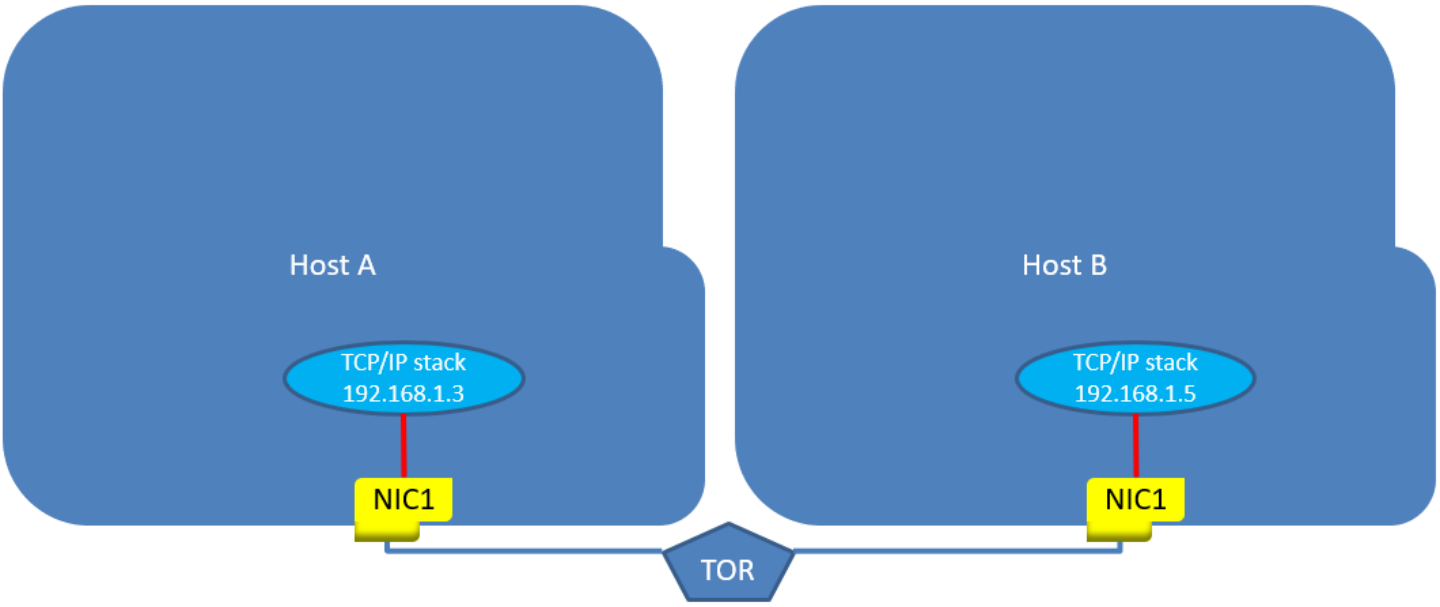

Figure 3 - Single port configuration

Step 1a – Dual pNIC configuration

Two-NIC configuration

Rename the two pNICs in each host to "NIC1" and "NIC2" (this optional step makes it possible to reuse the PowerShell cmdlets below as is).

```
PS> Rename-NetAdapter Ethernet NIC1
PS> Rename-NetAdapter "Ethernet 2" NIC2
```

Check that the name changes happened:

```
Name InterfaceDescription       ifIndex Status MacAddress         LinkSpeed
---- --------------------       ------- ------ ----------         ---------
NIC2 Chelsio Network Adapter #2       3 Up     00-07-43-2D-D6-D8   40 Gbps
NIC1 Chelsio Network Adapter          4 Up     00-07-43-2D-D6-D0   40 Gbps
```

Host A: Assign IP address 192.168.1.3 to NIC1 and assign IP address 192.168.2.3 to NIC2 on Host A

```
PS> New-NetIPAddress -InterfaceAlias NIC1 -IPAddress 192.168.1.3 -PrefixLength 24
PS> New-NetIPAddress -InterfaceAlias NIC2 -IPAddress 192.168.2.3 -PrefixLength 24
```

Confirm the addresses have been assigned:

```
PS> Get-NetIPAddress -InterfaceAlias NIC1,NIC2 | ft ipaddress
```

The output should show both the default IPv6 addresses and the assigned IPv4 addresses.

```
IPAddress
---------
fe80::25d3:1e0d:e9de:20d%12
fe80::b55e:e8dc:dea0:c7dd%7
192.168.1.3
192.168.2.3
```

Host B: Assign IP address 192.168.1.5 to NIC1 and assign IP address 192.168.2.5 to NIC2

```
PS> New-NetIPAddress -InterfaceAlias NIC1 -IPAddress 192.168.1.5 -PrefixLength 24
PS> New-NetIPAddress -InterfaceAlias NIC2 -IPAddress 192.168.2.5 -PrefixLength 24
```

Confirm the addresses have been assigned:

```
PS> Get-NetIPAddress -InterfaceAlias NIC1,NIC2 | ft ipaddress
```

The output should again show both the default IPv6 address and the assigned IPv4 address.

At the end of this step the configuration of your hosts should look like:

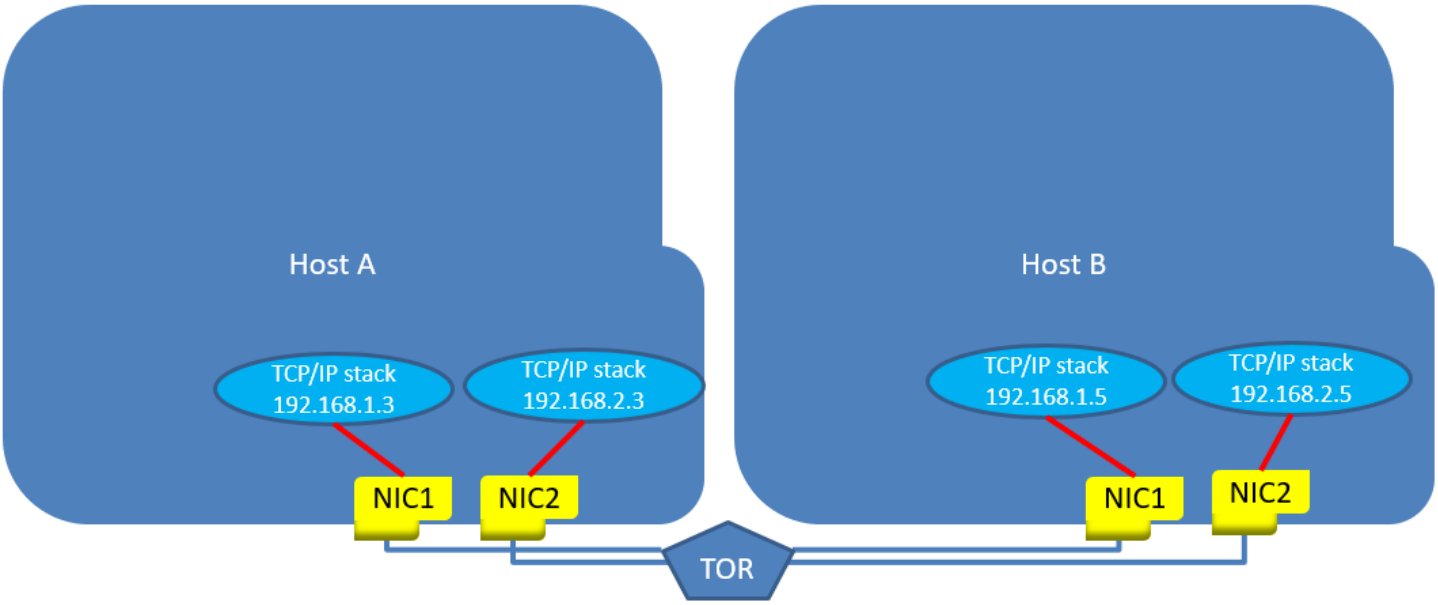

Figure 4 - Dual-port configuration

Step 1b – Check to see that the pNIC(s) have connectivity to the TOR

On each host execute

```
PS> Get-NetAdapter | ft -Autosize
```

The response will have the following columns:

```
Name InterfaceDescription    ifIndex Status MacAddress         LinkSpeed
---- --------------------    ------- ------ ----------         ---------
NIC1 Chelsio Network Adapter       3 Up     00-07-43-2D-D6-D8    40 Gbps
```

If the Status is not "Up" or the LinkSpeed is "0 bps" you don't have connectivity to the TOR from that pNIC. Check the cabling and try again until all interfaces show "Up" with a LinkSpeed that you expect (e.g., "10 Gpbs").

Step 1c – Single pNIC: Check host-to-host connectivity

Host A: Execute

```
PS> Test-NetConnection 192.168.1.5
```

The response should look like

```
ComputerName            : 192.168.1.5
RemoteAddress           : 192.168.1.5
InterfaceAlias          : NIC1
SourceAddress           : 192.168.1.3
PingSucceeded           : True
PingReplyDetails (RTT) : 0 ms
```

If the PingSucceeded line is not "True" check your firewall settings to ensure the interface is allowed to communicate to the outside world. Once you have a successful test you are ready to move to the next step.

Step 1c – Dual pNIC: Check host-to-host connectivity

Host A: Execute

```
PS> Test-NetConnection 192.168.1.5
```

The response should look like

```
ComputerName            : 192.168.1.5
RemoteAddress           : 192.168.1.5
InterfaceAlias          : NIC1
SourceAddress           : 192.168.1.3
PingSucceeded           : True
PingReplyDetails (RTT) : 0 ms
```

If the PingSucceeded line is not "True" check your firewall settings on both Host A and Host B to ensure both the source and destination interfaces are allowed to communicate to the outside world.

Repeat the test with the other address of Host B, i.e.,

```
PS> Test-NetConnection 192.168.2.5
```

Host B: Repeat the tests to ensure both Host A interfaces are available, i.e.,

```
PS> Test-NetConnection 192.168.1.3
PS> Test-NetConnection 192.168.2.3
```

Once you have all four successful tests (two from Host A, two from Host B) you are ready to move to the next step.

Step 2: Configure VLANs

Note 1: This step is optional for iWARP. This step should be optional for RoCEv1 and RoCEv2, too, but most network switches don't handle traffic with marked priorities (DCB Traffic Classes) unless they are also VLAN tagged, so we strongly recommend configuration of VLANs for any RoCE traffic. (More information below in Step 3.)

Note 2: At this point in the process the NICs are in ACCESS mode. However, when a switch is created later, the VLAN properties are applied at the vSwitch port level. Given a vSwitch will host multiple VLANs it is necessary for the Physical Switch (ToR) to have its port configured in Trunk mode. Consult the switch vendor documentation for instructions.

Step 2a – Single pNIC: Apply VLAN 101 to both Hosts pNICs

In both hosts execute:

```
PS> Set-NetAdapterAdvancedProperty NIC1 -RegistryKeyword VlanID -RegistryValue "101"
```

Because some adapters only pick up new registry keywords after being restarted, restart the pNIC on each host:

```
PS> Restart-NetAdapter NIC1
```

To see that the value has been set, execute:

```
PS> Get-NetAdapterAdvancedProperty -Name NIC1 | Where-Object {$_.RegistryKeyword -eq
    "VlanID"} | ft -AutoSize
```

The result should look like one of the lines below:

```
Name DisplayName     DisplayValue RegistryKeyword RegistryValue
```

```
---- -----------     ------------ --------------- -------------
NIC1 VLAN ID         101          VlanID          {101}
```

Note: Different hardware vendors may show different strings in the "Display Name" column.

Step 2a – Dual pNIC: Apply VLAN 101 and VLAN 102 to the Host pNICs

In both hosts execute:

```
PS> Set-NetAdapterAdvancedProperty NIC1 -RegistryKeyword VlanID -RegistryValue "101"
PS> Set-NetAdapterAdvancedProperty NIC2 -RegistryKeyword VlanID -RegistryValue "102"
```

Because some adapters only pick up new registry keywords after being restarted, restart the pNICs on each host:

```
PS> Restart-NetAdapter NIC1,NIC2
```

Check to see that the VLAN values have been set

```
PS> Get-NetAdapterAdvancedProperty -Name NIC1,NIC2 | Where-Object
    {$_.RegistryKeyword -eq "VlanID"} | ft -AutoSize
```

The result should look like one of the lines below:

```
Name DisplayName  DisplayValue  RegistryKeyword  RegistryValue
---- -----------  ------------  ---------------  -------------
NIC1 VLAN ID      101           VlanID           {101}
NIC2 VLAN ID      102           VlanID           {102}
```

Note: Different hardware vendors may show different strings in the "Display Name" column.

Step 2b – Check that connectivity to the switch and the other host is still present

Repeat Step 1c (above). If the interfaces do not show "Up" or the LinkSpeed shows "0 Gbps" the interfaces are not ready for use. Wait a short time and check again. It may take several seconds after a restart before the pNIC is visible on the network.

Single pNIC:

Host A: Execute

```
PS> Test-NetConnection 192.168.1.5
```

The response should look like

```
ComputerName            : 192.168.42.100
RemoteAddress           : 192.168.42.100
InterfaceAlias          : vEthernet (GuestRdma)
SourceAddress           : 192.168.42.101
PingSucceeded           : True
PingReplyDetails (RTT) : 0 ms
```

If the PingSucceeded line is not "True", either

a) the TOR is not set correctly to pass VLAN-tagged traffic. Consult your TOR documentation to ensure the ports on the TOR are set to trunk mode or at least are explicitly set to pass VLAN 102 traffic; or
b) the firewall on one or both hosts have not been set to pass the ping traffic. Check your firewall rules to make sure the firewall is set to pass pings through both directions. (To disable all firewall policies in Windows – not recommended for production environments – use Set-NetFirewallProfiles -All -Enabled FALSE.)

Dual pNIC:

Host A: Execute

```
PS> Test-NetConnection 192.168.1.5
```

The response should look like

```
ComputerName            : 192.168.1.5
RemoteAddress           : 192.168.1.5
InterfaceAlias          : vEthernet (GuestRdma)
SourceAddress           : 192.168.1.3
PingSucceeded           : True
PingReplyDetails (RTT) : 0 ms
```

Repeat the test with the other VLAN, i.e.,

```
PS> Test-NetConnection 192.168.2.5
```

If the PingSucceeded line is not "True" in either of the above tests, most likely either

a) the TOR is not set correctly to pass VLAN-tagged traffic. Consult your TOR documentation to ensure the ports on the TOR are set to trunk mode or at least are explicitly set to pass VLAN 102 traffic; or
b) the firewall on one or both hosts have not been set to pass the ping traffic. Check your firewall rules to make sure the firewall is set to pass pings through both directions. (To disable all firewall policies in Windows – not recommended for production environments – use `Set-NetFirewallProfiles -All -Enabled FALSE`.)

Step 3: Configure DCB

Note 1: This step is optional for iWARP. However, iWARP may benefit from DCB at large fabric scale, so configure DCB at your discretion.

Note 2: Some vendors claim that RoCEv2 with ECN has no requirement for DCB. While RoCEv2 with ECN may work very well in a single-rack environment without DCB, it is Microsoft's belief that RoCEv2 with ECN will still require DCB at any scale larger than a single rack due to the longer round trip delays and the impact on the required size of buffers throughout the network. We strongly encourage you to configure DCB for any RoCEv2-based RDMA deployment. (DCB is always required for any RoCEv1 deployment. RoCEv1 can't extend beyond a single Layer 2 broadcast domain, typically a single rack.)

These steps MUST be done identically on each of the hosts in your set-up.

Step 3a: Install DCB

Install the Data Center Bridging feature in Windows Server.

```
PS> Install-WindowsFeature Data-Center-Bridging
```

The response should be:

```
Success Restart Needed Exit Code Feature Result
------- -------------- --------- --------------
True    No             Success   {Data Center Bridging}
```

If the Success value is not "True" something has gone remarkably wrong. Try again; contact your support organization if it continues to fail.

Step 3B: Set policy for Cluster Heartbeats

While long a beneficial practice, we are now officially recommending creating a cluster heartbeat bandwidth reservation. This is particularly helpful in 10 Gbps Converged scenarios or where the bandwidth available can be easily oversubscribed. This section will setup a policy to tag Cluster Heartbeats (port 3343) with a priority tag. In this example, we use priority tag 7.

Note: Cluster heartbeat traffic should be considered the highest priority traffic in your network (priority 7). On Windows you must configure both the PFC and ETS settings to associate the bandwidth reservation with the cluster heartbeats. On the fabric make sure to reserve the appropriate bandwidth for the cluster heartbeats. Configuration of PFC on the network infrastructure is not required so long as the switch respects the ETS bandwidth reservation for this traffic. Please consult with your switch vendor.

```
PS> New-NetQosPolicy "Cluster" -Cluster -PriorityValue8021Action 7
```

Next, you need to reserve some bandwidth for the Cluster Heartbeat traffic. This example uses 1%, as this is the smallest amount that can be reserved. Cluster heartbeats use only a very small portion of traffic (43 Bytes) and so for a adapters that are > 10 Gbps, this is more than enough. In practice, we have found that 2% works best for 10 Gbps adapters.

```
PS> New-NetQosTrafficClass "Cluster" -Priority 7 -BandwidthPercentage 1 -Algorithm ETS
```

The response to the traffic class creation should look like:

```
Name      Algorithm Bandwidth(%) Priority  PolicySet        IfIndex IfAlias
----      --------- ------------ --------  ---------        ------- -------
Cluster   ETS         1            7       Global
```

Step 3C: Set policy for SMB-Direct

Set up a policy to tag SMB-Direct packets with a priority tag. In this example we use priority tag "3". Keep in mind that DCB's QoS policies apply globally, so SMB packets sent using RDMA ("NetDirect" is synonymous with RDMA) will always get tagged with the value "3" no matter what interface they are sent on. *Note: while this guide uses the tag value 3, any tag value between 1 and 7 inclusive can be used as long as it is used everywhere through the network in both the hosts and the switches/routers.*

```
PS> New-NetQosPolicy "SMB" -NetDirectPortMatchCondition 445
    -PriorityValue8021Action 3
```

Note: While an `SMB` template exists, it cannot be used to specify *NetDirect* (RDMA) traffic. For RDMA traffic, please use the `NetDirectPortMatchCondition parameter`

The response should look like:

```
Name           : SMB
Precedence     : 127
NetDirectPort  : 445
PriorityValue  : 3
```

Next you need to enable PFC for the SMB-Direct traffic:

```
PS> Enable-NetQosFlowControl -priority 3
```

Now reserve bandwidth for the SMB-Direct traffic. This example uses 50%, but you may want to reserve more or less depending on what you expect the ratio of non-Storage traffic to Storage traffic will be in your facility.

```
PS> New-NetQosTrafficClass "SMB" -priority 3 -bandwidthpercentage 50 -algorithm ETS
```

The response to the traffic class creation should look like:

```
Name Algorithm Bandwidth(%) Priority  PolicySet        IfIndex IfAlias
---- --------- ------------ --------  ---------        ------- -------
SMB  ETS       50           3         Global
```

Finally, set these policies on the interface you want to use

If you are using a Single Port configuration:

```
PS> Enable-NetAdapterQos -InterfaceAlias NIC1
```

If you are using a Dual Port configuration:

```
PS> Enable-NetAdapterQos -InterfaceAlias NIC1,NIC2
```

Step 3D: Block DCBX settings from the switch

By default, Windows Network Adapters (NICs) can accept DCB settings from the adjacent network switch through the use of the DCBX protocol. However, since the Windows operating system never looks at what settings the switch sent to the NIC, and in the steps in this section Windows will explicitly tell the NIC what DCB settings to use, it is safest to ensure that the NIC is told not to accept such settings from the network switch.

To disable DCBX in the NIC, If you are using a Single Port configuration:

```
PS> Set-NetQosDcbxSetting -InterfaceAlias NIC1 -Willing $False
```

If you are using a Dual Port configuration:

```
PS> Set-NetQosDcbxSetting -InterfaceAlias NIC1 -Willing $False
PS> Set-NetQosDcbxSetting -InterfaceAlias NIC2 -Willing $False
```

Step 3E: Set policy for the rest of the traffic (optional)

Make sure that all the non-SMB/RDMA traffic goes without a priority tag. While this shouldn't be necessary because the default priority tag is 0 (untagged), there is no harm in making sure

```
PS> New-NetQosPolicy "DEFAULT" -Default -PriorityValue8021Action 0
```

If you want to make sure PFC isn't on the non-SMB traffic you can actively disable it.

```
PS> Disable-NetQosFlowControl -priority 0,1,2,4,5,6,7
```

Before you proceed any further, verify with your network administrator that all ports on the physical switches that will have RoCE RDMA traffic have been configured with DCB enabled and with PFC on the identified traffic (traffic tagged with 3 and 7 in this example). *Appendix 1 has some examples of possible switch/router configurations. Consult your switch/router vendor for details.*

Step 3F: Validate your settings (optional)

While the above steps, if all entered correctly, are all you need, it can be a good idea to validate that you got what you asked for. The commands to check on NetQosFlowControl and NetAdapterQos are:

```
PS> Get-NetQosFlowControl
```

Which should return "True" for the priority tags for which you have turned on PFC and "False" for the rest, e.g.,

```
Priority   Enabled    PolicySet        IfIndex IfAlias
--------   -------    ---------        ------- -------
0          False      Global
1          False      Global
2          False      Global
3          True       Global
4          False      Global
5          False       Global
6          False      Global
7          False      Global
```

In the Single-port Configuration

```
PS> Get-NetAdapterQos -Name NIC1
```

which returns:

```
Name                        : NIC1
Enabled                     : True
Capabilities                :                           Hardware     Current
                                                        --------     -------
                              MacSecBypass          : NotSupported NotSupported
                              DcbxSupport           : None         None
                              NumTCs(Max/ETS/PFC)   : 8/8/8        8/8/8

OperationalTrafficClasses   : TC TSA    Bandwidth Priorities
                              -- ---    --------- ----------
                               0 ETS    49%       0-2,4-6
                               1 ETS    1%        7
                               2 ETS    50%       3

OperationalFlowControl      : Priority 3 Enabled
OperationalClassifications  : Protocol  Port/Type Priority
                              --------  --------- --------
                              Default               0
                              NetDirect 445         3
```

In the Dual-port Configuration

```
PS> Get-NetAdapterQos -Name NIC1,NIC2
```

which returns:

```
Name                          : NIC1
Enabled                       : True
Capabilities                  :                           Hardware     Current
                                                          --------     -------
                                MacSecBypass            : NotSupported NotSupported
                                DcbxSupport             : None         None
                                NumTCs(Max/ETS/PFC)     : 8/8/8        8/8/8

OperationalTrafficClasses     : TC TSA    Bandwidth Priorities
                                -- ---    --------- ----------
                                 0 ETS    49%       0-2,4-6
                                 1 ETS    1%        7
                                 2 ETS    50%       3

OperationalFlowControl        : Priority 3 Enabled
OperationalClassifications    : Protocol  Port/Type Priority
                                --------  --------- --------
                                Default             0
                                NetDirect 445       3

Name                          : NIC2
Enabled                       : True
Capabilities                  :                           Hardware     Current
                                                          --------     -------
                                MacSecBypass            : NotSupported NotSupported
                                DcbxSupport             : None         None
                                NumTCs(Max/ETS/PFC)     : 8/8/8        8/8/8

OperationalTrafficClasses     : TC TSA    Bandwidth Priorities
                                -- ---    --------- ----------
                                 0 ETS    49%       0-2,4-6
                                 1 ETS    1%        7
                                 2 ETS    50%       3

OperationalFlowControl        : Priority 3 Enabled
OperationalClassifications    : Protocol  Port/Type Priority
                                --------  --------- --------
                                Default             0
                                NetDirect 445       3
```

[Optional] Step 3G: Configure Co-existence with a Debugger

In Windows Server when a debugger gets attached it interferes with NetQos (DCB). To make this configuration possible the following command must be run:

```
PS> Set-ItemProperty HKLM:"\SYSTEM\CurrentControlSet\Services\NDIS\Parameters"
    AllowFlowControlUnderDebugger -type DWORD -Value 1 –Force
```

To validate that the Registry Keyword has been created, run:

```
PS> Get-ItemProperty HKLM:"\SYSTEM\CurrentControlSet\Services\NDIS\Parameters"
   | ft AllowFlowControlUnderDebugger
```

The return should be:

```
AllowFlowControlUnderDebugger
-----------------------------
                            1
```

Step 4: Test RDMA Connectivity

This step ensures the fabric is correctly configured and works in Native mode (Mode 1) operation. If Mode 1 doesn't work, Mode 2 and Mode 3 won't work either.

Step 4A: Create the directory C:\TEST

```
PS> cd \
PS> mkdir TEST
```

Step 4B: Gather the test tools to make testing easier

Download the DiskSpd.exe utility and extract into C:\TEST\ . The DiskSpd.exe utility can be found at https://gallery.technet.microsoft.com/DiskSpd-a-robust-storage-6cd2f223.

Download the Test-RDMA powershell script to C:\TEST\ . The Test-RDMA script can be found at https://github.com/Microsoft/SDN/blob/master/Diagnostics/Test-Rdma.ps1.

Step 4C: Ensure the NIC ports have RDMA enabled

For the Single-port configuration run

```
PS> Enable-NetAdapterRdma NIC1
```

For the Dual-port configuration run

```
PS> Enable-NetAdapterRdma NIC1,NIC2
```

Confirm that the NICs are now enabled for RDMA.

For the Single-port configuration run

```
PS> Get-NetAdapterRdma NIC1
```

The return should look like

```
Name InterfaceDescription           Enabled
---- ---------------------          -------
NIC1 Chelsio Network Adapter        True
```

Or perhaps

```
Name InterfaceDescription           Enabled
---- ---------------------          -------
NIC1 Mellanox ConnectX-4 VPI Adapter True
```

For the Dual-port configuration run

```
PS> Enable-NetAdapterRdma NIC1,NIC2
```

The return should look like

```
Name InterfaceDescription       Enabled
---- ---------------------      -------
NIC1 Chelsio Network Adapter    True
NIC2 Chelsio Network Adapter #2 True
```

Or perhaps

```
Name InterfaceDescription              Enabled
---- ---------------------             ------
NIC1 Mellanox ConnectX-4 VPI Adapter    True
NIC2 Mellanox ConnectX-4 VPI Adapter #2 True
```

Step 4D: Get the Interface Index and associated IP address of the RDMA NIC(s)

To get the Interface Index (ifIndex) and IPv4Address associated with your NICs, run:

```
PS> Get-NetIPConfiguration -InterfaceAlias "NIC*" |
    ft InterfaceAlias,InterfaceIndex,IPv4Address
```

The return should look like

(Single-port configuration)

```
InterfaceAlias InterfaceIndex IPv4Address
-------------- -------------- -----------
NIC1                        3 {192.168.1.3}
```

(Dual-port configuration)

```
InterfaceAlias InterfaceIndex IPv4Address
-------------- -------------- -----------
NIC1                        3 {192.168.1.3}
NIC2                        7 {192.168.2.3}
```

Step 4E: Check that SMB considers the RDMA interfaces as working

Now that you have the interface indexes (ifIndexes) of the RDMA-capable NICs, confirm that SMB also sees these interfaces as RDMA-capable.

```
PS C:\> Get-SmbClientNetworkInterface

Interface Index RSS Capable RDMA Capable Speed   IpAddresses
--------------- ----------- ------------ -----   -----------
3               True        True         40 Gbps {fe80::e14f:b55:b3dc:b03c, 192.168.1.3}
7               True        True         40 Gbps {fe80::9ce6:c07:9aab:d0f4, 192.168.2.3}
```

If for some reason the RDMA Capable column in the `Get-SmbClientNetworkInterface` output shows False, it may require a reboot of the host to get SMB to update the value.

Step 4F: Test the RDMA connectivity

Now that you have the local ifIndex, pass the ifIndex value to the Test-RDMA.ps1 script along with the IP address of the remote adapter on the same VLAN. (Reminder: NIC1, IPv4Address 192.168.1.3 is on the same VLAN as NIC1 on the other host which has IPv4Address 192.168.1.5. NIC2, IPv4Address 192.168.2.3 is on the same VLAN as NIC2 on the other host which has IPv4Address 192.168.2.5.)

If we are using RoCE as the RDMA protocol we run

```
PS> C:\TEST\Test-RDMA.PS1 -IfIndex 3 -IsRoCE $true -RemoteIpAddress 192.168.1.5
    -PathToDiskspd C:\TEST\Diskspd-v2.0.17\amd64fre\
```

The output should resemble this:

```
VERBOSE: Diskspd.exe found at C:\TEST\Diskspd-v2.0.17\amd64fre\\diskspd.exe
VERBOSE: The adapter NIC1 is a physical adapter
VERBOSE: Underlying adapter is RoCE. Checking if QoS/DCB/PFC is configured on each physical
adapter(s)
VERBOSE: QoS/DCB/PFC configuration is correct.
VERBOSE: RDMA configuration is correct.
VERBOSE: Checking if remote IP address, 192.168.1.5, is reachable.
VERBOSE: Remote IP 192.168.1.5 is reachable.
VERBOSE: Disabling RDMA on adapters that are not part of this test. RDMA will be enabled on them
later.
```

```
VERBOSE: Testing RDMA traffic now for. Traffic will be sent in a parallel job. Job details:
VERBOSE: 0 RDMA bytes written per second
VERBOSE: 0 RDMA bytes sent per second
VERBOSE: 662979201 RDMA bytes written per second
VERBOSE: 37561021 RDMA bytes sent per second
VERBOSE: 1023098948 RDMA bytes written per second
VERBOSE: 8901349 RDMA bytes sent per second
VERBOSE: Enabling RDMA on adapters that are not part of this test. RDMA was disabled on them
prior to sending RDMA traffic.
VERBOSE: RDMA traffic test SUCCESSFUL: RDMA traffic was sent to 192.168.1.5
```

If we are using iWARP as the RDMA protocol we run

```
PS> C:\TEST\Test-RDMA.PS1 -IfIndex 3 -IsRoCE $false -RemoteIpAddress 192.168.1.5
    -PathToDiskspd C:\TEST\Diskspd-v2.0.17\amd64fre\
```

The output should resemble this:

```
VERBOSE: Diskspd.exe found at c:\test\diskspd.exe
VERBOSE: The adapter C1 is a pNIC
VERBOSE: RDMA configuration is correct.
VERBOSE: Checking if remote IP address, 192.168.42.100, is reachable.
VERBOSE: Remote IP 192.168.42.100 is reachable.
VERBOSE: Disabling RDMA on adapters that are not part of this test. RDMA will be enabled on them
later.
VERBOSE: Testing RDMA traffic now for. Traffic will be sent in a parallel job. Job details:
VERBOSE: 881584596 RDMA bytes written per second
VERBOSE: 30395419 RDMA bytes sent per second
VERBOSE: 916403205 RDMA bytes written per second
VERBOSE: 32782735 RDMA bytes sent per second
VERBOSE: 854809218 RDMA bytes written per second
VERBOSE: 32463001 RDMA bytes sent per second
VERBOSE: 708712636 RDMA bytes written per second
VERBOSE: 37133310 RDMA bytes sent per second
VERBOSE: 855576900 RDMA bytes written per second
VERBOSE: 31471407 RDMA bytes sent per second
VERBOSE: 880404891 RDMA bytes written per second
VERBOSE: 32062793 RDMA bytes sent per second
VERBOSE: 840570441 RDMA bytes written per second
VERBOSE: 32459322 RDMA bytes sent per second
VERBOSE: Enabling RDMA on adapters that are not part of this test. RDMA was disabled on them
prior to sending RDMA traffic.
VERBOSE: RDMA traffic test SUCCESSFUL: RDMA traffic was sent to 192.168.42.100
```

If you are running the dual-port configuration, you should repeat this test with the second pair of NICs just to ensure the switch configuration is correct.

```
PS> C:\TEST\Test-RDMA.PS1 -IfIndex 7 -IsRoCE $true -RemoteIpAddress 192.168.2.5
    -PathToDiskspd C:\TEST\Diskspd-v2.0.17\amd64fre\
```

If this test fails check to ensure the network switch configuration aligns with the local host configuration.

Step 5: vSwitch creation and testing of Converged NIC

The next step is to create a vSwitch so that you can test the Converged NIC scenario that is used by e.g., Storage Spaces Direct (S2D). For both configurations we'll create the vSwitch in Switch Embedded Teaming (SET) mode even though the single NIC configuration doesn't require teaming.

There are three reasons we create the switch in SET mode:

1. There is no harm to having a team of one NIC;

2. In the single-NIC configuration the administrator may want to add a NIC later (if the switch is not created in SET mode, it can't be changed to SET mode later); and
3. Most importantly, it simplifies the writing of this guide.

Step 5a: Return the local host NICs to a state suitable for use with Hyper-V

Since we configured VLANs on the local NICs earlier, we will remove those VLANs. The Hyper-V switch requires the NICs to be in promiscuous mode (pass anything), so they can't have VLANs assigned. VLANs will be assigned at the virtual NIC level later. *Run these on Host A only, not on Host B. We'll fix the VLANs on Host B later.*

```
PS> Set-NetAdapterAdvancedProperty -Name NIC1 -RegistryKeyword VlanID
    -RegistryValue "0"
```

In the dual-port configuration also run

```
PS> Set-NetAdapterAdvancedProperty -Name NIC2 -RegistryKeyword VlanID
    -RegistryValue "0"
```

Step 5b: Create a vSwitch on a single NIC

Create the vSwitch. In the example below we enable three options, two of which can only be enabled at switch creation time. AllowManagementOS creates a host vNIC that we will use to do Converged NIC testing. EnableEmbeddedTeaming allows us to add another NIC to the switch later (for dual port configuration users). EnableIOV will enable us to create a virtual function (VF) in the guest and do Guest RDMA. (If you don't plan to continue to using Guest RDMA, leave off the -EnableIOV flag.)

```
PS> New-VMSwitch -Name RTest -NetAdapterName NIC1 -AllowManagementOS $true
    -EnableEmbeddedTeaming $true -EnableIov $true
```

The response should be something like:

```
Name  SwitchType NetAdapterInterfaceDescription
----  ---------- ------------------------------
RTest External   Teamed-Interface
```

Step 5c: Configure the Host vNIC for communication with Host B

The `-AllowManagementOS` flag on the `New-VMSwitch` cmdlet resulted in a new virtual adapter (vNIC) in the host partition. The next step is to configure that vNIC to communicate with Host B. Before we do that, however, we need to do a little housekeeping to keep things clear.

The `Get-NetAdapter` cmdlet shows us the vNIC:

```
PS> Get-NetAdapter
```

will return something like:

```
Name                 InterfaceDescription                ifIndex Status MacAddress         LinkSpeed
----                 --------------------                ------- ------ ----------         ---------
vEthernet (RTEST) Hyper-V Virtual Ethernet Adapter       27 Up     E4-1D-2D-07-40-71   40 Gbps
```

A Host vNIC is managed two ways: one representation is the NetAdapter view which operates on the "vEthernet (RTEST)" Name, the other mechanism is the VMNetworkAdapter view which drops the "vEthernet" prefix and simply uses the vSwitch name. The VMNetworkAdapter view allows for setting some vNIC properties that are not accessible via the NetAdapter view.

```
PS> Get-VMNetworkAdapter -ManagementOS
```

which will return something like

```
Name   IsManagementOs VMName SwitchName MacAddress   Status IPAddresses
----   -------------- ------ ---------- ----------   ------ -----------
RTEST  True                  RTEST      E41D2D074071 {Ok}
```

The first vNIC exposed in the host partition is traditionally used for management (e.g., remote access, etc.) while this guide is setting up the host to use RDMA from the host. While we could work with having the interface name the same as the switch name, it's more convenient to rename the host vNIC to a meaningful name.

```
PS> Rename-VMNetworkAdapter -ManagementOS -VMNetworkAdapterName Mgmt
```

Since we want to use a vNIC to carry SMB traffic, we create a new vNIC for that.

```
PS> Add-VMNetworkAdapter -ManagementOS -VMNetworkAdapterName SMB1
```

Let's review what the host vNICs are now:

```
PS> Get-VMNetworkAdapter -ManagementOS
```

The ooutput should look like:

```
Name IsManagementOs VMName SwitchName MacAddress   Status IPAddresses
---- -------------- ------ ---------- ----------   ------ -----------
Mgmt True                  Rtest      00155D579802 {Ok}
SMB1 True                  Rtest      00155D579803 {Ok}
```

Since the Get-NetAdapter and Get-VMNetworkAdapter cmdlet families report different names for these vNIC interfaces, it simplifies our lives to make them identical. We rename the Get-NetAdapter view of the vNICs.

```
PS> Rename-NetAdapter "*Mgmt*" Mgmt
PS> Rename-NetAdapter "*SMB1*" SMB1
```

Verify the name changes worked as expected:

```
PS> Get-NetAdapter

Name InterfaceDescription                ifIndex Status  MacAddress         LinkSpeed
---- --------------------                ------- ------  ----------         ---------
SMB1 Hyper-V Virtual Ethernet Adapter #2       7 Up      00-15-5D-57-98-03    40 Gbps
Mgmt Hyper-V Virtual Ethernet Adapter          9 Up      00-15-5D-57-98-02    40 Gbps
```

Finally, let's assign the IP addresses we want to the host vNICs: a new one to the Mgmt interface, and the one we were using already to the SMB interface.

```
PS> New-NetIPAddress -InterfaceAlias Mgmt -IPAddress 192.168.1.2 -PrefixLength 24
PS> New-NetIPAddress -InterfaceAlias SMB1 -IPAddress 192.168.1.3 -PrefixLength 24
```

Finally, we need to add the VLAN tag back to the SMB interface so it can communicate with the Host B network adapter.

```
PS> Set-VMNetworkAdapterVlan -ManagementOS -Access -VlanId 101
          -VMNetworkAdapterName SMB01
```

At the end of this step the configuration of the hosts looks like one of the figures below:

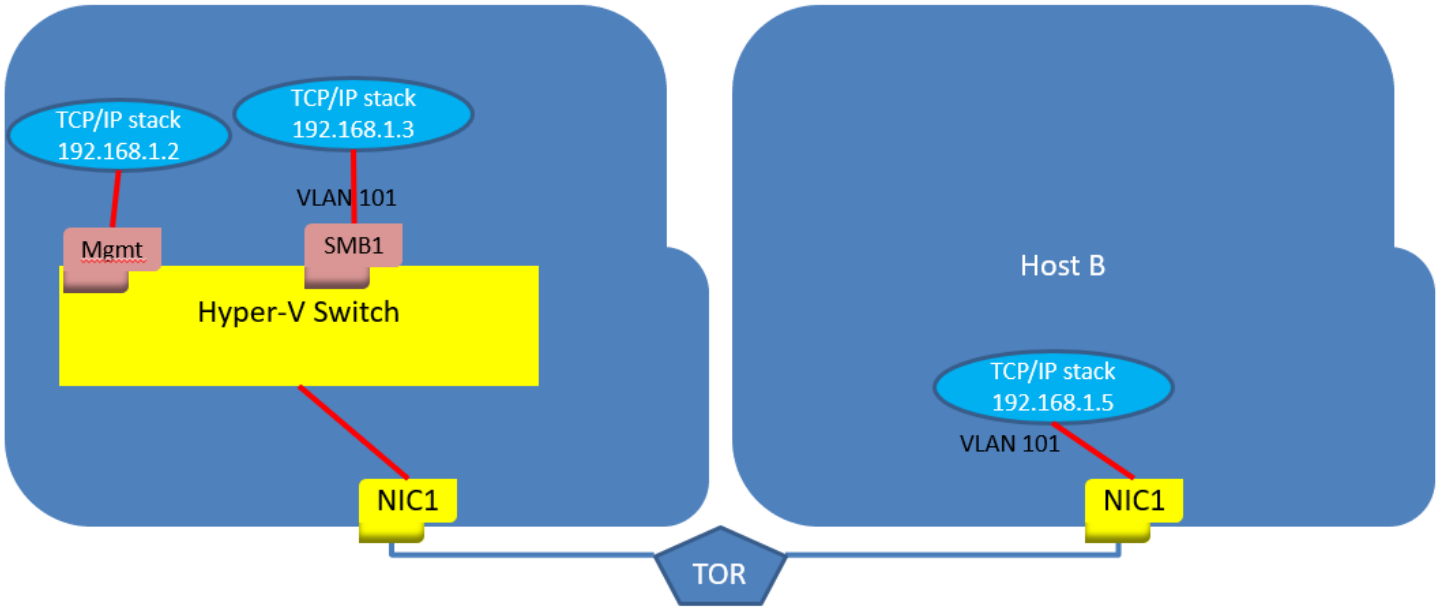

Figure 5 - After switch creation (single-port configuration)

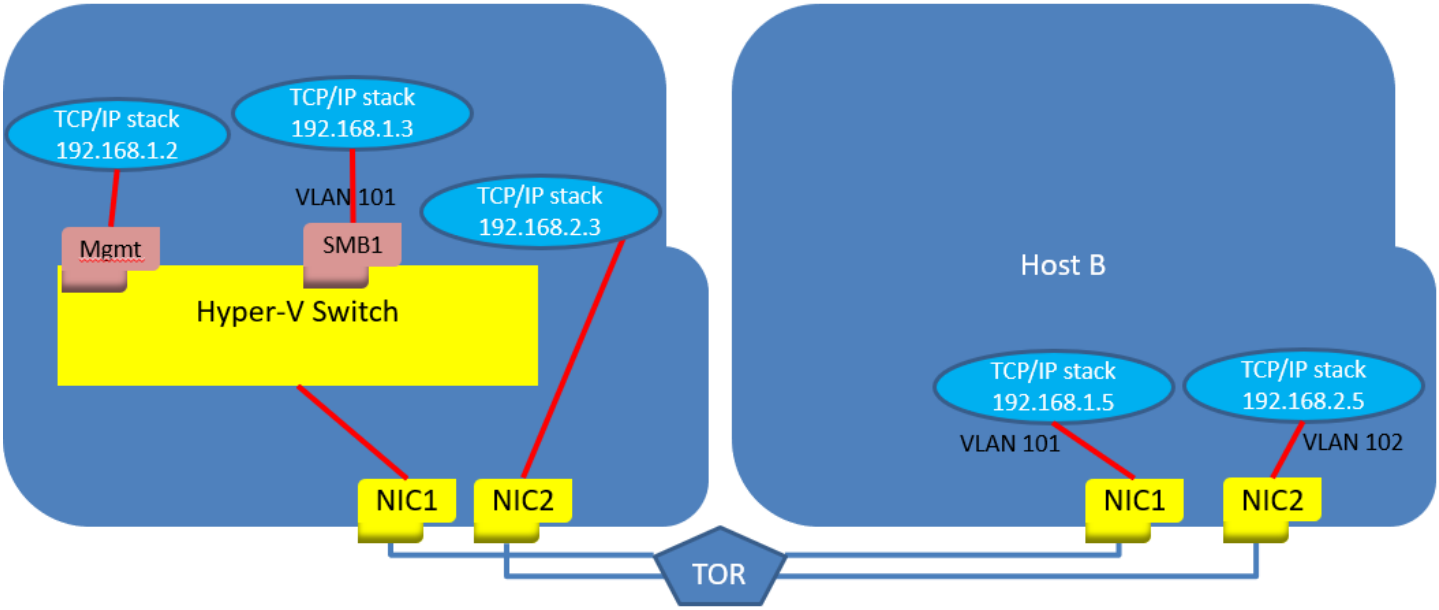

Figure 6 - After switch creation (dual-port configuration)

Step 5d: Assign VLANs to the Host vNIC

In a previous step, we removed the VLANs from the physical adapter to allow the pNIC to send all VLANs being sent across it. This is important because, for example, tenant VMs may have different VLANs then your storage networks.

Now we'll assign the specific VLAN that the host vNIC will use for SMB traffic. Let's first check to see if the VLAN is assigned.

Note: There are multiple mechanisms available to assign a VLAN to a host vNIC. In this section we will show one method of assigning the VLANs. For more information on the other mechanism, please see **Appendix 3: VLAN Management in Windows Server 2016 Version 1709**.

Check for the presence of a VLAN on the host vNIC. If the correct one (VLAN 101 in our example) isn't present, add it using Set-VMNetworkAdapterVlan as shown below:

```
PS> Set-VMNetworkAdapterVlan -ManagementOS -Access -VlanId 101
          -VMNetworkAdapterName SMB01
```

Step 5e: Test TCP-IP connectivity using the Host vNIC

Since we removed the pNIC's VLAN setting and added the vNIC's VLAN setting, we should make sure we can still communicate.

```
PS> Test-NetConnection 192.168.1.5

 ComputerName            : 192.168.1.5
 RemoteAddress           : 192.168.1.5
 InterfaceAlias          : vEthernet (GuestRdma)
 SourceAddress           : 192.168.1.3
 PingSucceeded           : True
 PingReplyDetails (RTT) : 0 ms
```

Observe the result to make sure that "PingSucceeded" shows "True". If it doesn't show "True", check the VLAN settings of the pNIC and the vNIC to make sure they were administered correctly (pNIC should have no VLAN set, vNIC should be set to 101).

```
PS C:\test> Get-NetAdapterAdvancedProperty NIC1 |ft VLANID

 VLANID
 ------
 (blank line)

PS C:\Users\Administrator> Get-VMNetworkAdapterVlan -ManagementOS

    VMName VMNetworkAdapterName Mode     VlanList
    ------ -------------------- ----     --------
           Mgmt                 Untagged
           SMB01                Access   101
```

Step 5e: Enable IeeePriorityTag on the host vNICs

The AllowIeeePriorityTag is off by default for all vNIC types (both host vNICs & vmNICs). This setting, however, only matters for traffic that flows through the vswitch. The vswitch uses this flag to determine whether it should preserve or zero-out the 802.1p priority value in each packet that it sees.

Traffic that is **not** affected by this flag would be:

- Host vNIC's RDMA traffic. This traffic bypasses the vswitch, and go directly to the NIC
- Traffic from IOV VMs. This traffic goes directly to the NIC, via the IOV vPort

Therefore, we recommend these settings to ensure that cluster heartbeat traffic priority tags are preserved as follows:

```
PS> Set-VMNetworkAdapter -ManagementOS -Name SMB1 -IeeePriorityTag on
```

Step 5f: Test RDMA connectivity using the Host vNIC

If you check the Host vNIC now to see if it is configured for RDMA, the answer will probably be False. I.e.,

```
PS> Get-NetAdapterRdma SMB1
```

should return something like:

```
 Name InterfaceDescription             Enabled
 ---- --------------------             -------
 SMB1 Hyper-V Virtual Ethernet Adapter False
```

To enable the vNIC for RDMA operation, enable RDMA:

```
PS> Enable-NetAdapterRdma *SMB1*
```

Check the result by running the Get-NetAdapterRdma cmdlet again:

```
PS> Get-NetAdapterRdma *SMB1*
```

should now return:

```
Name InterfaceDescription             Enabled
---- --------------------             -------
SMB1 Hyper-V Virtual Ethernet Adapter True
```

Now we can test the vNIC to see if RDMA is working with Host B.

Get the ifIndex of the vNIC:

```
PS> Get-NetAdapter
```

should return something like we saw in step 6c:

```
Name InterfaceDescription              ifIndex Status MacAddress         LinkSpeed
---- --------------------              ------- ------ ----------         ---------
SMB1 Hyper-V Virtual Ethernet Adapter       27 Up     E4-1D-2D-07-40-71   40 Gbps
```

We know the Host B NIC is configured with IPv4Address 192.168.1.5.

If we are running with RoCE as the RDMA protocol, we run:

```
PS> C:\TEST\Test-RDMA.PS1 -IfIndex 27 -IsRoCE $true -RemoteIpAddress 192.168.1.5
    -PathToDiskspd C:\TEST\Diskspd-v2.0.17\amd64fre\
```

If all the configuration was correct we should see:

```
PathToDiskspd C:\TEST\Diskspd-v2.0.17\amd64fre\
VERBOSE: Diskspd.exe found at C:\TEST\Diskspd-v2.0.17\amd64fre\\diskspd.exe
VERBOSE: The adapter vEthernet (RTEST) is a virtual adapter
VERBOSE: Retrieving vSwitch bound to the virtual adapter
VERBOSE: Found vSwitch: RTEST
VERBOSE: Found the following physical adapter(s) bound to vSwitch: M1
VERBOSE: Underlying adapter is RoCE. Checking if QoS/DCB/PFC is configured on each physical
adapter(s)
VERBOSE: QoS/DCB/PFC configuration is correct.
VERBOSE: RDMA configuration is correct.
VERBOSE: Remote IP 192.168.1.5 is reachable.
VERBOSE: Disabling RDMA on adapters that are not part of this test. RDMA will be enabled on them
later.
VERBOSE: Testing RDMA traffic now for. Traffic will be sent in a parallel job. Job details:
VERBOSE: 9162492 RDMA bytes sent per second
VERBOSE: 938797258 RDMA bytes written per second
VERBOSE: 34621865 RDMA bytes sent per second
VERBOSE: 933572610 RDMA bytes written per second
VERBOSE: 35035861 RDMA bytes sent per second
VERBOSE: Enabling RDMA on adapters that are not part of this test. RDMA was disabled on them
prior to sending RDMA traffic.
VERBOSE: RDMA traffic test SUCCESSFUL: RDMA traffic was sent to 192.168.1.5
```

If we are running with iWARP as the RDMA protocol we run:

```
PS> C:\TEST\Test-RDMA.PS1 -IfIndex 27 -IsRoCE $false -RemoteIpAddress 192.168.1.5
    -PathToDiskspd C:\TEST\Diskspd-v2.0.17\amd64fre\

 VERBOSE: Diskspd.exe found at c:\test\Diskspd-v2.0.17\amd64fre\diskspd.exe
 VERBOSE: The adapter SMB1 is a vNIC
 VERBOSE: Retrieving vSwitch bound to the virtual adapter
 VERBOSE: Found vSwitch: Rtest
 VERBOSE: Found the following physical adapter(s) bound to vSwitch: C1
 VERBOSE: RDMA configuration is correct.
 VERBOSE: Remote IP 192.168.1.5 is reachable.
 VERBOSE: Disabling RDMA on adapters that are not part of this test. RDMA will be enabled on them
 later.
 VERBOSE: Testing RDMA traffic. Traffic will be sent in a background job. Job details:
 VERBOSE: 854055239 RDMA bytes written per second
 VERBOSE: 32234131 RDMA bytes sent per second
 VERBOSE: 860933980 RDMA bytes written per second
 VERBOSE: 30619357 RDMA bytes sent per second
 VERBOSE: 861202064 RDMA bytes written per second
 VERBOSE: 27255016 RDMA bytes sent per second
 VERBOSE: Enabling RDMA on adapters that are not part of this test. RDMA was disabled on them
 prior to sending RDMA traffic.
 SUCCESS: RDMA traffic test SUCCESSFUL: RDMA traffic was sent to 192.168.1.5
```

Step 5g: (Dual-port configuration) Add and test the second port

Now that the single-port configuration is working we can add the second port.

To add a NIC to a SET team, use Add-VMSwitchTeamMember. (Information about managing Switch Embedded Teams can be found in section 4.2 of the Windows Server 2016 NIC and Switch Embedded Teaming User Guide found at https://gallery.technet.microsoft.com/Windows-Server-2016-839cb607.

```
PS> Add-VMSwitchTeamMember -VMSwitchName RTEST -NetAdapterName NIC2
```

Having two pNICs in the team is interesting, but to test RDMA on both we also need an additional host vNIC for SMB to use in SMB-Multichannel mode. We will add a vNIC names “SMB2”, assign the IP address that was used on NIC2 earlier in this exercise, and tag it with VLAN value of 102.

```
PS> Add-VMNetworkAdapter -ManagementOS -Name SMB2
PS> Rename-NetAdapter "*SMB2*" SMB2
PS> New-NetIPAddress -InterfaceAlias SMB2 -IPAddress 192.168.1.4
PS> Set-VMNetworkAdapterVlan -VMNetworkAdapterName SMB2 -VlanId "101" -Access
    -ManagementOS
PS> Set-VMNetworkAdapter -ManagementOS -Name SMB2 -IeeePriorityTag on
```

For best performance map the two SMB vNICs to the two pNICs. Since affinities between vNICs and physical NIC resources are random when the operating systems chooses them, it’s best to override the random assignment and make sure the two SMB interfaces don’t end up mapped to the same underlying pNIC.

```
PS> Set-VMNetworkAdapterTeamMapping -ManagementOS -VMNetworkAdapterName SMB1
    -PhysicalNetAdapterName NIC1
PS> Set-VMNetworkAdapterTeamMapping -ManagementOS -VMNetworkAdapterName SMB2
    -PhysicalNetAdapterName NIC2
```

Again, we need to get the ifIndex of the new vNIC:

```
PS> Get-NetAdapter
```

should return something like we saw in step 6c:

```
Name             InterfaceDescription             ifIndex Status MacAddress         LinkSpeed
----             --------------------             ------- ------ ----------         ---------
SMB1             Hyper-V Virtual Ethernet Adapter      27 Up     E4-1D-2D-07-40-71   40 Gbps
SMB2             Hyper-V Virtual Ethernet Adapter      41 Up     E4-1D-2D-07-40-72   40 Gbps
```

Now we can test to see if the new interface is also working for RDMA traffic.

```
PS> C:\TEST\Test-RDMA.PS1 -IfIndex 41 -IsRoCE $false -RemoteIpAddress 192.168.1.5
    -PathToDiskspd C:\TEST\Diskspd-v2.0.17\amd64fre\
```

The results should resemble those in step 6E.

Step 6 – Enabling SR-IOV for Guest RDMA

Guest RDMA is only supported starting in Windows Server 1709. The following steps will not work on any earlier edition of Windows Server.

Step 6A – Update the Network Card Drivers on the Host

Since none of the inbox drivers in Windows Server 1709 support Guest RDMA, you need to download and install the latest drivers from your network card vendor. Make sure the release notes for the driver you install indicates support for Guest RDMA. Once the new drivers are installed we can proceed. A reboot may be necessary as part of driver installation.

When you have installed the latest drivers, confirm they are installed by running:

```
PS> Get-NetAdapter NIC1 | fl
```

and check that the driver version indicated is the one you thought you were installing.

Step 6B – Enable SR-IOV on the Host Adapters

Back in Step 5B we created a Hyper-V switch and we told the system that -IovEnabled was $true. So we have SR-IOV enabled in the vSwitch already. Now we need to create a VM and make it SR-IOV and RDMA capable. Before we do that let's make sure the adapter(s) that are bound to the vSwitch have SR-IOV turned on.

To turn on SR-IOV in the single NIC Port configuration:

```
PS> Enable-NetAdapterSriov NIC1
```

To turn on SR-IOV in the dual NIC Port configuration:

```
PS> Enable-NetAdapterSriov NIC1,NIC2
```

Step 6C – Create and start a VM

Before you can create the VM you need to put a VHDX in a known location for the VM to use. The most common place is in the default directory: `c:\Users\Public\Public Documents\Hyper-V\Virtual hard disks\`. For ease of referencing it later, name it `VM1.vhdx`.

Now we can create the VM.

```
PS> new-vm VM1 -Generation 2 -switchname RTest -vhdpath
    "C:\users\public\documents\hyper-v\virtual hard disks\VM1.vhdx
```

Before we start the VM we need to set the VM's network interface into SR-IOV and RDMA-capable mode.

```
PS> set-VMNetworkAdapter VM1 -IovWeight 100 -IovQueuePairsRequested 8 #enable IOV
PS> set-VMNetworkAdapterRdma VM1 -RdmaWeight 100                       #enable RDMA
```

Now we can start the VM:

```
PS> start-VM VM1
```

Step 6D – Log into the VM and complete the Out-of-Box (OOB) Experience

Log into the VM and complete the initial settings for language, etc. For the purpose of this guide we'll assume you know how to do that.

Give your guest network interface an IP address in the same space as your host management vNIC:

```
PS> Net-NetIPAddress -IPAddress 192.168.1.10 -PrefixLength 24 -InterfaceAlias Ethernet
```

Step 6E – Copy the network drivers into the VM

Since the inbox drivers for Windows Server 1709 do not support Guest RDMA, you must get the latest drivers from the network adapter vendor and install them in the guest. Follow the guidance of your network vendor.

By default, if the network driver in the host supports SR-IOV, then when the guest becomes SR-IOV enabled the host will copy the driver to the guest for you. If in doubt, check to make sure the driver you are running for the VF is the latest available.

Once everything is installed running Get-NetAdapter in the guest should show (Chelsio example):

```
PS> Get-NetAdapter
```

should return

```
Name       Interface Description             ifIndex Status MacAddress         LinkSpeed
----       ---------------------             ------- ------ ----------         ---------
Ethernet   Microsoft Hyper-V Network Adapter       2 Up     00-15-5D-2A-63-00   40 Gbps
Ethernet 2 Chelsio VF Network Adapter             7 Up     00-15-5D-2A-63-00    7 Gbps
```

Step 6F – test connectivity

Now that the VF is installed in the VM, let's test connectivity to Host B.

```
PS> Test-NetConnection 192.168.1.5

ComputerName            : 192.168.1.5
RemoteAddress           : 192.168.1.5
InterfaceAlias          : Ethernet
SourceAddress           : 192.168.1.10
PingSucceeded           : True
PingReplyDetails (RTT)  : 0 ms
```

Step 7 – Enabling Guest RDMA

Step 7A – Enable the vmNIC for RDMA

Inside the guest, enable RDMA on the vmNIC and VF:

```
PS> Enable-NetAdapterRdma Ethernet,"Ethernet 2"
```

Check to see that it worked.

```
PS> Get-NetAdapterRdma

Name        Interface Description               Enabled
```

```
----        ---------------------               -------
Ethernet 2  Chelsio VF Network Adapter          True
Ethernet    Microsoft Hyper-V Network Adapter   True
```

Now use Get-NetAdapter one more time to get the Interface Indexes (IfIndex) of the adapter and the VF.

```
PS> Get-NetAdapter

Name       Interface Description             ifIndex Status MacAddress         LinkSpeed
----       ---------------------             ------- ------ ----------         ---------
Ethernet   Microsoft Hyper-V Network Adapter       2 Up     00-15-5D-2A-63-00   40 Gbps
Ethernet 2 Chelsio VF Network Adapter             7 Up     00-15-5D-2A-63-00    7 Gbps
```

Step 7B – Test Guest RDMA

The Test-Rdma script has an additional parameter for testing in the guest. Specifically, it needs the IfIndex of the VF (Parameter: VFIndex). *Note: In the guest the "IsRoCE" flag is ignored and can be set to any value.*

```
PS> C:\TEST\Test-RDMA.PS1 -IfIndex 3 -IsRoCE $false -RemoteIpAddress 192.168.1.5
    -PathToDiskspd C:\TEST\Diskspd-v2.0.17\amd64fre\ -VFIndex 2
```

The output should resemble:

```
VERBOSE: Diskspd.exe found at c:\test\Diskspd-v2.0.17\amd64fre\diskspd.exe
VERBOSE: The adapter SMB1 is a vNIC
CAUTION: Guest Virtual NIC being tested, Guest can't check host adapter settings.
VERBOSE: Retrieving vSwitch bound to the virtual adapter
VERBOSE: Found vSwitch: Rtest
VERBOSE: Found the following physical adapter(s) bound to vSwitch: C1
VERBOSE: RDMA configuration is correct.
VERBOSE: Remote IP 192.168.1.5 is reachable.
VERBOSE: Disabling RDMA on adapters that are not part of this test. RDMA will be enabled on them
later.
VERBOSE: Testing RDMA traffic. Traffic will be sent in a background job. Job details:
VERBOSE: 854055239 RDMA bytes written per second
VERBOSE: 32234131 RDMA bytes sent per second
VERBOSE: 860933980 RDMA bytes written per second
VERBOSE: 30619357 RDMA bytes sent per second
VERBOSE: 861202064 RDMA bytes written per second
VERBOSE: 27255016 RDMA bytes sent per second
VERBOSE: Enabling RDMA on adapters that are not part of this test. RDMA was disabled on them
prior to sending RDMA traffic.
SUCCESS: RDMA traffic test SUCCESSFUL: RDMA traffic was sent to 192.168.1.5
```

Appendix 2: Tools that may help

In addition to the tools mentioned at Step 4B: Gather the test tools to make testing easier, the following tools may assist in configuring switches and NICs for RDMA:

1. Validate-DCB – Configuration validation tool for the Windows hosts
   ```
   Install-Module Validate-DCB
   ```

2. Test-NetStack – Network stress testing simulator
   ```
   Install-Module Test-NetStack
   ```

3. The sample Switch configuration scripts found at
   https://github.com/Microsoft/SDN/tree/master/SwitchConfigExamples

4. The User Guides and Release Notes for your NIC and switch vendor.

Appendix 3: VLAN Management in Windows Server 2016 Version 1709

In Windows Server 2016 the VLAN assigned to the physical NIC is not copied to the host vNIC when the switch was created. In Windows Server 1709 the VLAN is copied to the host vNIC when the switch is created. Since we removed the VLANs from the physical hosts in step 6a you need to assign the VLAN value to the host vNIC. First check to see whether or not the VLAN is assigned.

Unfortunately, Windows Server stores VLAN information in two different places and using the wrong cmdlet will cause you to see incorrect information. To get a complete picture, you need to run two different cmdlets:

```
PS> Get-VMNetworkAdapterVlan -ManagementOS
PS> Get-VMNetworkAdapterIsolation -ManagementOS
```

As an example, if we set the VLAN using Set-VMNetworkAdapterIsolation and then use Get-VMNetworkAdapterVlan we will see incorrect information. To wit, here is a three cmdlet sequence:

```
# Set the VLAN for host vNIC to value 101
PS C:\WINDOWS\system32> set-VMNetworkAdapterIsolation -ManagementOS -IsolationMode Vlan
                        -DefaultIsolationID 101

# Check the value using Get-VMNetworkAdapterVlan
PS C:\WINDOWS\system32> Get-VMNetworkAdapterVlan -ManagementOS

 VMName VMNetworkAdapterName Mode     VlanList
 ------ -------------------- ----     --------
        SMB1                 Untagged

# Previous cmdlet says traffic is untagged, but now check with Get-VMNetworkAdapterIsolation

PS C:\WINDOWS\system32> Get-VMNetworkAdapterIsolation -ManagementOS
 IsolationMode        : Vlan
 AllowUntaggedTraffic : False
 DefaultIsolationID   : 101
 MultiTenantStack     : Off
 ParentAdapter        : VMInternalNetworkAdapter, Name = 'SMB1'
 IsTemplate           : False

# With this cmdlet the vNIC reports it has a VLAN assigned
```

The good thing is that it doesn't matter whether you use Set-VMNetworkAdapterVlan or Set-VMNetworkAdapterIsolation as they both work (except when the SDN-extension is used). But if you want to change or remove the VLAN you must use the same cmdlet family you used to assign the VLAN.

Check for the presence of a VLAN on the host vNIC. If the correct one (VLAN 101 in our example) isn't present, add it using Set-VMNetworkAdapterIsolation as shown below:

```
PS> set-VMNetworkAdapterIsolation -ManagementOS -IsolationMode Vlan
    -DefaultIsolationID 101
```