LAB GUIDE

Understanding and Configuring Data Center Bridging Solutions on Aruba CX Switches

a Hewlett Packard Enterprise company

!!IMPORTANT!!

*******NOT ALL FEATURES DISCUSSED IN THIS GUIDE CAN BE CONFIGURED IN THE EVE-NG ENVIRONMENT. THE GUIDE WILL DISCUSS HOW TO CONFIGURE THESE FEATURES (DCBX + PFC + ATTACHED HOST CONFIGS), HOWEVER YOU WILL NOT BE ABLE TO APPLY THEM TO THE EVE-NG SWITCHES AT THIS TIME*******

THIS GUIDE ASSUMES THAT THE AOS-CX OVA HAS BEEN INSTALLED AND WORKS IN GNS3 OR EVE-NG. PLEASE REFER TO GNS3/EVE-NG INITIAL SETUP LABS IF REQUIRED.

TABLE OF CONTENTS

Lab Objective

The objective of this lab is to get hands on experience deploying a RoCEv2 solution. Unfortunately, at this time not all features required for these solutions are supported within the EVE-NG environment. However, the Lab guide calls those sections out and users should be able to get a good feel of the overall solution even though some features cannot be configured. Stay tuned for updates to this lab as new versions of EVE-NG get produced.

Lab Network Layout

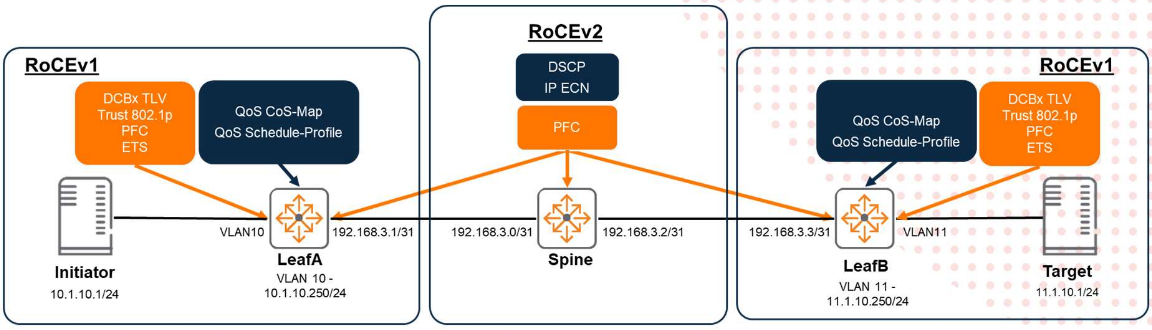

Figure 1. Lab Layout

RDMA Overview

Within today's enterprise, servers are required to handle massive amounts of data while providing 100% uptime. Over the years, adoption of server virtualization, big data analytics and the proliferation of mobile devices have continued to stress computing infrastructures. Users have noticed applications taking longer than they should to execute. When corporate and other users notice slowing of the systems, they become less productive. Many times, this type of delay happens because large amounts of data has to be processed by the CPU which then has to move from buffer spaces, down through the TCP stack, onto the wire between servers of the enterprise, and then back up the stack again on the other side. This transfer can cause the CPU to slow down processing of other tasks as the CPU is busy. Adding more servers may increase CPU processing power but it is not addressing the fact that the CPUs are getting over-utilized and it runs counter to the goal of doing more with less within today's organizations.

Remote Direct Memory Access (RDMA) Solution

RDMA enables the movement of data between servers with very little CPU involvement.

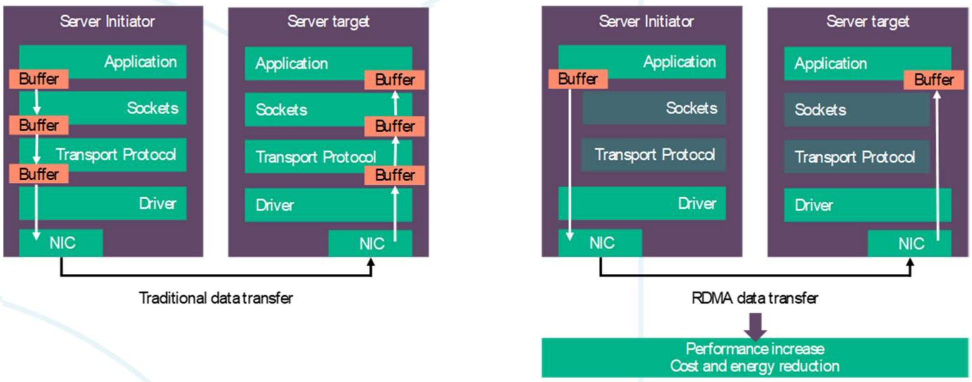

Figure 2. RDMA bypassing OS stack

Without RDMA, traditional movement of data utilizes TCP/IP buffer copies and significant overhead. Applications reply on the OS stack to move data from memory, virtual buffers through the stack, onto the wire, across the wire, and then again back up the wire. The receiving OS must retrieve the data and place it directly in the application(s)' virtual buffer space which leads to the CPU being occupied for the entire duration of read and write operations and is unavailable to perform other work.

RDMA solutions are able to bypass the OS stack. The OS is used to just establish a channel which applications use to directly exchange messages on. A network adapter transfers data directly to and from application memory eliminating the need to copy data between application memory and the data buffers within the operating system. Such communication requires no work to be done by CPUs, caches or context switches, and transfers continue in parallel with other system operations. When an application performs an RDMA Read or Write request, the application data is delivered directly to the network, reducing latency, CPU overhead, and enabling fast transfer of data.

Users running RDMA applications on an Ethernet network can see application performance improvements that derive from the offloading of data movement and higher availability of CPU resources to applications. Shifting the chore of data movement from the CPU makes both data movement and execution of applications more efficient. RDMA delivers performance and efficiency gains that are not available from any other communications protocol, including low latency, improved resource utilization, flexible resource allocation, fabric unification and scalability. Greater server productivity lowers the need for additional servers and lowers the total cost of ownership.

- RDMA is the direct read from or write to an application's memory
- Hardware offload moves data faster with significantly less overhead allowing the CPU to work on other applications
- CPU initiates the transfer and processes other operations while the transfer is in progress
- Ultra–low latency through stack bypass and copy avoidance
- Reduces CPU utilization
- Reduces memory bandwidth bottlenecks
- Enables high bandwidth and I/O utilization
- RDMA is useful when CPU cannot keep up and needs to perform other useful work

RDMA transport technologies

There are three main transport types of solutions that can be used to transport RDMA over an Ethernet network.

- InfiniBand (IB)
  - Protocol which supports RDMA natively from the beginning
  - Requires dedicated NICs and switches that supports this technology
  - Pure InfiniBand solutions can provide high performance at cost of dual networking fabrics
- Internet Wide Area RDMA Protocol (iWARP)
  - RDMA over TCP
  - iWARP defined by IETF and uses the TCP/IP stack in order to be compatible with any Ethernet/IP infrastructure
  - Data Center Bridging (DCB) Ethernet helps avoid congestion, but it is not required by the standard
  - Supports offload to the NIC
  - Goes up the TCP/IP stack to achieve protection for loss
- RDMA Over Converged Ethernet (RoCE)
  - Data Center Bridging (DCB) Ethernet should be configured, but it is not required by the standard
  - Requires a DCB switch to provide for a lossless fabric
  - NICs should support RoCE and offloading
  - Lower level Ethernet mechanisms used to protect for loss:
    - Priority Flow Control (PFC) to stave off loss
    - Enhanced transmission selection (ETS) to protect traffic classes (TC)
  - Uses upper InfiniBand layers in case of need for retransmission to recover from loss.

The Aruba CX solutions can support the RoCE based version and therefore the following content will focus on RoCE based networking.

What is RoCE?

RoCE is a network protocol that allows RDMA over Converged Ethernet, or RoCE (pronounced “rocky”). This critical technology is now expanding into enterprise markets where Ethernet networks are ubiquitous. RoCE is geared for high performance within an advanced data center architecture eliminating dedicated storage area networks (SANs) by converging compute, network, and storage onto a single fabric. Utilizing advanced reliable Ethernet and DCB with RDMA techniques, RoCE provides lower CPU overhead and increases enterprise data center application performance.

Today’s dynamic evolving enterprise, let it be local, remote cloud, or hybrid data centers, require high performance technologies like RoCE to support increasingly data intensive applications and the move to hyper converged scale-out solutions which leverage distributed computing/storage models.

Aruba Networks currently supports RoCE solutions using the CX 8325/8360/8400 Switches.

The benefits of implementing RoCE are:

- Lower cost of ownership
- Greater return on investment that span traditional and today’s hyper converged infrastructures
- Reduces CPU over utilization
- Reduces Host Memory Bottlenecks
- Helps to better leverage the storage media evolution which has brought 10,000x performance improvement factor
- Offloads memory access process
- Increases throughput and lowers latency between compute and storage systems

RoCE aspects

The initial RoCE v1 solution simply replaced the IB Link Layer with an Ethernet link layer. In this solution RoCE was a Layer 2 based Ethernet solution. The latest version of RoCE, which is called RoCE v2, replaced the IB Network layer with a standard IP and UDP Header so traffic is routable now.

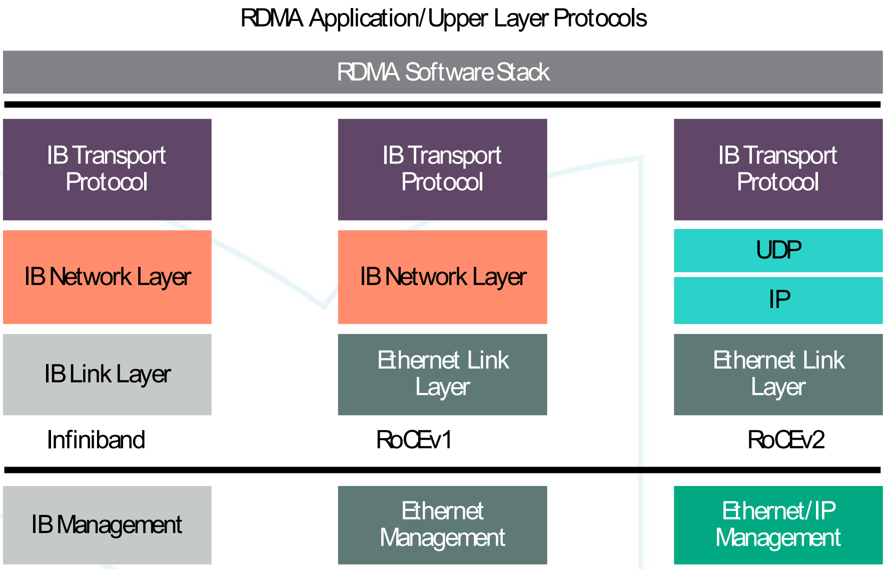

Figure 3. IB/RoCE evolution

The InfiniBand Annexes for RoCEv1 and RoCEv2 do not actually mandate that no loss occur on the Ethernet network. However, even though not mandatory or required by the standard, RoCE based solutions will perform poorly in lossy fabrics because in times of congestion devices will start to drop packets, causing retransmissions, reducing throughput, and increasing delay. In these situations, the solution will simply not get the RDMA benefits needed to reduce CPU load and latency.

Aruba Networks recommends, in production environments, RoCE based solutions should be deployed as a lossless fabric.

RoCE design recommendations

Lossless network fabrics

One of the objectives when designing network solutions that incorporate RoCE is to deploy a lossless fabric. Even though the RoCE standards do not necessarily demand lossless networks they will perform much better when deployed in a lossless manner, especially under heavy congestion. With that in mind, Aruba Networks recommends that a lossless fabric be considered a requirement for RoCE implementations.

Lossless fabrics can be built on Ethernet fabrics by leveraging the following DCB protocols.

- **Priority- based flow control (PFC):** IEEE standard 802.1Qbb, is a link-level flow control mechanism. The flow control mechanism is similar to that used by IEEE 802.3x Ethernet PAUSE, but it operates on individual priorities. Instead of pausing all traffic on a link, PFC allows you to selectively pause traffic according to its class.
- **Enhanced Transmission Selection (ETS):** ETS helps to ensure that when using PFC, lossless traffic does not get continually paused because other types of traffic are using the whole bandwidth of a link. With ETS you can tell an interface that a certain percentage of the bandwidth is guaranteed for each traffic type. This way each traffic type gets a minimum amount of bandwidth.
- **Data Center Bridging Exchange (DCBX):** This protocol helps to ensure that the NIC and the Switch are configured properly. DCBx is a discovery and capability exchange protocol which discover peers and exchanges configuration information. The protocol allows auto exchange of Ethernet parameters and discovery functions between switches and endpoints. If the server NIC has the DCBx willing bit turned on then after you configure the switch with the needed DCB and traffic marking rules, then DCBx will ensure the server NIC also knows how to mark and treat traffic on that link.
- **Quantized Congestion Notification (QCN):** This protocol provides a means for a switch to notify a source that there is congestion on the network. The source will then reduce the flow of traffic. This help to keep the critical traffic flowing while also reducing the need for pauses. This is only supported in pure Layer 2 environments and seen very rarely now that RoCEv2 is the predominant RoCE solution.
- **IP Explicit Congestion Notification (IP ECN):** IP explicit congestion notification is not officially part of the DCB protocol suite; however, it can be leveraged to enable end-to-end congestion notification between two endpoints on TCP/IP based networks. The two endpoints are an ECN-enabler sender and an ECN-enabler receiver. ECN must be enabled on both endpoints and on all the intermediate devices between endpoints for ECN to work properly. ECN notifies networks about congestion with the goal of reducing packet loss and delay by marking the sending device to decrease the transmission rate until congestion clears, without pausing packets.

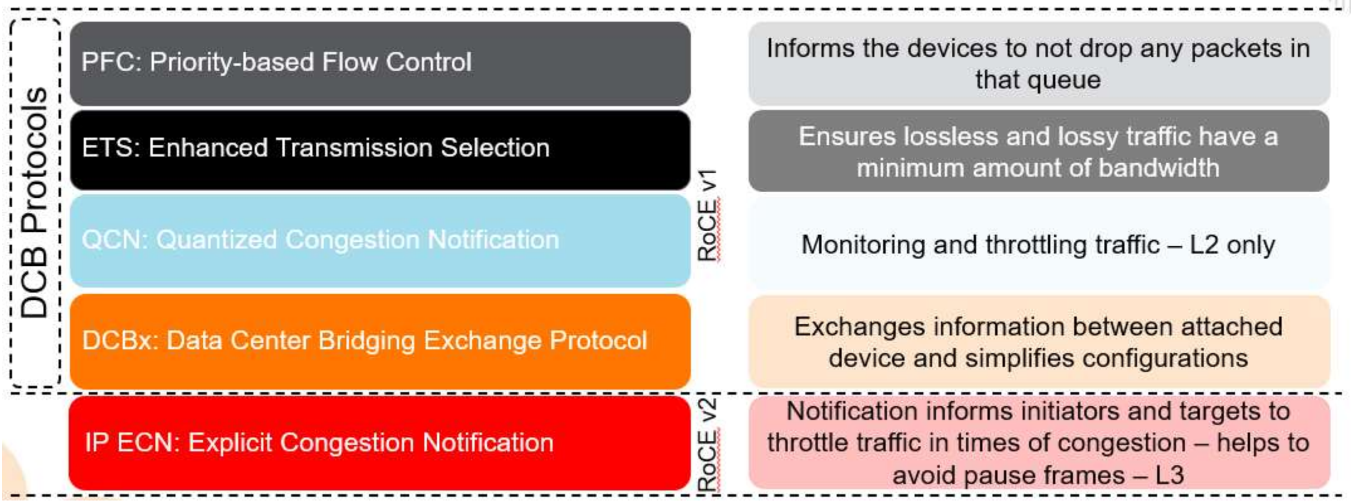

Figure 4. DCB Components

Note:
IP ECN is not specifically a DCB standard, but it must be used in RoCEv2 solutions.

Attached hosts (initiators/targets) should be configured so that the lossless traffic flows are sent to the attached switch with the proper 802.1P/DSCP values.

The RoCEv1 configuration on the switches would apply Queue-Profiles to each switch to ensure those marked packets are in the proper queue. A Queue-Schedule would be created on each switch if any of the links was carrying both lossless and lossy traffic. The Queue-Schedule will get applied to each interface that carries both types of traffic. PFC will be applied to the needed interface/queues to ensure traffic in the proper queue do not get dropped. DCBx LLDP should be turned on and Trust 802.1P should be applied to all interfaces in path carrying lossless traffic.

Finally, the DCBx Application TLV can be used to tell the attached host to send lossless traffic marked with the proper 802.1p code point. This helps ensure the switch treats the important traffic properly. If the attached host does not have the willing bit turned on then the admins will need to manually configure the Hypervisor to mark traffic properly.

RoCEv2 Configuration guidance and details

RoCEv2 solutions are very similar to the RoCEv1 solution, however, the solution will now have a L3 hop in the traffic path. The hop towards the hosts will most likely still be L2, however the hop between switches will now be L3.

Because of this, that L3 link will need to trust DSCP. The receiving switch will then be able to honor the DSCP value and place that traffic in the proper queue based on the queue-profile.

RoCE Congestion Management and ECN

RCM provides the capability to avoid congestion hot spots and optimize the throughput of the fabric even over Layer 3 links.

The RoCEv2 Congestion Management feature is composed of three points:

- The congestion point (CP): Detects congestion and marks packets using DCQN bits.
- The notification point (NP) (receiving end node): Reacts to the marked packets by sending congestion notification packets (CNPs).
- The reaction Point (RP) (transmitting end node): Reduces the transmission rate according to the received CNPs.

With RoCE RCM, when congestion occurs the CP sees congestion, but it continues to forward the traffic to the destination NP. Now that the NP destination knows there is congestion it replies to the source node RP and informs it that there is congestion. Now the source RP reacts by decreasing and later on increasing the Tx "transmit" rates according to the feedback provided. The source RP node keeps increasing the Tx rates until the system reaches a steady state of non-congested flow with traffic rates as high as possible.

On the networking side RoCE RCM is configured using IP ECN. Packets are marked with IP ECN bits by each switch at a configurable ECN threshold, allowing TCP to function normally and helping to prevent pauses. Both endpoints and each device in the traffic path needs to support this feature. ECN uses the DS field in the IP header to mark the congestion status along the packet transmission path.

ECN operates as follows:

- When the average queue size exceeds the lower limit, and is below the upper limit, before the device drops a packet which should be dropped according to the drop probability, the device examines the ECN field of the packet
    - If the ECN field shows that packet is sent out ECN-capable terminal, the device sets both the ECT bit and CE bit to 1 and forwards the packet
    - If the ECN field shows that a packet has experienced congestion (Both the ECT bit and CE bit are 1), the device forwards the packet without modifying the ECN field
    - If both ECT bit and CE bit are 0s, the device drops the packet
- When average queue size exceeds the upper limit, the device drops packet, regardless of whether the pack is sent from ECN-capable terminal

Lab Overview/Tasks - RoCEv1 Switch Configuration

Steps that should be taken when configuring RoCEv1 solutions with Aruba CX:

1. Lab Setup
2. Enable LLDP and DCBx (THIS STEP WILL NOT WORK IN EVE-NG ENVIRONMENT AT THIS TIME)
3. Configure QoS Queue-Profile (2 queue profile allows for best absorption – lossy traffic in 1 queue / lossless traffic in the other queue)
4. Configure Global Trust
5. Configure QoS Schedule Profile
6. Apply QoS Queue-Profile and QoS Schedule-Profile
7. Configure PFC on interface/queues that require lossless Ethernet (THIS STEP WILL NOT WORK IN EVE-NG ENVIRONMENT AT THIS TIME)
8. Configure Application TLV (if hosts support DCBx) (THIS STEP WILL NOT WORK IN EVE-NG ENVIRONMENT AT THIS TIME)
9. Host Facing Interfaces
10. Configure the Hosts
11. Configure Leaf-Spine Interfaces
12. Configure Routing (this example uses OSPF)
13.

Task 1 - Lab setup

For this lab refer to Figure 6 for topology and IP address details.

Step	Command
Start all the devices, including hosts and switches	
Open each switch console and log in with user “admin” and no password	
Change all hostnames as shown in the topology	`Switch(config)# hostname LeafA`

On all devices, bring up required ports	`LeafA(config)# int 1/1/1`
	`LeafA(config-if)# no shut`
	`LeafA(config-if)# int 1/1/8`
	`LeafA(config-if)# no shut`

Validate LLDP neighbors appear as expected	`LeafA (config)# show lldp neighbor`

LeafA Output

```
LeafA(config)# show lldp neighbor-info

LLDP Neighbor Information
=========================

Total Neighbor Entries          : 2
Total Neighbor Entries Deleted  : 0
Total Neighbor Entries Dropped  : 0
Total Neighbor Entries Aged-Out : 0

LOCAL-PORT  CHASSIS-ID         PORT-ID     PORT-DESC      TTL      SYS-NAME
---------------------------------------------------------------------------
1/1/1       08:00:09:fa:51:2a  1/1/1       1/1/1          120      HostA
1/1/8       08:00:09:f7:20:c5  1/1/8       1/1/8          120      Spine
```

Task 2 - Enable LLDP and DCBx

DCBx is a discovery and capability exchange protocol which exchanges configuration information between attached devices. The exchanged info can be useful in troubleshooting mismatches between endpoints. LLDP DCBx can be enabled either globally or on an interface level.

Please note that for this Task you will not be able to configure DCBx on the EVE Lab switches as it is unsupported at this time

However, with real switches you would follow the below steps (See the Aruba CX Fundamentals Guide for more details on LLDP.):

Step	Command
Enable LLDP (enabled by default) (“no” form disables)	`Switch(config)# lldp`
Verify LLDP status	`Switch(config)# show lldp configuration`
Enable DCBx Globally (disabled by default) (“no” form disables)	`Switch(config)# lldp dcbx`
Enabling DCBx on an interface. (enabled by default once enabled globally).	`Switch(config-if)# lldp dcbx`
Verify DCBx status	`Switch(config)# show dcbx <interface>`

Expected output from actual physical switch

```
Switch(config)# show dcbx int 1/1/10
   DCBX admin state: enabled
```

```
    DCBX operational state: active

    Priority Flow Control (PFC)
    ---------------------------
    Operational state : inactive

    Local advertisement:
        Willing                   : No
        MacSec ByPass Capability : No
        Max traffic classes      : 1
```

Task 3 - Configure QoS Queue-Profile

This step involves creating a queue profile which ensures that traffic gets placed into the proper queues. For example, each switch has 8 queues, but if we need to treat some traffic with PFC we should consider changing the queuing scheme (using the queue-profile) to a two queue model. In this model, we place all lossy traffic in one queue, while all lossless traffic gets placed into the other queue. This allows for better burst absorption.

Below is a simple 2-queue configuration example on Aruba CX switches. In this example, all 802.1p 4 traffic will be placed into queue 1, while all other traffic will get placed into queue 0.

Step	Command

Configure QoS Queue Profile	`LeafA(config)# qos queue-profile SMB`
	`LeafA(config-queue)# map queue 0 local-priority 0`
	`LeafA(config-queue)# map queue 0 local-priority 1`
	`LeafA(config-queue)# map queue 0 local-priority 2`
	`LeafA(config-queue)# map queue 0 local-priority 3`
	`LeafA(config-queue)# map queue 1 local-priority 4`
	`LeafA(config-queue)# map queue 0 local-priority 5`
	`LeafA(config-queue)# map queue 0 local-priority 6`
	`LeafA(config-queue)# map queue 0 local-priority 7`
	`LeafA(config-queue)# exit`

Verify QoS Queue-Profile	`LeafA(config)# show qos queue-profile SMB`

Expected output

```
  LeafA(config)# show qos queue-profile SMB
  queue_num local_priorities name
  --------- ---------------- ----
   0         0,1,2,3,5,6,7
   1         4
```

Task 4 - Configure Global Trust

We need to ensure that the proper trust configurations are applied to the proper ports. With RoCE based solutions that largely rely on the 802.1p marking we need to ensure that those marking are being trusted properly.

A best practice would be to set the global trust mode to CoS. This will be applied to all interfaces that do not already have an individual trust mode configured. A DSCP override can then be applied to any Layer 3 interfaces that do not carry the 802.1p tag.

Step	Command
Configure Global Trust	`LeafA(config)# qos trust cos`
Verify Global Trust Mode	`LeafA(config)# show qos trust`

Expected output

```
  LeafA(config)# show qos trust
   qos trust cos
```

Task 5 - Configure QoS Schedule Profile

A schedule profile must be always defined on all interfaces and each port can have its own schedule profile. The schedule profile determines the order in which queues transmit a packet and the amount of service defined for each queue.

The objective of a RoCE based Schedule Profile is to ensure that the lossless traffic has enough bandwidth based on average bandwidth consumption of the port. There is no one size fits all for these ETS settings so this parameter may end up getting modified as the admins tune the setting for the specific environment.

The switch is automatically provisioned with a schedule profile named factory-default, which assigns WFQ/DWRR to all queues with a weight of 1. The default profile named "factory-default" is applied to all interfaces as well as a predefined profile named "strict." The strict profile uses the strict priority algorithm to service all queues of the queue profile to which you apply it.

There are three permitted configurations for a schedule profile:

1. All queues use the same scheduling algorithm (i.e., WFQ).
2. All queues use strict priority.
3. The highest queue number uses strict priority, and all the remaining (lower) queues use the same algorithm (i.e., WFQ).

Only limited changes can be made to an applied schedule profile. Any other changes will result in an unusable schedule profile, and the switch will revert to the factory default profile until the profile is corrected:

1. The weight of a dwrr queue.
2. The bandwidth of a strict queue.
3. The algorithm of the highest numbered queue can be swapped between dwrr and strict, and vice versa.

The below example shows an ETS configuration within a 2-queue environment. The settings use a weight to set the amount of available bandwidth for each queue. The settings in the example below would ensure that 50% of bandwidth will be applied to both queue 0 and queue 1.

Step	Command
Create a new schedule-profile called test SMB (no form removes)	`LeafA(config)# qos schedule-profile SMB1`
Configure each queue with appropriate bandwidth/algorithm. Example shown applies 50% to each queue	`LeafA(config-schedule)# dwrr queue 0 weight 15` `LeafA(config-schedule)# dwrr queue 1 weight 15`
Verify QoS Schedule-Profile configuration	`LeafA(config-schedule)# show qos schedule-profile SMB1`

Expected output

```
LeafA(config-schedule)# show qos schedule-profile SMB1
queue_num algorithm     weight max-bandwidth_kbps burst_KB
--------- ------------- ------ ------------------ --------
0         dwrr          15
1         dwrr          15
```

Task 6 - Apply QoS Queue-Profile and QoS Schedule-Profile

Once you are satisfied with the QoS Queue-Profile and QoS Schedule-Profile configurations you should apply the configurations to each switch.

Step	Command
Apply QoS Schedule-Profile and QoS Queue-Profile	`LeafA(config)# apply qos queue-profile SMB schedule-profile SMB1`
Verify QoS Queue-Profile	`LeafA(config)# show qos queue-profile`

Expected output

```
LeafA(config)# show qos queue-profile
profile_status profile_name
-------------- ------------
applied        SMB
complete       factory-default
```

Step	Command
Verify QoS Schedule-Profile	`LeafA(config)# show qos schedule-profile`

Expected output

```
LeafA(config)# show qos schedule-profile
profile_status profile_name
-------------- ------------
applied        SMB1
complete       factory-default
complete       strict
```

Task 7 - Configure PFC on interface/queues that require lossless Ethernet

PFC enables flow control over a unified 802.3 Ethernet media interface, for local area network (LAN) and storage area network (SAN) technologies. PFC is intended to eliminate packet loss due to congestion on the network link. This allows loss sensitive protocols, such as RoCE to coexist with traditional loss-insensitive protocols over the same unified fabric.

Before the development of PFC a global pause was used which is port based. This means that in times of congestion all traffic on that port would get paused. PFC is port and queue based, which means it allows us to pause only traffic in a certain queue on a port. This way PFC simply pauses traffic that is in a lossless queue, so that no packets get dropped due to buffer congestion. All other traffic that is in the other queues on the port are not included for this type of checking and may be dropped as congestion occurs.

If you need to ensure that zero packets get dropped, then PFC must be deployed.

PFC details:

- Must be enabled on all endpoints and switches in the flow path
- Enables pause per hardware queue on an Ethernet device
- PFC is port and queue based (not flow based)
- Uses 802.1p CoS “Class of Service” values in 802.1Q VLAN tag to differentiate up to eight levels of CoS
- On L3 interfaces PFC requires preservation of 802.1Q tags
- On L3 interfaces if 802.1Q tags are not possible then traffic needs to be remarked to DSCP values
- Pause frames propagate hop-by-hop, without knowledge of the flows that are causing the congestion.
- To enable PFC at the interface level, DCBx must first be enabled.
- You can only configure 1 PFC priority per interface on 8325/8400
- You can configure 2 PFC priority per interface on 8360
- For the CX 8325 a reboot is required to enable PFC on the first interface: Subsequently, PFC can be enabled on more interfaces as long as they had never previously linked up since boot (i.e. dark ports).

When we configure PFC the idea is to configure PFC on an interface so that it knows not to drop traffic marked with a specific CoS value. By this point the queue-profile to be used should be applied, so admins should just need to enable PFC for the proper code points as needed.

The below example enables PFC for code point 4 on interface 1/1/1 and 1/1/2. In times of congestion, the switch will generate a pause frame and send it to the queue that traffic marked with an 802.1p value of 4 uses.

Please note that for this Task you will not be able to configure DCBx on the EVE Lab switches as it is unsupported at this time

Step	Command

Log into the required interfaces and apply flow-control for priority 4	`LeafA(config)# interface 1/1/1`
	`LeafA(config-if)# flow-control priority 4`
	`The setting will not be applied until configuration is saved to startup-config and the switch is rebooted.`
	`LeafA(config-if)# exit`
	`LeafA(config)# interface 1/1/8`
	`LeafA(config-if)# flow-control priority 4`
	`The setting will not be applied until configuration is saved to startup-config and the switch is rebooted.`
	`LeafA(config)# write memory`
	`LeafA(config)# boot system`

Expected output from actual physical switch

```
SpineA-RU25(config)# show dcbx int 1/1/10
  DCBX admin state: enabled
  DCBX operational state : active

  Priority Flow Control (PFC)
  ---------------------------
  Operational state : priority_mismatch

  Local advertisement:
      Willing                  : No
      MacSec ByPass Capability : No
      Max traffic classes      : 1


      Priority     Enabled
      0            False
      1            False
      2            False
      3            False
      4            True
      5            False
      6            False
      7            False
   -- MORE --, next page: Space, next line: Enter, quit: q
```

Task 8 - Configure DCBx Application TLV

The DCBx application to priority map TLV gets advertised in the DCBX application priority messages sent to attached devices. These messages tell the DCBX peer (with willing bit on) to send the application traffic with the configured priority so that the network can receive and queue traffic properly.

You can configure multiple applications in this manner. Take note if the attached device does not honor the DCBx application TLVs then the device will need to be manually configure3d to mark traffic properly.

The below example shows the DCBx Application TVL syntax, as well as some example configurations.

Please note that for this Task you will not be able to configure DCBx on the EVE Lab switches as it is unsupported at this time

Step	Command
DCBx Syntax	`switch(config)# dcbx application {ISCSI | TCP-SCTP <PORT-NUM> | TCP-SCTP-UDP <PORT-NUM> | UDP <PORT-NUM> | ether <ETHERTYPE>} priority <PRIORITY>`
Example: Mapping iSCSI traffic to priority 4	`switch(config)# dcbx application iscsi priority 4`
Example: Mapping TCP Port to priority 4	`switch(config)# dcbx application tcp-sctp 860 priority 4`

Expected output from actual physical switch

```
SpineA-RU25(config)# show dcbx int 1/1/10
   ……
   Application Priority Map
   -------------------------

   Local advertisement:
   Protocol         Port/Type        Priority
   -----------------------------------------
   iscsi                             4
```

Task 9 - Host Facing Interfaces

The final steps for finishing RoCEv1 configuration is to complete the VLAN assignments and host facing interface configurations on each Leaf switch.

Step	Command

Log into each Leaf switch and add the appropriate VLAN configurations	`LeafA(config)# vlan 10`
	`LeafA (config-vlan-10)# int vlan 10`
	`LeafA(config-if-vlan)# ip address 10.1.10.250/24`
	`LeafA(config-if-vlan)# int 1/1/1`
	`LeafA(config-if)# no routing`
	`LeafA(config-if)# vlan access 10`

Expected output

```
  LeafA(config)# show vlan
    ----------------------------------------------------------------------------------
  VLAN  Name                     Status  Reason                   Type    Interfaces
  ----------------------------------------------------------------------------------
  1     DEFAULT_VLAN_1           down    no_member_port           default
  10    VLAN10                   up      ok                       static  1/1/1

  LeafA(config)# show int vlan 10

  Interface vlan10 is up
   Admin state is up
   Description:
   Hardware: Ethernet, MAC Address: 08:00:09:a6:2a:01
   IPv4 address 10.1.10.250/24

   Statistic                          RX                   TX                Total
   ---------------- -------------------- -------------------- --------------------
   L3 Packets                          0                    0                    0
   L3 Bytes                            0                    0                    0

  LeafA(config)# sho run int 1/1/1
  interface 1/1/1
      no shutdown
      no routing
      vlan access 10
      exit
```

Note that if PFC was supported on EVE then we would also see the PFC config as a part of this output

Task 10 – Configure the Hosts

Host configurations with real physical hosts will vary based on the host OS. The hosts in this lab are VPCS, so we will simply configure an IP address to ensure pings pass through. Because these are VPCS hosts we will not be configuring any DCBx type configurations. As an example this link provides details on configuring DCB on a Linux Host https://man7.org/linux/man-pages/man8/dcb-dcbx.8.html.

Step	Command
Log into each VPCS Host and add the appropriate IP configurations	`HostA - VPCS> ip 10.1.10.1/24 10.1.10.250`

Expected output

```
VPCS> sho ip

  NAME        : VPCS[1]
  IP/MASK     : 10.1.10.1/24
  GATEWAY     : 10.1.10.250
  DNS         :
  MAC         : 00:50:79:66:68:06
  LPORT       : 20000
  RHOST:PORT  : 127.0.0.1:30000
  MTU         : 1500


VPCS> ping 10.1.10.250

  84 bytes from 10.1.10.250 icmp_seq=1 ttl=64 time=1.144 ms
  84 bytes from 10.1.10.250 icmp_seq=2 ttl=64 time=1.125 ms
  84 bytes from 10.1.10.250 icmp_seq=3 ttl=64 time=1.210 ms
  84 bytes from 10.1.10.250 icmp_seq=4 ttl=64 time=1.978 ms
  84 bytes from 10.1.10.250 icmp_seq=5 ttl=64 time=1.300 ms
```

Task 11 – Configure Leaf-Spine Interfaces

In this step, configure the Spine to Leaf interfaces and ensure each directly connected segment can reach each other.

Step	Command

Configure the required interfaces on the Spine switch with the appropriate IP addresses	`Spine(config)# int 1/1/8`
	`Spine(config-if)# no shutdown`
	`Spine(config-if)# ip address 192.168.3.0/31`
	`Spine(config)# exit`
	`Spine(config)# int 1/1/9`
	`Spine(config-if)# no shutdown`
	`Spine(config-if)# ip address 192.168.3.2/31`

Configure the required interfaces on the Leaf switches with the appropriate IP addresses	`LeafA(config)# int 1/1/8`
	`LeafA(config-if)# no shutdown`
	`LeafA(config-if)# ip address 192.168.3.1/31`
	`LeafA(config-if)# exit`
	
	`LeafB(config)# int 1/1/9`
	`LeafB(config-if)# no shutdown`
	`LeafB(config-if)# ip address 192.168.3.3/31`
	`LeafB(config-if)# exit`

Confirm that each Leaf switch can reach directly connected Spine interface.

Expected output

```
LeafA(config)# ping 192.168.3.0
```

```
  PING 192.168.3.0 (192.168.3.0) 100(128) bytes of data.
  108 bytes from 192.168.3.0: icmp_seq=1 ttl=64 time=2.62 ms
  108 bytes from 192.168.3.0: icmp_seq=2 ttl=64 time=1.85 ms
  108 bytes from 192.168.3.0: icmp_seq=3 ttl=64 time=2.06 ms
  108 bytes from 192.168.3.0: icmp_seq=4 ttl=64 time=1.87 ms
  108 bytes from 192.168.3.0: icmp_seq=5 ttl=64 time=2.18 ms

  --- 192.168.3.0 ping statistics ---
  5 packets transmitted, 5 received, 0% packet loss, time 4003ms
  rtt min/avg/max/mdev = 1.852/2.115/2.624/0.281 ms

  LeafB(config-if)# ping 192.168.3.2
  PING 192.168.3.2 (192.168.3.2) 100(128) bytes of data.
  108 bytes from 192.168.3.2: icmp_seq=1 ttl=64 time=3.57 ms
  108 bytes from 192.168.3.2: icmp_seq=2 ttl=64 time=2.30 ms
  108 bytes from 192.168.3.2: icmp_seq=3 ttl=64 time=2.90 ms
  108 bytes from 192.168.3.2: icmp_seq=4 ttl=64 time=1.92 ms
  108 bytes from 192.168.3.2: icmp_seq=5 ttl=64 time=2.11 ms

  --- 192.168.3.2 ping statistics ---
  5 packets transmitted, 5 received, 0% packet loss, time 4004ms
  rtt min/avg/max/mdev = 1.915/2.558/3.567/0.602 ms
```

Task 12 – Configure Routing

For this step, configure the OSPF routing on the Spines and Leafs to ensure there is rack-to-rack connectivity. The goal is to make sure each Host can reach each other.

Step	Command
Configure OSPF on the Spine switch	Spine(config)# router ospf 1 Spine(config-ospf-1)# router-id 1.1.1.1 Spine(config-ospf-1)# area 0.0.0.1 Spine(config-ospf-1)# exit Spine(config)# int 1/1/8-1/1/9 Spine(config-if-<1/1/8-1/1/9>)# ip ospf 1 area 0.0.0.1 Spine(config-if-<1/1/8-1/1/9>)# exit
Configure OSPF on the LeafA switch	LeafA(config)# router ospf 1 LeafA(config-ospf-1)# router-id 2.2.2.2 LeafA(config-ospf-1)# area 0.0.0.1 LeafA(config-ospf-1)# exit LeafA(config)# int 1/1/8 LeafA(config-if)# ip ospf 1 area 0.0.0.1 LeafA(config-if)# exit
Configure OSPF on the LeafB switch	LeafB(config)# router ospf 1 LeafB(config-ospf-1)# router-id 3.3.3.3 LeafB(config-ospf-1)# area 0.0.0.1 LeafB(config-ospf-1)# exit LeafB(config)# int 1/1/9 LeafB(config-if)# ip ospf 1 area 0.0.0.1 LeafB(config-if)# exit

Confirm that each Leaf switch can reach the other Leaf & confirm that each host can reach each host.
 	
<u>Expected output</u>
 	
LeafA(config)# ping 11.1.10.250
  PING 11.1.10.250 (11.1.10.250) 100(128) bytes of data.
  108 bytes from 11.1.10.250: icmp_seq=1 ttl=63 time=3.26 ms
  108 bytes from 11.1.10.250: icmp_seq=2 ttl=63 time=3.57 ms
  108 bytes from 11.1.10.250: icmp_seq=3 ttl=63 time=3.00 ms	

```
  108 bytes from 11.1.10.250: icmp_seq=4 ttl=63 time=3.45 ms
  108 bytes from 11.1.10.250: icmp_seq=5 ttl=63 time=2.87 ms

  --- 11.1.10.250 ping statistics ---
  5 packets transmitted, 5 received, 0% packet loss, time 4005ms
  rtt min/avg/max/mdev = 2.869/3.228/3.568/0.263 ms


LeafB(config-if)# ping 10.1.10.250
  PING 10.1.10.250 (10.1.10.250) 100(128) bytes of data.
  108 bytes from 10.1.10.250: icmp_seq=1 ttl=63 time=4.10 ms
  108 bytes from 10.1.10.250: icmp_seq=2 ttl=63 time=2.63 ms
  108 bytes from 10.1.10.250: icmp_seq=3 ttl=63 time=3.14 ms
  108 bytes from 10.1.10.250: icmp_seq=4 ttl=63 time=3.48 ms
  108 bytes from 10.1.10.250: icmp_seq=5 ttl=63 time=2.55 ms

  --- 10.1.10.250 ping statistics ---
  5 packets transmitted, 5 received, 0% packet loss, time 4004ms
  rtt min/avg/max/mdev = 2.554/3.180/4.098/0.570 ms


HostB - VPCS> ping 10.1.10.1
  84 bytes from 10.1.10.1 icmp_seq=1 ttl=61 time=9.866 ms
  84 bytes from 10.1.10.1 icmp_seq=2 ttl=61 time=3.352 ms
  84 bytes from 10.1.10.1 icmp_seq=3 ttl=61 time=3.680 ms
  84 bytes from 10.1.10.1 icmp_seq=4 ttl=61 time=3.434 ms
  84 bytes from 10.1.10.1 icmp_seq=5 ttl=61 time=3.015 ms
```

Task 13 – Configure RoCEv2 (ECN)

RoCEv2 solutions are very similar to the RoCEv1 solution, however, the solution will now have a L3 hop in the traffic path. The hop towards the hosts will most likely still be L2, however the hop between switches will now be L3.

Because of this, that L3 link will need to trust DSCP. The receiving switch will then be able to honor the DSCP value and place that traffic in the proper queue based on the queue-profile.

RoCE Congestion Management and ECN

RCM provides the capability to avoid congestion hot spots and optimize the throughput of the fabric even over Layer 3 links.

The RoCEv2 Congestion Management feature is composed of three points:

- The congestion point (CP): Detects congestion and marks packets using DCQN bits.
- The notification point (NP) (receiving end node): Reacts to the marked packets by sending congestion notification packets (CNPs).
- The reaction Point (RP) (transmitting end node): Reduces the transmission rate according to the received CNPs.

With RoCE RCM, when congestion occurs the CP sees congestion, but it continues to forward the traffic to the destination NP. Now that the NP destination knows there is congestion it replies to the source node RP and informs it that there is congestion. Now the source RP reacts by decreasing and later on increasing the Tx “transmit” rates according to the feedback provided. The source RP node keeps increasing the Tx rates until the system reaches a steady state of non-congested flow with traffic rates as high as possible.

On the networking side RoCE RCM is configured using IP ECN. Packets are marked with IP ECN bits by each switch at a configurable ECN threshold, allowing TCP to function normally and helping to prevent pauses. Both endpoints and each device in the traffic path needs to support this feature. ECN uses the DS field in the IP header to mark the congestion status along the packet transmission path.

ECN operates as follows:

- When the average queue size exceeds the lower limit, and is below the upper limit, before the device drops a packet which should be dropped according to the drop probability, the device examines the ECN field of the packet
    - If the ECN field shows that packet is sent out ECN-capable terminal, the device sets both the ECT bit and CE bit to 1 and forwards the packet
    - If the ECN field shows that a packet has experienced congestion (Both the ECT bit and CE bit are 1), the device forwards the packet without modifying the ECN field
    - If both ECT bit and CE bit are 0s, the device drops the packet
- When average queue size exceeds the upper limit, the device drops packet, regardless of whether the pack is sent from ECN-capable terminal

---

Note: For ECN to work, all switches in path between two ECN-enabled endpoints must have ECN enabled
Note applying the ECN threshold will vary based on if the device is an 8360 or 8325/8400

Step	Command

Create a threshold profile with ECN action on queue.	`Spine(config)# qos threshold-profile ECN`
	
	`Spine(config-threshold)# queue 1 action ecn all threshold 50 percent (8360 and EVE-NG)`
	
	`Spine(config-threshold)# queue 1 action ecn all threshold 40 kbytes (8325/8400)`

(Option 1) Apply profile globally (all ports)
 	
(Option 2) Apply profile to specific Ethernet or LAG interfaces
	`Spine(config)# apply qos threshold-profile ECN`
	
	`Spine(config)# int 1/1/8-1/1/9`
	`Spine(config-if-<1/1/8-1/1/9>)# apply qos threshold-profile ECN`
	
	`Spine(config)# int lag 10`
	`Spine(config-lag-if)# apply qos threshold-profile ECN`

```
Verify threshold is applied (example used Option 2 and applied ECN only to the 2 in-use
interfaces)

Expected output
Spine(config)# show qos threshold-profile ECN
  Queue Action  Threshold
  ----- -------- ---------
  1     ecn             50

  Port       Status
  ---------- -------
  1/1/8      applied
  1/1/9      applied
```

Step	Command
Add Trust DSCP configuration to Spine	`Spine(config)# int 1/1/8-1/1/9` `Spine(config-if-<1/1/8-1/1/9>)# qos trust dscp`
Add Trust DSCP configuration to each Leaf	`LeafA(config)# int 1/1/8` `LeafA(config-if)# qos trust dscp`

Stay tuned for updates to this lab – as when features that we cannot configure now in the EVE-NG environment get added, this document will get updated.

Final Switch Configurations

<table>
<tr><th></th><th>Configurations</th></tr>
<tr><td>LeafA Final Configuration.</td><td><pre>
LeafA(config)# sho run
Current configuration:
!
!Version ArubaOS-CX Virtual.10.08.0001BG
!export-password: default
hostname LeafA
led locator on
!
!
!
!
!
!
ssh server vrf mgmt
vlan 1,10
interface mgmt
    no shutdown
    ip dhcp
qos queue-profile SMB
    map queue 0 local-priority 0,1,2,3,5,6,7
    map queue 1 local-priority 4
qos schedule-profile SMB1
    dwrr queue 0 weight 15
    dwrr queue 1 weight 15
qos threshold-profile ECN
    queue 1 action ecn all threshold 50 percent
apply qos queue-profile SMB schedule-profile SMB1
qos trust cos
interface 1/1/1
    no shutdown
    qos trust cos
    no routing
    vlan access 10
interface 1/1/8
    no shutdown
    qos trust dscp
    apply qos threshold-profile ECN
    ip address 192.168.3.1/31
    ip ospf 1 area 0.0.0.1
interface vlan 10
    ip address 10.1.10.250/24
    ip ospf 1 area 0.0.0.1
!
!
!
!
!
router ospf 1
    router-id 2.2.2.2
    area 0.0.0.1
https-server vrf mgmt
</pre></td></tr>
<tr><td>LeafB Final Configuration.</td><td><pre>
LeafB(config)# sho ru

Current configuration:
</pre></td></tr>
</table>

```
!
!Version ArubaOS-CX Virtual.10.08.0001BG
!export-password: default
hostname LeafB
led locator on
ntp server pool.ntp.org minpoll 4 maxpoll 4 iburst
ntp enable
!
!
!
!
!
!
ssh server vrf mgmt
vlan 1,11
interface mgmt
    no shutdown
    ip dhcp
qos queue-profile SMB
    map queue 0 local-priority 0,1,2,3,5,6,7
    map queue 1 local-priority 4
qos schedule-profile SMB1
    dwrr queue 0 weight 15
    dwrr queue 1 weight 15
qos threshold-profile ECN
    queue 1 action ecn all threshold 50 percent
apply qos queue-profile SMB schedule-profile SMB1
qos trust cos
interface 1/1/1
    no shutdown
    qos trust cos
    no routing
    vlan access 11
interface 1/1/9
    no shutdown
    qos trust dscp
    apply qos threshold-profile ECN
    ip address 192.168.3.3/31
    ip ospf 1 area 0.0.0.1
interface vlan 11
    ip address 11.1.10.250/24
    ip ospf 1 area 0.0.0.1
!
!
!
!
!
router ospf 1
    router-id 3.3.3.3
    area 0.0.0.1
https-server vrf mgmt
```

Spine

```
Spine# sho run
Current configuration:
!
!Version ArubaOS-CX Virtual.10.08.0001BG
!export-password: default
hostname Spine
led locator on
ntp server pool.ntp.org minpoll 4 maxpoll 4 iburst
ntp enable
!
!
!
!
!
```

```
!
ssh server vrf mgmt
vlan 1
interface mgmt
    no shutdown
    ip dhcp
qos queue-profile SMB
    map queue 0 local-priority 0,1,2,3,5,6,7
    map queue 1 local-priority 4
qos schedule-profile SMB1
    dwrr queue 0 weight 15
    dwrr queue 1 weight 15
qos threshold-profile ECN
    queue 1 action ecn all threshold 50 percent
apply qos queue-profile SMB schedule-profile SMB1
qos trust cos
interface 1/1/8
    no shutdown
    qos trust dscp
    apply qos threshold-profile ECN
    ip address 192.168.3.0/31
    ip ospf 1 area 0.0.0.1
interface 1/1/9
    no shutdown
    qos trust dscp
    apply qos threshold-profile ECN
    ip address 192.168.3.2/31
    ip ospf 1 area 0.0.0.1
!
!
!
!
!
router ospf 1
    router-id 1.1.1.1
    area 0.0.0.1
https-server vrf mgmt
```

a Hewlett Packard Enterprise company

www.arubanetworks.com

3333 Scott Blvd. Santa Clara, CA 95054
1.844.472.2782 | T: 1.408.227.4500 | FAX: 1.408.227.4550 | info@arubanetworks.com