UG1534 U25N SmartNIC User Guide

version 1.8

AMD, Inc.

October 17, 2023

Contents

U25N **1**

Introduction 1

2 Supported Services 1

2.1 U25N Modes: Legacy and Switchdev Mode 1

2.2 Open vSwitch 1

2.2.1 Port to Port 2

2.2.2 Port to VM or VM to Port 2

2.2.3 VM to VM 2

2.2.5 Supported Overlay Network protocols 2

2.2.5.1 L2GRE 2

2.2.5.1 VXLAN 2

2.2.6 LACP 2

2.2.7 VLAN Push/Pop 2

2.2.8 Conntrack 2

2.3 IPsec 3

2.4 Stateless Firewall 3

2.5 DPDK on U25N 3

2.6 Statictics counter 3

2.7 Debug 3

3. U25N Installation 3

3.1 Basic Requirements and Component Versions Supported 3

3.2 U25N Driver 4

3.2.1 Solarflare Linux Utilities Installation 4

3.2.2 U25N Driver Installation 4

3.2.3 Updating U25N Firmware 5

3.2.4 sfboot Configuration 6

3.3 U25N Shell 6

3.3.1 Shell Version Check 6

3.3.2 Testing Golden Image Functionality 7

3.3.2.1 Checking Basic NIC Functionality 7

3.3.3 Deployment Image Flashing 7

3.3.3.1 Loading Partial Bitstream into Deployment Shell 8

3.4 Steps to change MPSoC Linux kernel image (For advanced users) 9

3.5 Reverting the U25N SmartNIC to Golden Image 10

4 Detailed Applications Description 10

4.1 Legacy and Switchdev Modes 10

4.1.1 Legacy NIC (Default) 10

4.1.2 Switchdev Mode 11

4.2 OVS 11

4.2.1 Installing OVS 11

4.2.2 Classification Fields (Matches) 11

4.2.3 Number of Flow Supported 12

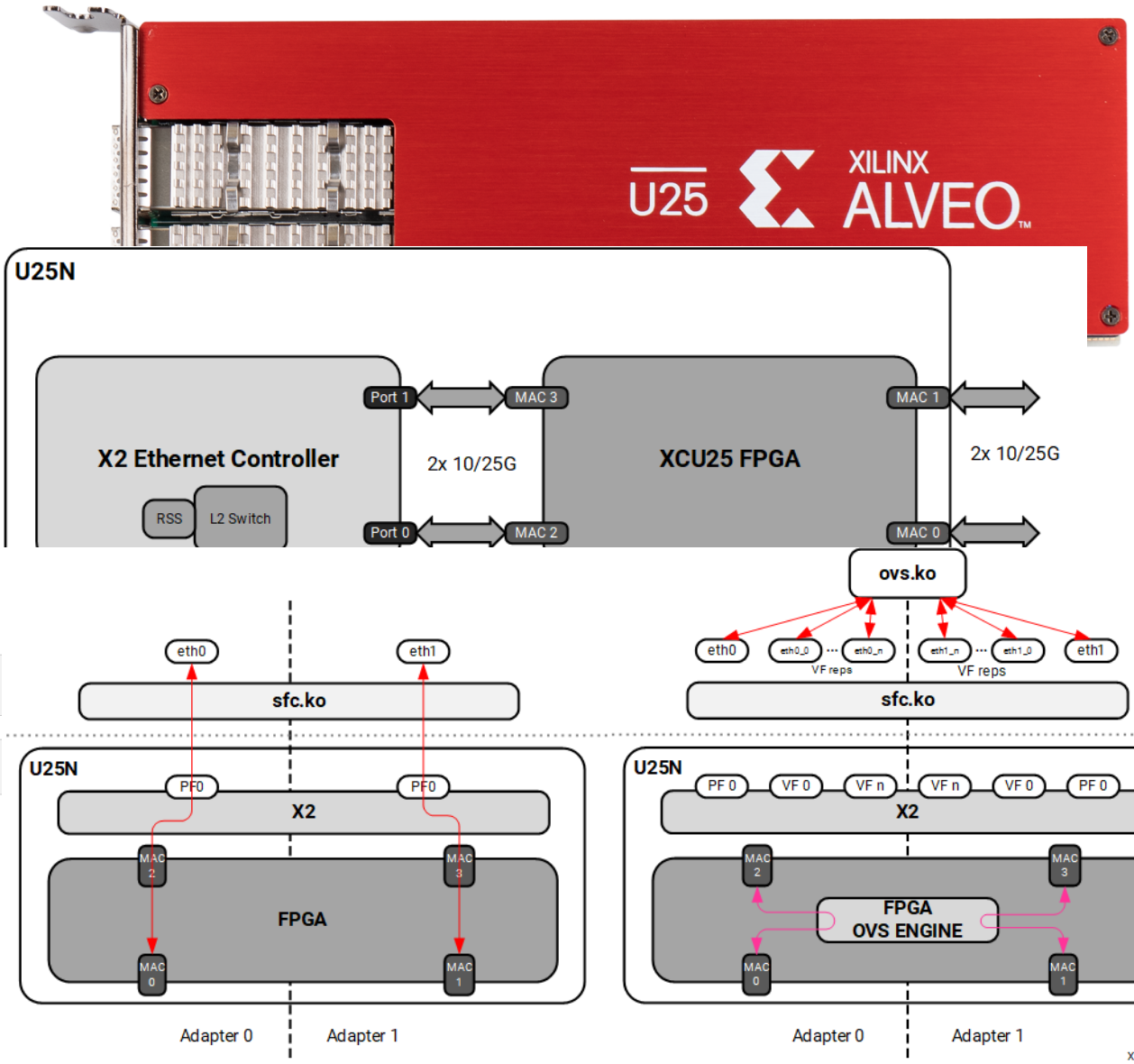
4.2.4 Port to Port 12

4.2.5 Port to VM or VM to Port 13

4.2.6 VM to VM 15
4.2.7 Tunnels (Encapsulation/Decapsulation) 17
4.2.7.1 L2GRE 18
L2GRE Server 1 Configuration 18
L2GRE Server 2 Configuration 19
4.2.7.2 VXLAN 19
VXLAN Server 1 Configuration 20
VXLAN Server 2 Configuration 21
4.2.7.3 VM to VM or VM to Port or Port to VM Tunnel 21
Tunnel Server 1 Configuration 21
Tunnel Server 2 Configuration 24
4.2.8 LACP 24
4.2.8.1 LAG Creation 24
4.2.8.2 LAG Deletion 27
4.2.9 Connection Tracking 28
4.2.10 OVS Configuration 28
4.2.11 Functionality Check 29
Ping Test 29
4.2.12 Uninstalling OVS Setup 30
4.3 IPsec 30
4.3.1 Supported XFRM Parameters 30
4.3.2 Classification Fields (Matches) 31
4.3.2.1 Encryption 31
4.3.2.2 Decryption 31
4.3.3 strongSwan Installation 31
4.3.3.1 IPSec Server 1 Configuration 31
4.3.3.2 IPSec Server 2 Configuration 32
4.3.3.3 Server 1: Steps to Run IPsec 33
4.3.3.4 Server 2: Steps to Run IPsec 33
4.4 Stateless Firewall 34
Kernel Upgrade 34
nftables 34
4.4.3 Classification Fields (Matches) 34
Driver Installation 35
4.5 Rate Limiting 36
4.5.1 Ingress rule: 36
4.5.1.1 Add Rate Limit Policy: 36
4.5.1.2 Delete Rate Limit Policy 36
4.5.2 Egress rule: 36
4.5.2.1 Add Rate Limit Policy: 36
4.5.2.2 Delete Rate Limit Policy 37
4.6 Mirroring : 37
4.6.1 Local Mirroring 37
4.6.1.1 Add Local Ingress Mirroring Rule: 37
4.6.1.2 Delete Local Ingress Mirroring Rule: 37
4.6.1.3 Add Local Egress Mirroring Rule: 37

4.6.1.4 Delete Local Egress Mirroring Rule: 37
4.6.2 Remote Mirroring 38
4.6.2.1 Setup VxLAN tunnel between local and remote servers 38
4.6.2.2 Add Remote Ingress Mirroring Rule: 38
4.6.2.3 Delete Remote Ingress Mirroring Rule: 38
4.6.2.4 Add Remote Egress Mirroring Rule: 38
4.6.2.5 Delete Remote Egress Mirroring Rule: 38
4.7 Statistics 38
4.7.1 OVS Commands 38
4.7.2 MAE Rules 39
4.7.3 IPsec Statistics 39
4.8 Debug Commands 39
DPDK 42
APPENDIX A U25N Shell Programming 43
Entering into U-Boot Mode 43
APPENDIX B VM Installation 45
APPENDIX C OpenStack Installation 46
1. Hardware Requirements and U25N Installation 46
2. Software Requirements 47
2.1 OpenStack Releases 47
2.2 OS versions 47
2.3 OpenStack Deployment Tool 47
3. OpenStack Installation 47
3.1 Grub Configuration 47
3.2 Add Stack User (optional) 47
3.3 Download DevStack 48
3.4 Update DevStack Configuration 48
3.5 U25N Open vSwitch Hardware Offload 49
3.6 Launch VM instance 52
4 OpenStack Configuration Sample 53

U25N

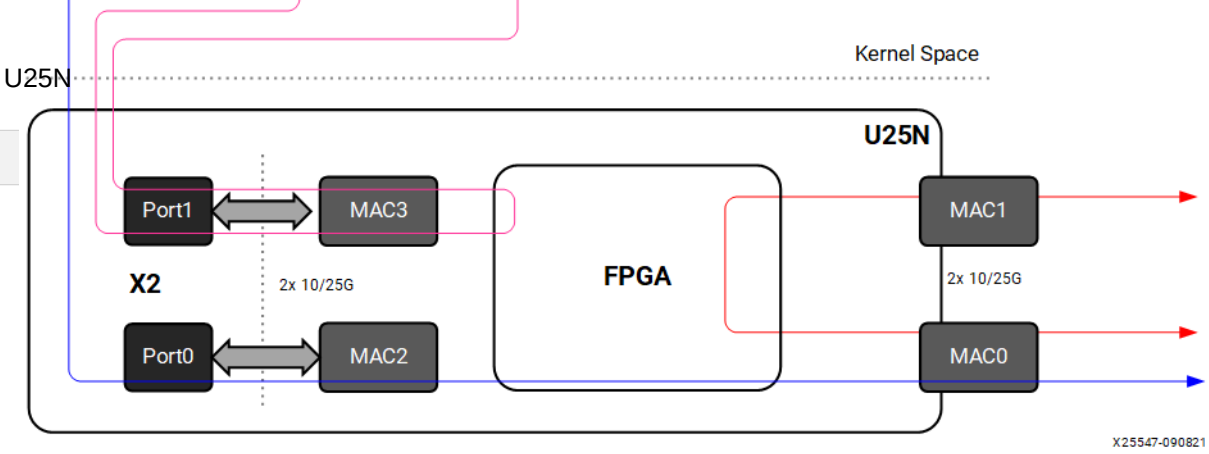


2.2 Open vSwitch

Open vSwitch (OVS) is a software-defined virtual switch that a hypervisor uses to manage traffic between virtual instances. It performs not only classic layer-2 switching, but also supports arbitrarily complex rules matching on any packet field, and performing any transformation.

OVS groups packet flows requiring the same switching behavior into megaflows that can require filtering at layer 2, layer 3, or layer 4 (potentially changing over time) of the Network stack. Deployments can require very large numbers of simultaneous megaflows to be supported. The downside of OVS is that it is resource-intensive. Cloud providers have to dedicate a significant number of cores to run this infrastructure, instead of renting those cores to their customers for running business applications.The U25N SmartNIC can offload the virtual switching off of the host processor and hence improve CPU efficiency .

OVS supports various dataflows: port to VM, VM to VM, and port to port. The following section contains detailed information about these dataflows.



artNIC and vice
Initial incoming
emon runs) for
SmartNIC which

2.2.2 Port to VM or VM to Port

In this configuration, traffic flows from the host or VMs to U25N physical ports, and vice versa. This configuration is achieved by creating virtual functions (VFs) corresponding to a physical function (PFs) using sfboot command (for more information refer to the Solarflare Server Adapter user guide at Solarflare Server Adapter User Guide). In this configuration packet switching is done by OVS.

2.2.3 VM to VM

In this configuration, traffic flows from a VM to another VM via OvS kernel module present in the SmartNIC. Packet switching is done by OvS in this configuration.

2.2.5 Supported Overlay Network protocols

2.2.5.1 L2GRE

OVS supports layer 2 over generic routing encapsulation (L2GRE). L2GRE tunnel bridges two discrete LAN segments over the internet by creating a tunnel. At the origin of the tunnel, packets are encapsulated and at the terminating end, packets are decapsulated. The constant encapsulation and decapsulation puts a tremendous load on the host CPU.

With the help of hardware offload, packet encapsulation and decapsulation is offloaded to the U25N SmartNIC and does not consume valuable host CPU cycles.

2.2.5.1 VXLAN

OVS supports Virtual eXtensible Local Area Network or VXLAN. VXLAN is an overlay network that transports an L2 network over an existing L3 network. For more information on VXLAN, please see RFC 7348. Like L2GRE the VXLAN tunnel creation and packet encapsulation/decapsulation process can be offloaded to the U25N Smart NIC, thus saving valuable CPU cycles.

2.2.6 LACP

Link aggregation is the combining of two 25Gbps links in parallel, in order to increase throughput beyond a single 25Gbps connection, to provide redundancy in case one of the links should fail, or both. PF bond and VF_Rep bond is added to the OvS bridge.

2.2.7 VLAN Push/Pop

OVS supports offload of vlan header push/pop actions.

- strip_vlan: Strips the VLAN tag from a packet.
- push_vlan: Push a new VLAN tag onto the packet.

2.2.8 Conntrack

Conntrack is a connection tracking module for stateful packet inspection. OVS can use the connection tracking system to match on the TCP segments from connection setup to connection tear down where OpenFlow flow can be used to match on the state of the packet.

2.3 IPsec

Internet protocol security (IPsec) is a secure network protocol suite that authenticates and encrypts packets to provide a secure channel between two endpoints over an internet protocol network. Encrypted packets are forwarded to the CPU and handled by an open-source application called strongSwan (see https://www.strongswan.org). Due to the computing complexity of encryption and decryption of packets, handling IPsec in the host CPU consumes significant compute cycles. Offloading this operation to U25N hardware easily increases system-level throughput and lowers CPU utilization.

Currently, IPsec transport mode is supported in the gateway mode, i.e., traffic from one port is encrypted and sent to the other port.

2.4 Stateless Firewall

Stateless firewalls make use of a packet's source, destination, and other parameters to figure out whether it presents a threat. Stateless firewall should be added on the ingress of ports so that filtering based on nftables rules can be applied. nftables is a subsystem of the Linux kernel providing filtering and classification of network packets/datagrams/frames. First-level filtering is done based upon nftables rules. The hardware offload is available for nftables through the netdev family and the ingress hook. This also includes base chain hardware offloading.

2.5 DPDK on U25N

Data Plane Development Kit or DPDK is a kernel bypass technique to accelerate packet processing. DPDK enables the applications running in user-space to interact directly with NIC, bypassing the Linux kernel space. DPDK uses Poll Mode Driver (PMD) to interact with the underlying NIC. DPDK can be run on the X2 PF or VF. To run the testpmd/pktgen application on the U25N, refer to Solarflare libefx-based Poll Mode Driver DPDK. For more information about how to run DPDK on the U25N, refer to DPDK.

2.6 Statictics counter

Statistics data will be updated periodically from U25N hardware. Counts would be available for each service separately. Commands are provided to display the counter values.

2.7 Debug

A set of debug command is provided which collects

- debug logs from PS and send them to host CPU.
- dump the default rules and offloaded rules of OVS.

3. U25N Installation

3.1 Basic Requirements and Component Versions Supported

- OS requirement:
  - Ubuntu 18.04, 20.04 or 22.04.

```
***Note*:** LTS version of Ubuntu is recommended.
```

  - Red Hat Enterprise Linux 8.3 or 8.4 (tested legacy mode only)
- Kernel requirements:
  - OVS Functionality ≥ 4.15.
  - Stateless Firewall Functionality ≥ 5.5.
  - Conntrack Functionality is tested on Ubuntu 20.04.4 LTS with kernel 5.15.0-46-generic.

- PCIe® Gen3 x16 slot.
- Requires passive airflow. More details can be found in *Alveo U25N SmartNIC Data Sheet* (DS1005).
- U25N driver version: 5.3.3.1008.3 (minimum).
- X2 firmware version: v7.8.17.1011 (minimum).

  ***Note*:** To install the latest firmware, refer to the Updating U25N Firmware section.
- OVS version: 2.12 or above. For more information about OVS and its installation, refer to https://docs.openvswitch.org/en/latest/intro/install/general.

  ***Note*:** To support Connection Tracking feature, OVS version 2.13.5 or above is required.

3.2 U25N Driver

***Note*:** Root user privileges are required to execute following steps and commands.

Once the server is successfully booted with U25N Smart NIC, validate that the U25N Smart NIC is detected using the lspci command:

```
lspci | grep Solarflare
```

It will list two PCIe ID's corresponding to U25N, similar to the following sample output:

```
af:00.0 Ethernet controller: Solarflare Communications XtremeScale SFC9250
10/25/40/50/100G Ethernet Controller (rev 01)
af:00.1 Ethernet controller: Solarflare Communications XtremeScale SFC9250
10/25/40/50/100G Ethernet Controller (rev 01)
```

3.2.1 Solarflare Linux Utilities Installation

The Solarflare Linux Utilities package can be found from the U25N software package downloaded from the lounge.

If you are using an Ubuntu server:

- Download and unzip the package on the target server.

  ***Note*:** Alien package must be downloaded using the command

  ```
  sudo apt install alien
  ```
- Create the `.deb` file:

  ```
  sudo alien sfutils■<version>.x86_64.rpm
  ```

  This command generates the `sfutils_<version>_amd64.deb` file.
- Install the `.deb` file:

  ```
  sudo dpkg -i sfutils_<version>_amd64.deb
  ```

If you are using Red Hat Enterprise Linux server:

- Install the binary RPM:

  ```
  sudo rpm -Uvh sfutils-<version>.x86_64.rpm
  ```

The server should have sfupdate, sfkey, sfctool and sfboot utilities installed now.

3.2.2 U25N Driver Installation

***Note*:** Please stop Open vSwitch if it is running before U25N driver installation

- If you are using an Ubuntu server
  - Install the dkms package with the following command if not available:

    ```
    sudo apt-get install dkms
    ```

***Note*:** If the driver version is latest, please ignore this U25N driver installation step.

- Check the driver version of U25N interface using the following command:

```
ethtool -i u25eth0 | grep version #u25eth0 is the first port of U25N.
```

- If legacy version debian package already exists, before installing the latest debian package, remove the already existing package.

```
modprobe mtd
modprobe mdio
rmmod sfc
dpkg -r sfc-dkms
```

- Then install debian driver package

```
dpkg -i sfc-dkms_<version>_all.deb
modprobe sfc
```

- If you are using Red Hat Enterprise Linux server
  - Install dependency “unifdef-2.12” package if it is not installed. Please refer to https://dotat.at/prog/unifdef/.
  - Unload existing sfc driver if legacy driver is already loaded

```
rmmod sfc
```

  - Remove sfc rpm package (if it’s installed already)

```
rpm -qa | grep sfc #to get sfc rpm package name
rpm -e <kernel-module-sfc-RHEL8-4-<version>.x86_64.rpm>
```

  - Install the rpm package

```
rpm -i RHEL8.3-<version>.x86_64.rpm (for 8.3 Kernel)
rpm -i RHEL8.4-<version>.x86_64.rpm (for 8.4 Kernel)
modprobe sfc
```

***Note*:** Once the package is installed, after each power cycling or reboot, rmmod sfc and modprobe sfc is needed.

3.2.3 Updating U25N Firmware

- Step 1: Make sure U25N driver is loaded using `lsmod` command

```
lsmod | grep sfc
```

Eg :

```
lsmod | grep sfc
sfc                   626688  0
```

- Step 2: Execute the following step to update firmware

```
sfboot --list # This should output U25N interfaces
sfupdate #This should work or do the below step
sfupdate -i <PF_interface> --write --force --yes # u25eth0 is the PF0 interface of U25N
```

***Note*:** Please use the PF0 interface. No reboot is required.

- Step 3: Confirm the firmware version using the command `sfupdate`. Version should be 1011.

For example:

```
sfupdate | grep Bundle
Bundle type:        U25N
Bundle version:     <bundle version>
```

3.2.4 sfboot Configuration

***Note*:** Ignore the following steps if the U25N is operating in SR-IOV mode and the required VF are already assigned. Available VFs and SR-IOV mode can be verified by running the sfboot command as a root user.

For more information on enabling VFs and SR-IOV, refer to the Solarflare Server Adapter User Guide

X2 switch mode must be configured in SR-IOV mode for enabling VFs. Each U25N PF can have up to 120 VFs. The following command demonstrates how to switch to the SR-IOV mode and enable VFs.

```
sudo sfboot switch-mode=sriov vf-count=<No. of VF required>
```

***Note*:** Perform a power cycle to update the configuration. Updated configuration will be retained post cold boot.

3.3 U25N Shell

The U25N shell is programmed on the U25N with a known good image which is called “golden image” out of the factory. The Following sections demonstrate how to verify the shell. If the validation fails, refer to U25N Shell Programming.

3.3.1 Shell Version Check

- Step 1 : Check the driver version of U25N interface using the following command:

  ***Note*:** Ignore this step, if the U25N driver version is latest.

  ```
  ethtool -i u25eth0 | grep version
  ```

  ***Note*:** To install the latest sfc driver, refer to U25N Driver.

- Step 2 : Use the u25n_update utility (found inside RELEASE_DIR/utils directory) to find out the U25N shell version. Execute the following command to read the U25N shell version.

  ```
  ./u25n_update get-version <PF interface>
  ```

  For example:

  ```
  ./u25n_update get-version u25eth0

  Sample output:
  [u25n_update] - Image Upgrade/Erase Utility V3.1

  Reading version from the hardware

  X2 Firmware Version : 7.8.17.1011
  PS Firmware Version : 5.00
  Deployment shell Version : 0x0812210D # Timestamp of Shell
  Bitstream Version : 0xA00E60D3        # Timestamp of offload
  Features supported:  MAE IPSEC        # Represent NIC by default
  Timestamp : 08-12-2021 16:30:45


  STATUS: SUCCESSFUL                    # Successful U25N version read
  ```

  ***Note*:** If the u25n_update command returns status as failed, this implies that the shell is not programmed with the Golden image. Please contact support. Refer to U25N Shell Programming to flash the Golden image to the U25N shell.

  The following section demonstrates how to validate the Golden Image Functionality. For flashing deployment image to the U25N shell refer to Deployment Image Flashing.

  ***Note*:** If the Golden image version is same as the Deployment image version, flashing is not required.

3.3.2 Testing Golden Image Functionality

- The U25N shell with Golden image includes Image Upgrade and Basic NIC functions.
- U25N hardware in legacy mode where the acceleration logic is simple passthrough from ethernet ports to host driver.

3.3.2.1 Checking Basic NIC Functionality

Connect the U25N physical port to a traffic generator or any peer NIC device.

- Run ping after assigning IP addresses on source and remote ports.
- Run iperf server and client on any ports to verify traffic.
- Run DPDK testpmd at the X2 PF interface. Packets sent to the physical port will be received at the testpmd application corresponding to the bound PF.

***Note*:** Refer to DPDK on U25N to run dpdk-testpmd.

3.3.3 Deployment Image Flashing

***Note*:** U25N must have the Golden Image before performing any of the following steps. Refer to Shell Version Check to check the Shell version. Ignore the flashing if Golden image version is same as deployment version in the release.

***Note*:** Before flashing the image ensure the U25N card is in legacy mode with no OVS software application running. Also, remove all associated OVS databases(if any)

To check the mode of operation:(Legacy/Switchdev modes)

```
devlink dev eswitch show pci/0000:<pci_id> # pci_id is the BDF of U25N Device
```

- Step 1 : In case U25N driver version is latest, this step can be ignored. The following command can be used to check the driver version:

  ```
  ethtool -i u25eth0 | grep version
  ```

  ***Note*:** To install the latest sfc driver, please refer to U25N Driver.

- Step 2 : Ensure that the U25N card is in legacy mode. Below is the command to validate:

  ```
  devlink dev eswitch show pci/0000:<pci_id> # pci_id is the BDF of U25N Device
  ```

  The command mentioned above should return the following sample output:

  ```
  pci/0000:<pci_id>: mode legacy
  ```

  Use the following command to switch mode in case the card is not already in the legacy mode:

  ```
  devlink dev eswitch set pci/0000:<pci_id> mode legacy
  ```

- Step 3 : The u25n_update utility uses U25N PF0 network interface to flash the image. Execute the following command to commence image flash.

  ***Note*:** The interface must be the PF0 interface of the target U25N card. The file must be a `.bin` file. Ensure the image path to be flashed is correct.

  ```
  # The bin/ub file is in the `shell` directory of the release.
  ./u25n_update upgrade <path_to_bin_file> <PF0_interface>
  ```

  For example:

  ```
  # From Release_dir/utils:
  ./u25n_update upgrade ../shell/<boot file name>.bin u25eth0
  ```

  Post successful image flash you will the following prompt:

  ```
  STATUS: SUCCESSFUL
  ```

Reset the U25N hardware to boot from the updated deployment image.

```
_update reset <PF0_interface>
```

ple:

```
_update reset u25eth0
```

ccessful reset the following prompt will be shown:

```
: SUCCESSFUL # The above command takes a few seconds to complete to ensure proper reset sequence is
```

- Step 5 : To retrieve U25N Deployment image version details, use the following commands:

  ```
  ./u25n_update get-version <PF0_interface>
  ```

  For example:

  ```
  ./u25n_update get-version u25eth0
  [u25n_update] - Image Upgrade/Erase Utility V3.1

  Reading version from the hardware

  X2 Firmware Version : 7.8.17.1011
  PS Firmware Version : 5.00
  Deployment shell Version : 0x0812210D
  Bitstream Version : 0xA00E60D3
  Features supported:  No Features supported
  Timestamp : 08-12-2021 16:30:45


  STATUS: SUCCESSFUL
  ```

3.3.3.1 Loading Partial Bitstream into Deployment Shell

***Note*:** The U25N must be flashed with the Deployment image before running the following commands. Refer to the Deployment Image Flashing to flash the deployment image to the U25N shell.

***Note*:** In case partial Bitstream already exists in the U25N card and/or if a warm reboot has already been done, ensure a fpga reset is performed using u25n_update application before following the steps mentioned below. Use the following command to reset the FPGA.

```
./u25n_update reset <PF0_interface>
```

For example:

```
./u25n_update reset u25eth0
```

Use the PF0 interface of U25N card to initiate the FPGA reset.

- Step 1. In case U25N driver version is latest, this step can be ignored. Driver version of the U25N can be validated using the following command:

  ```
  ethtool -i <PF0_interface> | grep version
  ```

  ***Note*:** To install the latest sfc driver, refer to U25N Driver.

  ***Note*:** Before flashing the image, make sure the U25N Smart NIC is in legacy mode, with OVS software application running, and all OVS databases are removed.

- Step 2. The U25N Smart NIC should be in legacy mode. Please refer to step 2 of Deployment Image Flashing for how to verify and change the NIC mode.
- Step 3. Image flashing happens through the U25N PF0 network interface using the u25n_update utility.
  - The image is flashed using the following CLI command:

***Note*:** The interface used in the above command should be the PF0 interface of the target U25N SmartNIC. The file must be a `.bit` file. Ensure the path to the image to be flashed is correct.

```
./u25n_update upgrade <path_to_bit_file> <PF0_interface>
# the path to bit files is bits directory of release package.
```

For example:

```
./u25n_update upgrade ../bits/ovs_<version>.bit u25eth0
```

- After the image is flashed successfully, the following message is displayed:

```
STATUS: Successful
```

- The u25n_update utility can be used to retrieve the bitstream version. Version retrieval happens at the U25N PF0 network interface using the u25n_update utility. Use the following command to retrieve the image version:

```
./u25n_update get-version <PF0_interface>
```

For example:

```
./u25n_update get-version u25eth0
```

Post a successful image retrieval, the following message is displayed:

```
Bitstream version: 0xA00D10D1
STATUS: SUCCESSFUL
```

3.4 Steps to change MPSoC Linux kernel image (For advanced users)

Follow this Section only when the file system needs to be updated for the existing Deployment image.

Deployment image must be flashed to the U25N shell before following the below mentioned steps. Refer to Deployment Image Flashing to flash the Deployment image.

***Note*:** Before flashing the image to the shell ensure the card is in legacy mode with no OVS software application running and all OVS databases have been removed.

- Step 1 : If the U25N driver version is 5.3.3.1008.3, this step can be ignored. Driver version of the U25N can be validated by using the following command:

```
ethtool -i <PF0_interface> | grep version
```

***Note*:** To install the latest sfc driver, refer U25N Driver section.

- Step 2 : U25N Cards should be in legacy mode. Please refer to step 2 of Deployment Image Flashing for how to verify and change the NIC mode.
- Step 3 : Image flashing is performed through the U25N PF0 network interface using the u25n_update utility.

  Image is flashed using a CLI command

  ***Note*:** The interface should be the PF0 interface of the target U25N card. The file must be a `.ub` file. Make sure the path to the image to be flashed is given correctly.

```
./u25n_update upgrade <path_to_bin_or_ub_file> <PF0_interface>
```

  For example:

```
./u25n_update upgrade image.ub u25eth0
```

  Post successful image flash, the following message will be displayed:

```
STATUS: SUCCESSFUL
```

- Step 4 : Reset the U25N hardware to boot with the new kernel image using the following command:

```
./u25n_update reset <PF0_interface>
```

For example:

```
./u25n_update reset u25eth0
```

After successful image flash, the following message will be displayed:

```
STATUS: SUCCESSFUL
```

3.5 Reverting the U25N SmartNIC to Golden Image

Reverting to the factory (or golden) image is recommended in the following cases:

- Preparing to flash a different shell onto the SmartNIC.
- The SmartNIC no longer appears on lspci after programming a custom image onto the SmartNIC.
- To recover from the state when the deployment image has become unresponsive even after performing multiple resets using the u25n_update utility.

***Note*:** After the factory reset is performed, the U25N loaded with the golden image (A known good image). It has the essential provision to upgrade the deployment image.

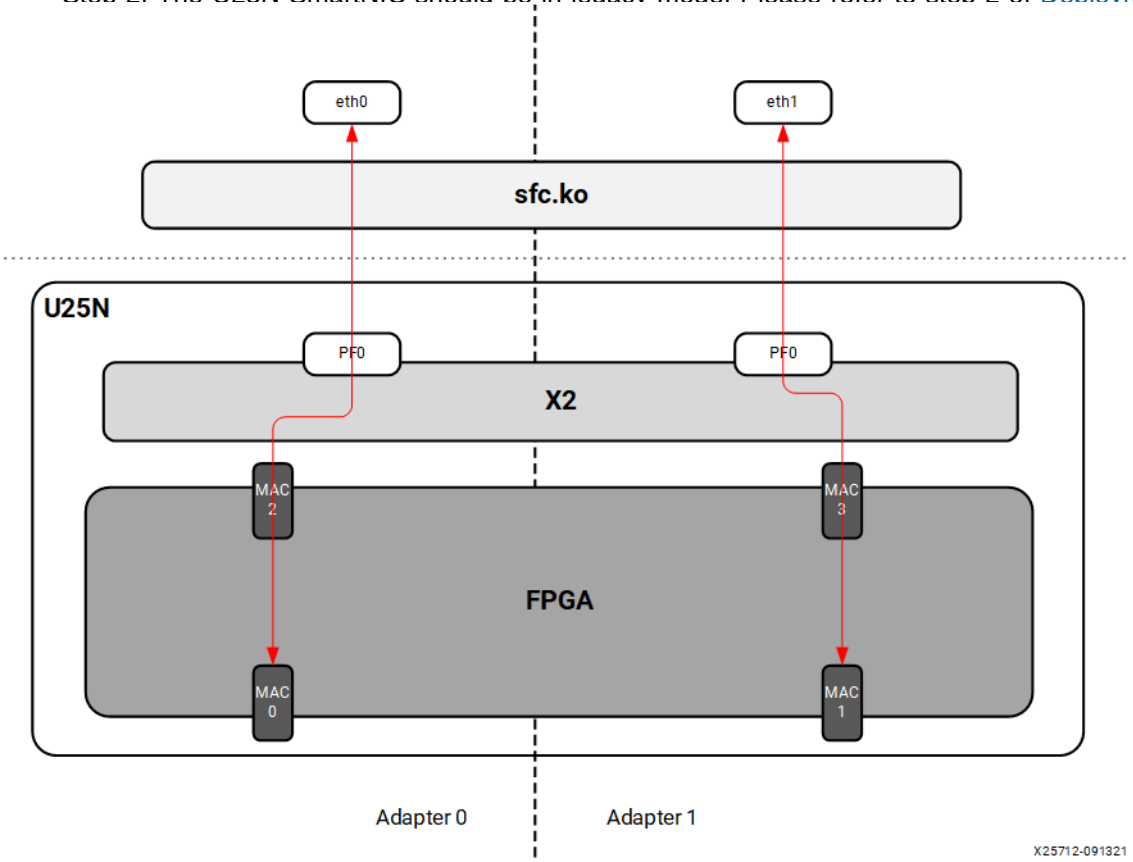
- Step 1. In case the U25N driver version is latest, this step can be ignored. Driver version of the U25N interface can be validated using the following command:

  ```
  ethtool -i <PF0_interface> | grep version
  ```

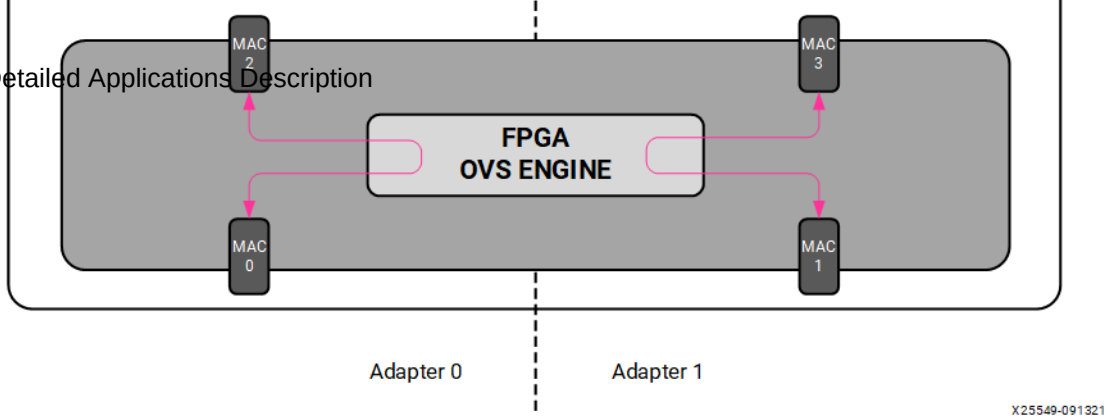
  ***Note*:** To install the latest sfc driver, refer to the U25N Driver.

  ***Note*:** Before flashing the image, make sure the SmartNIC is in legacy mode, no OVS software application is running, and OVS databases are removed.

- Step 2. The U25N SmartNIC should be in legacy mode. Please refer to step 2 of Deployment Image Flashing



any modifications on
the internal MAC3
uration.



re added to the PF0
e adapter or both. A
SR-IOV virtual ports

4.2 OVS

4.2.1 Installing OVS

OVS is a multilayer software switch licensed under the open source Apache 2 license. It implements a production quality switch platform that supports standard management interfaces and opens the forwarding functions to programmatic extension and control. OVS is well suited to function as a virtual switch in virtualized environments.

Follow the below-mentioned steps to install OVS. For additional infomation of OVS installation, please refer to Installing Open vSwitch

The OVS source code is available in its Git repository

```
git clone https://github.com/openvswitch/ovs.git
```

After cloning, the ovs directory will be in the current directory path. “cd” to the ovs directory as mentioned below:

```
cd ovs
```

Execute the following commands sequentially as the root user:

```
./boot.sh
./configure
make
make install
```

Export the OVS path:

```
export PATH=$PATH:/usr/local/share/openvswitch/scripts
```

Perform a version check:

```
ovs-vswitchd --version
```

***Note*:** Version 2.12 and 2.14 have been tested.

Maximum flows supported and tested: 8k

4.2.2 Classification Fields (Matches)

Supported Keys and Actions

```
Keys

1. ipv4/ipv6 src_ip

2. ipv4/ipv6 dst_ip

3. ip_tos

4. ip_proto

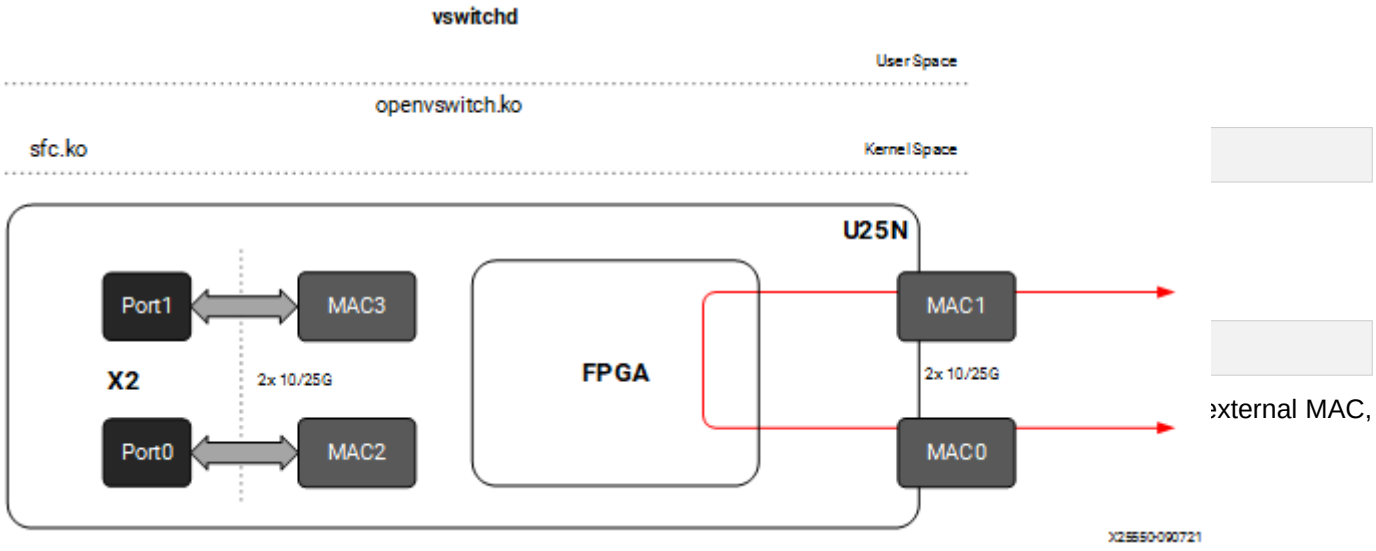
5. ovlan Outer

6. ivlan Inner
```

7. ether_type
8. tcp/udp src_port
9. tcp/udp dst_port
10. src_mac
11. dst_mac
12. vni
13. Ingress port

Actions

1. do_decap
2. do_decr_ip_ttl
3. do_src_mac
4. do_dst_mac
5. do_vlan_pop
6. do_vlan_push
7. do_encap



xternal MAC,

- Step 1. Refer to Basic Requirements and Component Versions Supported for the required OS/software version.
- Step 2. Check the driver version of U25N interface using the following command:

  ***Note*:** Ignore this step, if the U25N driver version is latest.

  ```
  ethtool -i u25eth0 | grep version
  ```

  ***Note*:** To install the latest sfc driver, refer to U25N Driver.
- Step 3. Put both U25N PF interfaces in ready state:

  List the interfaces

  ```
  ifconfig -a
  ```

Search for U25N interfaces using the `sfboot` command:

```
sfboot --list # shows the U25N interfaces
```

Example Output:

```
Adapter list:
   u25eth0
   u25eth1
```

Bring U25N interfaces up

```
ifconfig <PF_interface> up
```

For example:

```
ifconfig u25eth0 up
ifconfig u25eth1 up
```

- Step 4. Put the U25N PF interfaces into switchdev mode:

  ***Note*:** Ensure that the PF interface link is up before switching to switchdev mode.

  The `lspci | grep Sol` command gives us the PCIe® device bus ID required to execute the following command.

```
devlink dev eswitch set pci/0000:<pci_id> mode switchdev # pci_id is the BDF of U25N Device
```

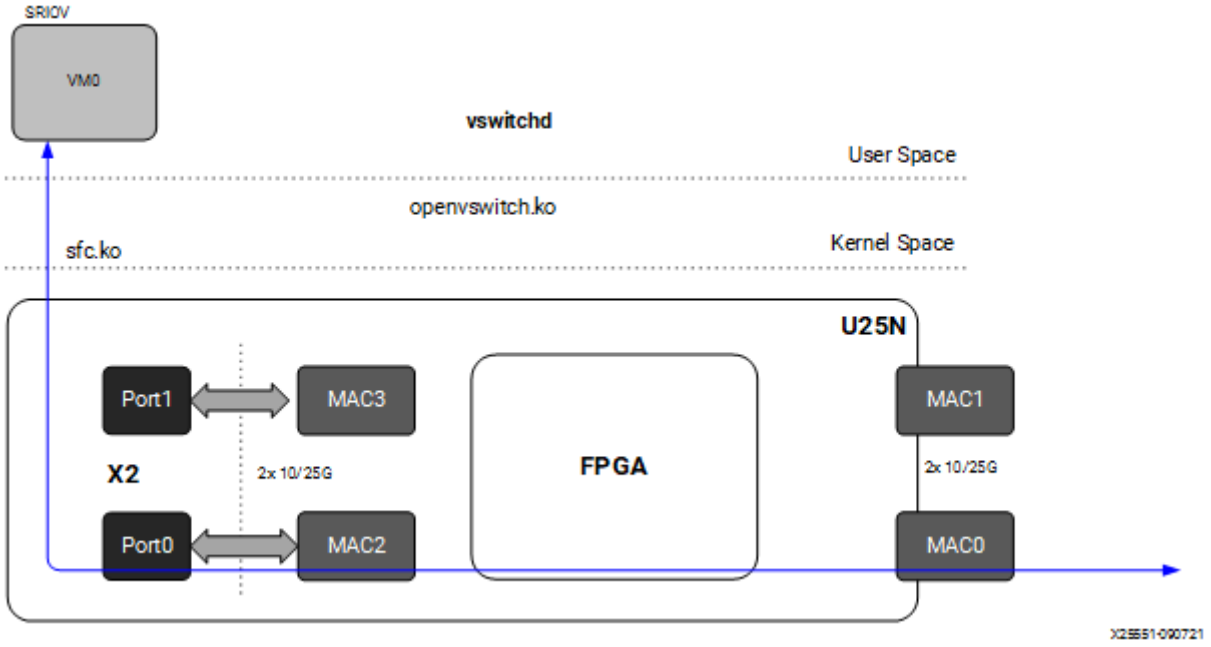
  Example Output:

```
devlink dev eswitch set pci/0000:af:00.0 mode switchdev
devlink dev eswitch set pci/0000:af:00.1 mode switchdev
```

  - Setp 5. Stop network manager

```
systemctl stop NetworkManager
```

  - Step 6. Follow the steps mentioned in OVS Configuration to create an OVS bridge. After creating the OVS bridge, continue with the next step.



  - Step 1. Refer to Basic Requirements and Component Versions Supported for the required OS/software version.

    For VM use cases, VFs need to be created for the corresponding PF for binding to the VM. The number of VF counts should be configured by the sfboot command with sriov_numvfs.

    ***Note*:** For more information, refer to sfboot Configuration.

  - Step 2. Check the driver version of U25N interface using the following command:

***Note*:** Ignore this step, if the U25N driver version is latest.

```
ethtool -i u25eth0 | grep version
```

***Note*:** To install the latest sfc driver, refer to U25N Driver.

- Step 3. Put both U25N PF interfaces in ready state:

  List the interfaces

  ```
  ifconfig -a
  ```

  Search for U25N interfaces using the `sfboot` command:

  ```
  sfboot --list # shows the U25N interfaces
  ```

  Example Output:

  ```
  Adapter list:
     u25eth0
     u25eth1
  ```

  Bring U25N interfaces up

  ```
  ifconfig <PF_interface> up
  ```

  For example:

  ```
  ifconfig u25eth0 up
  ifconfig u25eth1 up
  ```

Fs to the U25N PF. In the following command, a single VF is enabled on the PF0 interface:

be created on the PF1 interface based on the use case. The sriov_numvfs count should be less than or equal to the VF count specifie
nabled only in legacy mode. To check the U25N mode, excecute the following steps.

```
h show pci/0000:<pci_id> # pci_id is the BDF of U25N Device
```

```
ode legacy
```

nge to legacy mode using the following command:

```
h set pci/0000:<PCIe device bus id> mode legacy
```

f VFs.

```
s/net/<PF_interface>/device/sriov_numvfs
```

```
/net/u25eth0/device/sriov_numvfs
```

ve mentioned command, a VF PCIe ID and VF interface will be created. The VF PCIe device ID can be listed with the command `lspci`

ID is

```
ontroller: Solarflare Communications XtremeScale SFC9250 10/25/40/50/100G Ethernet Controller (Virtu
 used for binding the VF to a VM.
```

- Step 5. The VF interface can be found using the `ifconfig -a` command. To differentiate VF from PF, use the ip link show command. This gives the VF interface ID and VF interface mac address under the PF interface.
- Step 6. Ensure that the VF interface is up:

  ```
  ifconfig <VF_interface> up
  ```

- Step 7. Ensure that the PF interface is in switchdev mode:

***Note*:** Ensure the PF interface link is up before doing switchdev mode.

The `lspci | grep Sol` command gives the PCIe device bus ID.

```
devlink dev eswitch set pci/0000:<pci_id> mode switchdev # pci_id is the BDF of U25N Device
```

For example:

```
devlink dev eswitch set pci/0000:af:00.0 mode switchdev
```

- Step 8. Running the above command creates a VF representor interface. The VF representor interface name will be the PF interface name followed by `_0` for the first VF representor and `_1` for the second V representor, and so on.

  Note*:The total number of VF representor interfaces created are based on the sriov_numvfs value configured above.

  ```
  ip link show | grep <PF_interface>
  ```

  For example:

  ```
  ip link show | grep u25eth0
  ```

  ***Note*:** Here u25eth0 is the PF interface and u25eth0_0 is the VF representor interface.

  Now make the VF representor interface up using the ifconfig command:

  ```
  ifconfig <VF_rep_interface> up
  ```

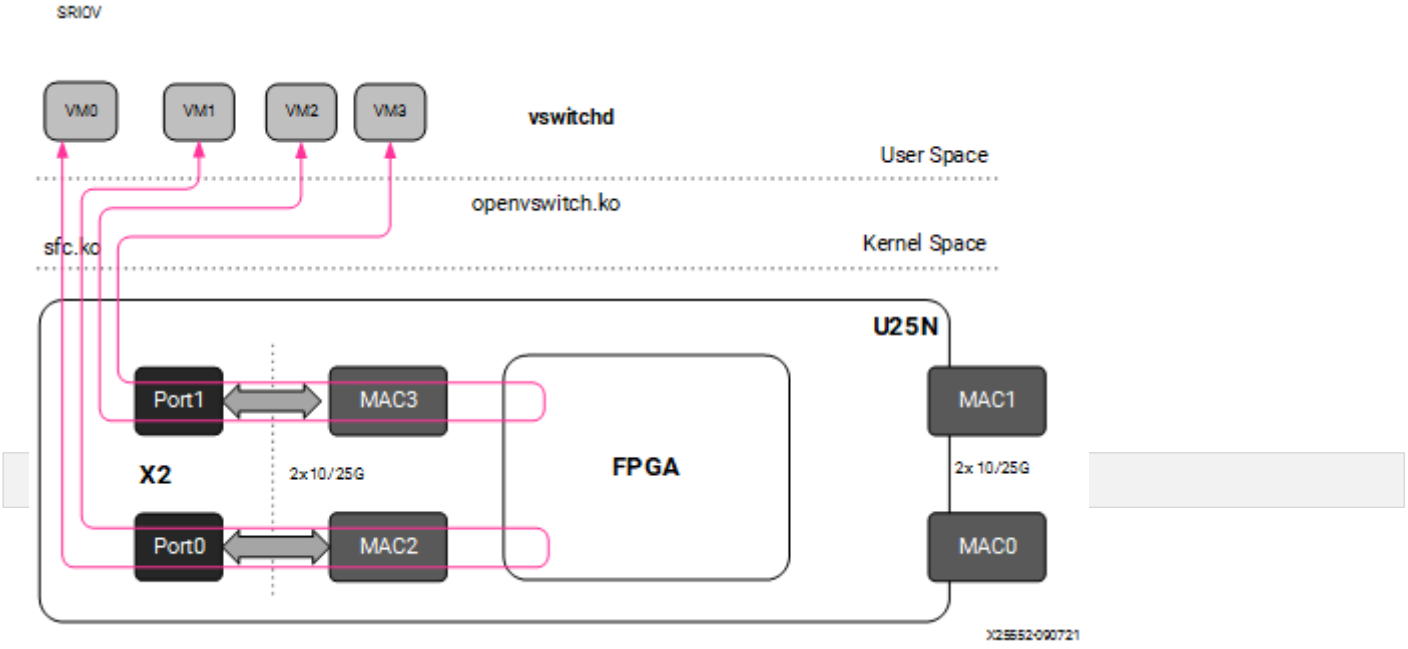
- Step 9. Follow the steps mentioned in OVS Configuration to create an OVS bridge. After creating the OVS bridge, proceed to the next step.
- Step 10. Add PF interfaces as ports to the OVS bridge:

  ```
  ovs-vsctl add-port <bridge_name> <PF0_interface>
  ```

  For example:

  ```
  ovs-vsctl add-port br0 u25eth0
  ```

- Step 11. Add a VF representor interface as a port to the OVS bridge:



- Step 1. Refer to Basic Requirements and Component Versions Supported for the required OS/software version. For VM use cases, VFs need to be created for the corresponding PF for binding to the VM. The number of VF counts should be configured by the sfboot command with sriov_numvfs.

***Note*:** For more information, refer to sfboot Configuration.

- Step 2. Check the driver version of U25N interface using the following command:

  ***Note*:** Ignore this step, if the U25N driver version is latest.

  ```
  ethtool -i u25eth0 | grep version
  ```

  ***Note*:** To install the latest sfc driver, refer to U25N Driver.

- Step 3. Put both U25N PF interfaces in ready state:

  List the interfaces

  ```
  ifconfig -a
  ```

  Search for U25N interfaces using the `sfboot` command:

  ```
  sfboot --list # shows the U25N interfaces
  ```

  Example Output:

  ```
  Adapter list:
     u25eth0
     u25eth1
  ```

  Bring U25N interfaces up

  ```
  ifconfig <PF_interface> up
  ```

  For example:

  ```
  ifconfig u25eth0 up
  ifconfig u25eth1 up
  ```

Fs to the U25N PF. In the following command, two VFs are enabled on the PF0 interface:

be created on the PF1 interface based on the use case. The sriov_numvfs count should be less than or equal to the VF count specifie
nabled only in legacy mode. To check the U25N mode, excecute the following steps.

```
h show pci/0000:<pci_id> # pci_id is the BDF of U25N Device


ode legacy
```

nge to legacy mode using the following command:

```
h set pci/0000:<PCIe device bus id> mode legacy
```

f VFs.

```
s/net/<PF_interface>/device/sriov_numvfs


/net/u25eth0/device/sriov_numvfs
```

ve mentioned command, a VF PCIe ID and VF interface will be created. The VF PCIe device ID can be listed with the command `lspci`

ID is

```
ontroller: Solarflare Communications XtremeScale SFC9250 10/25/40/50/100G Ethernet Controller (Virtu
 used for binding the VF to a VM.
```

- Step 5. The VF interface can be found using the `ifconfig -a` command. To differentiate VF from PF, use the ip link show command. This gives the VF interface ID and VF interface mac address under the PF interface.
- Step 6. Ensure that the VF interface is up:

```
ifconfig <VF_interface> up
```

- Step 7. Ensure that the PF interfaces are in switchdev mode.

  ***Note*:** Ensure the PF interface link is up before doing switchdev mode.

  The `lspci | grep Sol` command gives the PCIe device bus ID:

  ```
  devlink dev eswitch set pci/0000:<PCIe device bus id> mode switchdev
  ```

  For example:

  ```
  devlink dev eswitch set pci/0000:af:00.0 mode switchdev
  ```

- Step 8. Running the above command creates two VF representor interfaces. The VF representor interface name will be the PF interface name followed by `_0` for the first VF representor and `_1` for the second V representor, and so on.

  ***Note*:**The total number of VF representor interfaces created are based on the sriov_numvfs value configured above.

  ```
  ip link show | grep <PF_interface>
  ```

  For example:

  ```
  ip link show | grep u25eth0

  u25eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
  mode DEFAULT group default qlen 1000
  u25eth0_0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
  master ovs-system state UP mode DEFAULT group default qlen 1000
  u25eth0_1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
  master ovs-system state UP mode DEFAULT group default qlen 1000
  ```

  ***Note*:** Here u25eth0 is the PF interface, and u25eth0_0 and u25eth0_1 are the VF representor interfaces.

  Now make the VF representor interfaces up using the ifconfig command:

  ```
  ifconfig <VF_rep_interface> up
  ```

- Step 9. Follow the steps mentioned in OVS Configuration to create an OVS bridge. After creating the OVS bridge, proceed to the next step.
- Step 10. Add two VF representor interfaces to the OVS bridge:

  ```
  ovs-vsctl add-port <bridge_name> <VF_rep_interface 1>
  ovs-vsctl add-port <bridge_name> <VF_rep_interface 2>
  ```

  For example:

  ```
  ovs-vsctl add-port br0 u25eth0_0
  ovs-vsctl add-port br0 u25eth0_1
  ```

- Step 11. Ensure that the the OVS bridge is up:

  ```
  ifconfig <bridge_name> up
  ```

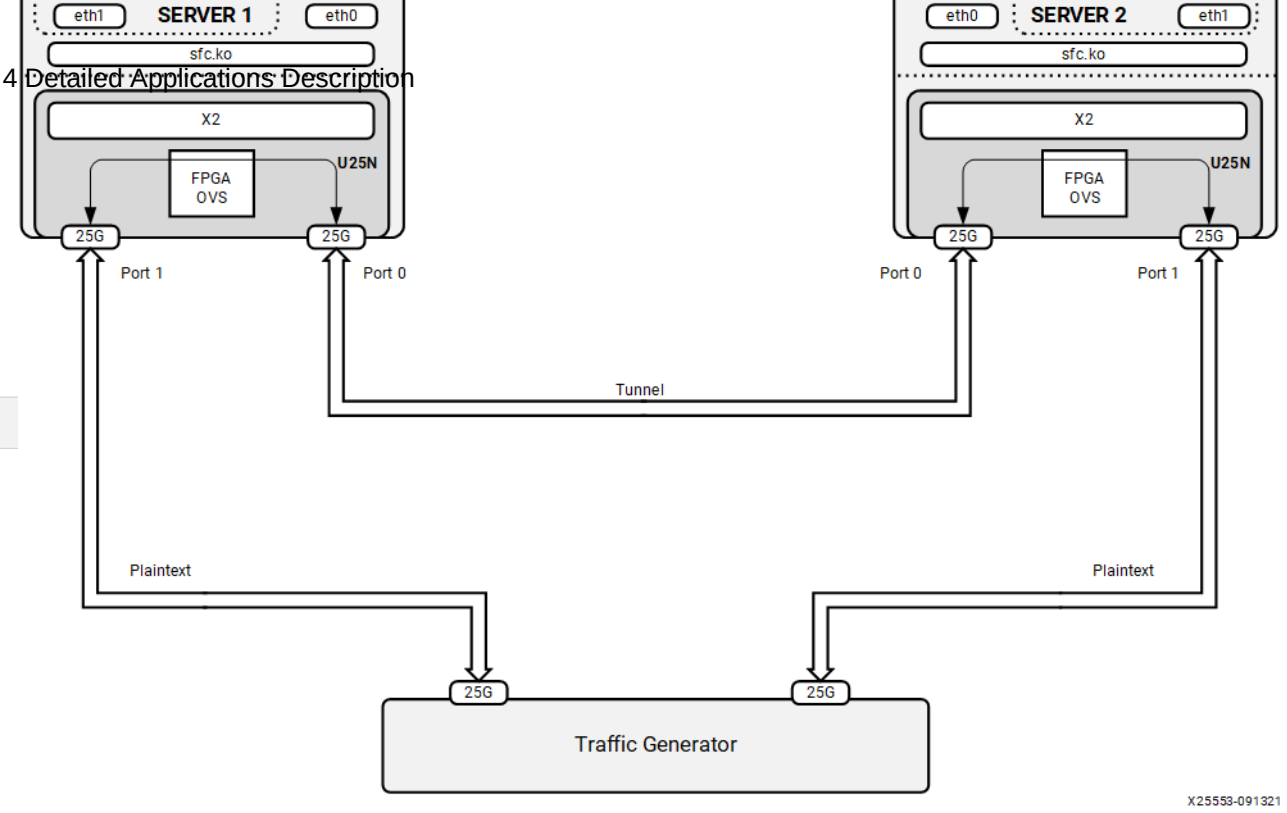
- Step 12. Print the brief overview of the database contents:

  ```
  ovs-vsctl show
  ```

- Step 14. Refer VM Installation to instantiate the VM.
- Step 15. Refer to Functionality Check to check OVS functionality.

4.2.7 Tunnels (Encapsulation/Decapsulation)

U25N Smart NIC supports offloading of tunnels using encapsulation and decapsulation actions.



ı L2GRE
acts as a

L2GRE Server 1 Configuration

- Step 1. Refer to Basic Requirements and Component Versions Supported for the required OS/software version.
- Step 2. Check the driver version of U25N interface using the following command:

  ***Note*:** Ignore this step, if the U25N driver version is latest.

  ```
  ethtool -i u25eth0 | grep version
  ```

  ***Note*:** To install the latest sfc driver, refer to U25N Driver.
- Step 3. Put both U25N PF interfaces in ready state:

  List the interfaces

  ```
  ifconfig -a
  ```

  Search for U25N interfaces using the `sfboot` command:

  ```
  sfboot --list # shows the U25N interfaces
  ```

  Example Output:

  ```
  Adapter list:
     u25eth0
     u25eth1
  ```

  Bring U25N interfaces up

  ```
  ifconfig <PF_interface> up
  ```

  For example:

  ```
  ifconfig u25eth0 up
  ifconfig u25eth1 up
  ```
- Step 4. Assign tunnel local IP to PF0 interface:

  ```
  ifconfig <PF0_interface> <local_ip> up
  ```

  For example:

  ```
  ifconfig u25eth0 10.16.0.2/24 up
  ```

- Step 5. Put the U25N PF0 interface into switchdev mode:

  ***Note*:** Ensure that the PF interface link is up before switching to switchdev mode.

  The `lspci | grep Sol` command gives us the PCIe® device bus ID required to execute the following command.

  ```
  devlink dev eswitch set pci/0000:<pci_id> mode switchdev # pci_id is the BDF of U25N Device
  ```

  Example Output:

  ```
  devlink dev eswitch set pci/0000:af:00.0 mode switchdev
  ```

  - Step 6. Follow the steps mentioned in OVS Configuration to create an OVS bridge. After creating the OVS bridge, continue with the next step.
  - Step 7. Create GRE interfaces:

    ```
    ovs-vsctl add-port <bridge_name> gre0 -- set interface gre0 type=gre
    options:local_ip=<ip_address> options:remote_ip=<ip_address>
    ```

    For example:

    ```
    ovs-vsctl add-port br0 gre0 -- set interface gre0 type=gre
    options:local_ip=10.16.0.2 options:remote_ip=10.16.0.1
    ```

  - Step 8. Add a PF1 interface as a port to the OVS bridge:

    ```
    ovs-vsctl add-port <bridge_name> <PF1_interface>
    ```

    For example:

    ```
    ovs-vsctl add-port br0 u25eth1
    ```

  - Step 9. Ensue the OVS bridge is up:

    ```
    ifconfig <bridge_name> up
    ```

    For example:

    ```
    ifconfig br0 up
    ```

  - Step 10. Print a brief overview of the database contents:

    ```
    ovs-vsctl show
    ```

L2GRE Server 2 Configuration

The setup steps for server 2 is as same as server 1 except for local_ip and remote_ip in step 4 and step 6.

In step 4, the tunnel local IP assigned to PF0 interface should be the tunnel remote IP in above server 1 configuration.

For example:

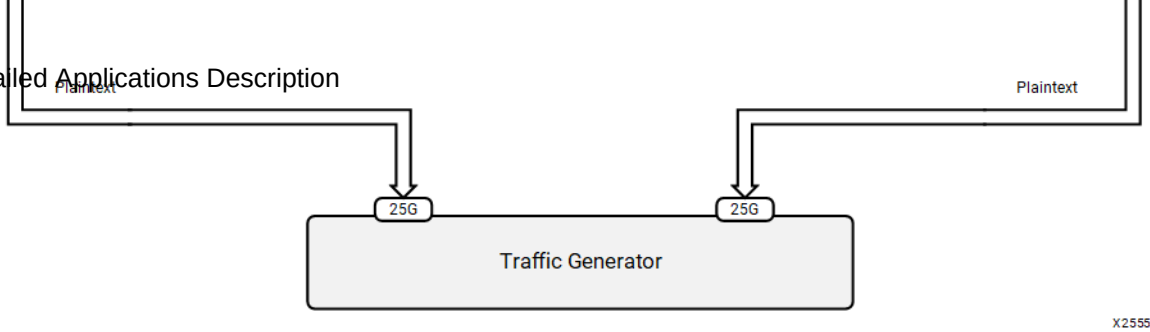
```
ifconfig u25eth0 10.16.0.1/24 up
```

In step 6, the tunnel local IP and remote IP should be swapped in above server 1 configuration.

```
ovs-vsctl add-port br0 gre0 -- set interface gre0 type=gre
options:local_ip=10.16.0.1 options:remote_ip=10.16.0.2
```

4.2.7.2 VXLAN

- Maximum tunnel support = 1K
- Maximum supported flows = 8K
- Maximum MTU size = 1400



, VXLAN
ne tunnel

VXLAN Server 1 Configuration

- Step 1. Refer to Basic Requirements and Component Versions Supported for the required OS/software version.
- Step 2. Check the driver version of U25N interface using the following command:

  ***Note*:** Ignore this step, if the U25N driver version is latest.

  ```
  ethtool -i u25eth0 | grep version
  ```

  ***Note*:** To install the latest sfc driver, refer to U25N Driver.
- Step 3. Put both U25N PF interfaces in ready state:

  List the interfaces

  ```
  ifconfig -a
  ```

  Search for U25N interfaces using the `sfboot` command:

  ```
  sfboot --list # shows the U25N interfaces
  ```

  Example Output:

  ```
  Adapter list:
     u25eth0
     u25eth1
  ```

  Bring U25N interfaces up

  ```
  ifconfig <PF_interface> up
  ```

  For example:

  ```
  ifconfig u25eth0 up
  ifconfig u25eth1 up
  ```
- Step 4. Assign tunnel local IP to PF0 interface:

  ```
  ifconfig <PF0_interface> <local_ip> up
  ```

  For example:

  ```
  ifconfig u25eth0 10.16.0.2/24 up
  ```
- Step 5. Put the U25N PF0 interface into switchdev mode:

  ***Note*:** Ensure that the PF interface link is up before switching to switchdev mode.

  The `lspci | grep Sol` command gives us the PCIe® device bus ID required to execute the following command.

  ```
  devlink dev eswitch set pci/0000:<pci_id> mode switchdev # pci_id is the BDF of U25N Device
  ```

  Example Output:

  ```
  devlink dev eswitch set pci/0000:af:00.0 mode switchdev
  ```
- Step 5. Follow the steps mentioned in OVS Configuration to create an OVS bridge. After creating the OVS bridge, continue with the next step.
- Step 6. Create VXLAN interfaces:

```
ovs-vsctl add-port br0 vxlan0 -- set interface vxlan0 type=vxlan
options:local_ip=<ip_address> options:remote_ip=<ip_address> options:key=<key_id>
```

For example:

```
ovs-vsctl add-port br0 vxlan0 -- set interface vxlan0 type=vxlan
options:local_ip=10.16.0.2 options:remote_ip=10.16.0.1 options:key=123
```

- Step 7. Add a PF1 interface as a port to the OVS bridge:

```
ovs-vsctl add-port <bridge_name> <PF1_interface>
```

For example:

```
ovs-vsctl add-port br0 u25eth1
```

- Step 8. Ensue the OVS bridge is up:

```
ifconfig <bridge_name> up
```

For example:

```
ifconfig br0 up
```

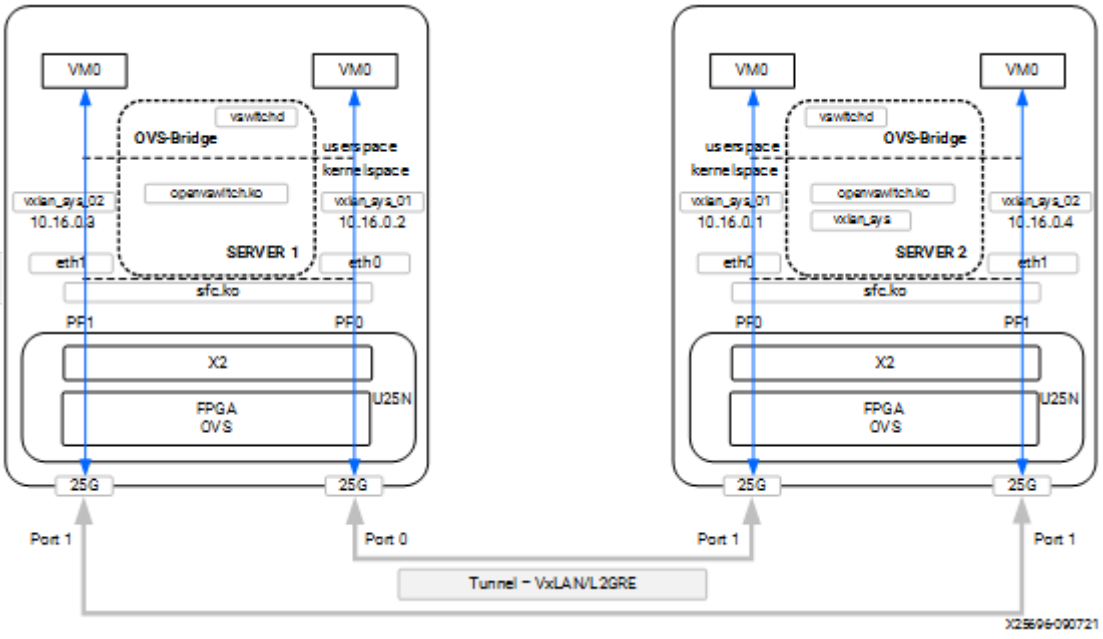
- Step 9. Print a brief overview of the database contents:

```
ovs-vsctl show
```

VXLAN Server 2 Configuration

The setup steps for server 2 is as same as server 1 except for local_ip and remote_ip in step 4 and step 6.

In step 4, the tunnel local IP assigned to PF0 interface should be the tunnel remote IP in above server 1 configuration.



Tunnel Server 1 Configuration

- Step 1. Refer to Basic Requirements and Component Versions Supported for the required OS/software version.

  For VM use cases, VFs need to be created for the corresponding PF for binding to the VM. The number of VF counts should be configured by the sfboot command with sriov_numvfs.

  ***Note*:** For more information, refer to sfboot Configuration.

- Step 2. Check the driver version of U25N interface using the following command:

***Note*:** Ignore this step, if the U25N driver version is latest.

```
ethtool -i u25eth0 | grep version
```

***Note*:** To install the latest sfc driver, refer to U25N Driver.

- Step 3. Put both U25N PF interfaces in ready state:

  List the interfaces

  ```
  ifconfig -a
  ```

  Search for U25N interfaces using the `sfboot` command:

  ```
  sfboot --list # shows the U25N interfaces
  ```

  Example Output:

  ```
  Adapter list:
     u25eth0
     u25eth1
  ```

  Bring U25N interfaces up

  ```
  ifconfig <PF_interface> up
  ```

  For example:

  ```
  ifconfig u25eth0 up
  ifconfig u25eth1 up
  ```

Fs to the U25N PF. In the following command, two VFs are enabled on the PF0 interface:

be created on the PF1 interface based on the use case. The sriov_numvfs count should be less than or equal to the VF count specifie
nabled only in legacy mode. To check the U25N mode, excecute the following steps.

```
h show pci/0000:<pci_id> # pci_id is the BDF of U25N Device
```

```
ode legacy
```

nge to legacy mode using the following command:

```
h set pci/0000:<PCIe device bus id> mode legacy
```

f VFs.

```
s/net/<PF_interface>/device/sriov_numvfs
```

```
/net/u25eth0/device/sriov_numvfs
/net/u25eth1/device/sriov_numvfs
```

ve mentioned command, a VF PCIe ID and VF interface will be created. The VF PCIe device ID can be listed with the command `lspci`

ID is

```
ontroller: Solarflare Communications XtremeScale SFC9250 10/25/40/50/100G Ethernet Controller (Virtu
 used for binding the VF to a VM.
```

- Step 5. The VF interface can be found using the `ifconfig -a` command. To differentiate VF from PF, use the ip link show command. This gives the VF interface ID and VF interface mac address under the PF interface.
- Step 6. Ensure that the VF interface is up:

  ```
  ifconfig <VF_interface> up
  ```

- Step 7. Ensure that the PF interfaces are in switchdev mode.

  ***Note*:** Ensure the PF interface link is up before doing switchdev mode.

  The `lspci | grep Sol` command gives the PCIe device bus ID:

  ```
  devlink dev eswitch set pci/0000:<PCIe device bus id> mode switchdev
  ```

  For example:

  ```
  devlink dev eswitch set pci/0000:af:00.0 mode switchdev
  devlink dev eswitch set pci/0000:af:00.1 mode switchdev
  ```

- Step 8. Running the above command creates two VF representor interfaces. The VF representor interface name will be the PF interface name followed by `_0` for the first VF representor and `_1` for the second V representor, and so on.

  ***Note*:**The total number of VF representor interfaces created are based on the sriov_numvfs value configured above.

  ```
  ip link show | grep <PF_interface>
  ```

  For example:

  ```
  ip link show | grep u25eth

  u25eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
  mode DEFAULT group default qlen 1000
  u25eth0_0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
  master ovs-system state UP mode DEFAULT group default qlen 1000
  ip link show | grep <u25eth1>
  u25eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
  mode DEFAULT group default qlen 1000
  u25eth1_0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
  master ovs-system state UP mode DEFAULT group default qlen 1000
  ```

  ***Note*:** Here u25eth0 and u25eth1 is the PF interface, and u25eth0_0 and u25eth1_0 are the VF representor interfaces.

  Now make the VF representor interfaces up using the ifconfig command:

  ```
  ifconfig <VF_rep_interface> up
  ```

- Step 9. Follow the steps mentioned in OVS Configuration to create an OVS bridge. After creating the OVS bridge, proceed to the next step.

RE interfaces:

ation is for the VXLAN. Similarly, the GRE tunnel could also be used.

```
r0 vxlan0 -- set interface vxlan0 type=vxlan options:local_ip=<ip_address> options:remote_ip=<ip_add
r1 vxlan1 -- set interface vxlan0 type=vxlan options:local_ip=<ip_address> options:remote_ip=<ip_add
```

```
r0 vxlan0 -- set interface vxlan0 type=vxlan options:local_ip=10.16.0.2 options:remote_ip=10.16.0.1
r1 vxlan1 -- set interface vxlan type=vxlan options:local_ip=10.16.0.3 options:remote_ip=10.16.0.4 o
```

- Step 11. Add VF representor enabled on each PF interface to a separate OVS bridge.

  ```
  ovs-vsctl add-port <bridge_name_0> <VF_rep_interface 1>
  ovs-vsctl add-port <bridge_name_1> <VF_rep_interface 2>
  ```

  For example:

  ```
  ovs-vsctl add-port br0 u25eth0_0
  ovs-vsctl add-port br1 u25eth1_0
  ```

- Step 12. Ensure the the two bridges are up:

```
ifconfig <bridge_name> up
```

For example:

```
ifconfig br0 up
ifconfig br1 up
```

- Step 13. Print a brief overview of the database contents:

```
ovs-vsctl show
```

- Step 14. Refer VM Installation to instantiate the VM.

Tunnel Server 2 Configuration

The setup steps for server 2 is as same as server 1 except for local_ip and remote_ip in step 10.

In step 10, the tunnel local IP and remote IP should be swapped in above server 1 configuration.

```
ovs-vsctl add-port br0 vxlan0 -- set interface vxlan0 type=vxlan options:local_ip=10.16.
ovs-vsctl add-port br1 vxlan1 -- set interface vxlan type=vxlan options:local_ip=10.16.0
```

4.2.8 LACP

4.2.8.1 LAG Creation

- Step 1. Refer to Basic Requirements and Component Versions Supported for the required OS/software version.
- Step 2. Check the driver version of U25N interface using the following command:

  ***Note*:** Ignore this step, if the U25N driver version is latest.

```
ethtool -i u25eth0 | grep version
```

  ***Note*:** To install the latest sfc driver, refer to U25N Driver.

- Step 3. Put both U25N PF interfaces in ready state:

  List the interfaces

```
ifconfig -a
```

  Search for U25N interfaces using the `sfboot` command:

```
sfboot --list # shows the U25N interfaces
```

  Example Output:

```
Adapter list:
   u25eth0
   u25eth1
```

  Bring U25N interfaces up

```
ifconfig <PF_interface> up
```

  For example:

```
ifconfig u25eth0 up
ifconfig u25eth1 up
```

- Step 4. Put the U25N PF interfaces into switchdev mode:

  ***Note*:** Ensure that the PF interface link is up before switching to switchdev mode.

  The `lspci | grep Sol` command gives us the PCIe® device bus ID required to execute the following command.

```
devlink dev eswitch set pci/0000:<pci_id> mode switchdev # pci_id is the BDF of U25N Device
```

Example Output:

```
devlink dev eswitch set pci/0000:af:00.0 mode switchdev
devlink dev eswitch set pci/0000:af:00.1 mode switchdev
```

Fs to the U25N PF. In the following command, two VFs are enabled on the PF0 interface:

be created on the PF1 interface based on the use case. The sriov_numvfs count should be less than or equal to the VF count specifie
nabled only in legacy mode. To check the U25N mode, excecute the following steps.

```
h show pci/0000:<pci_id> # pci_id is the BDF of U25N Device


ode legacy
```

nge to legacy mode using the following command:

```
h set pci/0000:<PCIe device bus id> mode legacy
```

f VFs.

```
s/net/<PF_interface>/device/sriov_numvfs


/net/u25eth0/device/sriov_numvfs
/net/u25eth1/device/sriov_numvfs
```

ve mentioned command, a VF PCIe ID and VF interface will be created. The VF PCIe device ID can be listed with the command `lspci`

ID is

```
ontroller: Solarflare Communications XtremeScale SFC9250 10/25/40/50/100G Ethernet Controller (Virtu
 used for binding the VF to a VM.
```

- Step 6. The VF interface can be found using the `ifconfig -a` command. To differentiate VF from PF, use the ip link show command. This gives the VF interface ID and VF interface mac address under the PF interface.
- Step 7. Ensure that the VF interface is up:

  ```
  ifconfig <VF_interface> up
  ```

- Step 8. Ensure that the PF interfaces are in switchdev mode.

  ***Note*:** Ensure the PF interface link is up before doing switchdev mode.

  The `lspci | grep Sol` command gives the PCIe device bus ID:

  ```
  devlink dev eswitch set pci/0000:<PCIe device bus id> mode switchdev
  ```

  For example:

  ```
  devlink dev eswitch set pci/0000:af:00.0 mode switchdev
  devlink dev eswitch set pci/0000:af:00.1 mode switchdev
  ```

- Step 9. Running the above command creates two VF representor interfaces. The VF representor interface name will be the PF interface name followed by `_0` for the first VF representor and `_1` for the second V representor, and so on.

  ***Note*:**The total number of VF representor interfaces created are based on the sriov_numvfs value configured above.

  ```
  ip link show | grep <PF_interface>
  ```

  For example:

```
ip link show | grep u25eth

u25eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
mode DEFAULT group default qlen 1000
u25eth0_0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
master ovs-system state UP mode DEFAULT group default qlen 1000
ip link show | grep <u25eth1>
u25eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
mode DEFAULT group default qlen 1000
u25eth1_0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel
master ovs-system state UP mode DEFAULT group default qlen 1000
```

***Note*:** Here u25eth0 and u25eth1 is the PF interface, and u25eth0_0 and u25eth1_0 are the VF representor interfaces.

Now make the VF representor interfaces up using the ifconfig command:

```
ifconfig <VF_rep_interface> up
```

- Step 10: Creating PF bond.

  Install the bonding driver

  ```
  modprobe bonding
  ```

  Create a PF bond interface. Then set bond mode as 802.3ad and xmit hash policy as layer 3+4

  ```
  ip link add bond0 type mode 802.3ad xmit_hash_policy layer3+4
  ```

  Make both PF interfaces down

  ```
  ifconfig <PF_interface> down
  ```

  Example Output:

  ```
  ifconfig u25eth0 down
  ifconfig u25eth1 down
  ```

  Adding PF slave interfaces to master bond interface

  ```
  echo "+u25eth0" > /sys/class/net/bond0/bonding/slaves
  echo "+u25eth1" > /sys/class/net/bond0/bonding/slaves
  ```

  Make the bond interface up

  ```
  ifconfig bond0 up
  ```

  Now the bond interface should show the supported rate as 50Gbps

  ```
  ethtool bond0
  ```

  Master bond and its corresponding slave interfaces would have the same MAC address

- Step 11: Creating a VF bond

  Create a VF bond interface. Then set bond mode as balance-xor mode and xmit hash policy as layer 3+4

  ```
  ip link add vfbond0 type mode balance-xor xmit_hash_policy layer3+4
  ```

  Make both VF interfaces down

  ```
  ifconfig <vf_interface> down
  ```

  Adding VF slave interfaces to master VF bond interface

  ```
  echo "+u25eth0n0" > /sys/class/net/vfbond0/bonding/slaves
  echo "+u25eth1n0" > /sys/class/net/vfbond0/bonding/slaves
  ```

  Make the VF bond interface up

  ```
  ifconfig vfbond0 up
  ```

Now the bond interface should show the supported rate as 50Gbps

```
ethtool vfbond0
```

Master vfbond and its corresponding slave interfaces would have the same MAC address

- Step 12: Creating VF_Rep bond

  Create a VF_Rep bond interface. Then set bond mode as balance-rr mode

  ```
  ip link add repbond0 type mode balance-rr
  ```

  Make both VF_rep interfaces down

  ```
  ifconfig <VF_rep_interface> down
  ```

  Adding VF_Rep slave interfaces to master VF_Rep bond interface

  ```
  echo "+u25eth0_0" > /sys/class/net/repbond0/bonding/slaves
  echo "+u25eth1_0" > /sys/class/net/repbond0/bonding/slaves
  ```

  Make the VF_Rep bond interface up

  ```
  ifconfig repbond0 up
  ```

  Now the VF_Rep bond interface should show the supported rate as 50Gbps

  ```
  ethtool repbond0
  ```

  Master VF_Rep bond and its corresponding slave interfaces would have the same MAC address

- Step 13: Follow the steps mentioned in OVS Configuration to create an OVS bridge. After creating the OVS bridge, proceed to the next step.
- Step 14: Adding VF bond and VR_rep bond to OVS bridges.

  ```
  ovs-vsctl add-port br0 bond0
  ovs-vsctl add-port br0 repbond0
  ```

- Step 15: Ensure that the bridge is up:

  ```
  ifconfig <bridge_name> up
  ```

  For example:

  ```
  ifconfig br0 up
  ```

- Step 16: Print a brief overview of the database contents:

  ```
  ovs-vsctl show
  ```

- Step 17: Refer VM Installation to instantiate the VM with macvtap.
- Step 18: Post successful instantiation refer Functionality Check to validate functionality.

4.2.8.2 LAG Deletion

- Step 1: Delete ovs-bridge

  ```
  ovs-vsctl del-br <bridge_name>
  ```

- Step 2: Delete VF_Rep bond interface

  ```
  echo "-repbond_interface" > /sys/class/net/bonding_masters
  ```

- Step 3: Delete VF bond interface

  ```
  echo "-vfbond_interface" > /sys/class/net/bonding_masters
  ```

- Step 4: Delete PF bond interface

  ```
  echo "-bond_interface" > /sys/class/net/bonding_masters
  ```

  Example output:

```
echo "-bond0" > /sys/class/net/bonding_masters
```

- Step 5: After deleting bond interfaces, remove driver module

```
rmmod sfc
```

4.2.9 Connection Tracking

***Note*:** Refer to OVS Conntrack Tutorial for detail of OVS connection tracking feature.

Before using connection tracking feature, please install below kernel modules.

```
modprobe sch_netem
modprobe nf_flow_table
modprobe nf_conntrack
modprobe nft_flow_offload
```

Below scenario uses Port to VM or VM to Port data path as an example. Conntrack feature works in Port to Port and VM to VM scenarios too.

Before below steps, please follow the step 1 to 14 in Port to VM or VM to Port to setup OVS and create PF and VF for the bridge.

Configuring TCP flow table timeout

```
echo 3600 > /proc/sys/net/netfilter/nf_flowtable_tcp_timeout
```

Maximum number of conntrack connections

```
echo 524288 > /proc/sys/net/netfilter/nf_conntrack_max
```

Then following below steps to configure connection tracking rules to the OVS bridges.

- Step 1. Clean all existing Openflow rules of the bridges if they have.

```
ovs-ofctl del-flows <br0>
ovs-ofctl del-flows <br1>
```

or testing PF0 and PF1:

```
0 "table=0, priority=1, arp, actions=normal"
0 "table=0, priority=3, icmp, actions=normal"
0 "table=0, priority=2, icmp6, actions=normal"
0 "table=0, priority=1, ip, actions=resubmit(,1)"
0 "table=0, priority=1, ipv6, actions=resubmit(,1)"
0 "table=0, priority=0, actions=drop"
```

```
0 "table=1, priority=1, ct_state=-trk, ip, actions=ct(table=1,zone=0x1)"
0 "table=1, priority=1, ct_state=-trk, ipv6, actions=ct(table=1)"
0 "table=1, priority=1, ct_state=+trk+new, ip, in_port=<PF_interface>, actions=ct(commit,zone=0x1),
0 "table=1, priority=1, ct_state=+trk+new, ip, in_port=<VF_interface>, actions=ct(commit,zone=0x1),
0 "table=1, priority=1, ct_state=+trk+new, ipv6, actions=ct(commit),normal"
0 "table=1, priority=1, ct_state=+trk+est, ip, in_port=<PF_interface>, actions=output:\"<VF_interfa
0 "table=1, priority=1, ct_state=+trk+est, ip, in_port=<VF_interface>, actions=output:\"<PF_interfa
0 "table=1, priority=1, ct_state=+trk+est, ipv6, actions=normal"
0 "table=1, priority=0, actions=drop"
```

4.2.10 OVS Configuration

- Step 1. Export the OVS path:

```
export PATH=$PATH:/usr/local/share/openvswitch/scripts
export PATH=$PATH:/usr/local/bin
```

- Step 2. Stop OVS and remove any database to get rid of old configurations:

```
ovs-ctl stop
rm /usr/local/etc/openvswitch/conf.db
```

- Step 3. Start OVS:

```
ovs-ctl start
```

- Step 4. Enable hardware offload:

```
ovs-vsctl set Open_vSwitch . other_config:hw-offload=true
ovs-vsctl set Open_vSwitch . other_config:tc-policy=none
```

- Step 5. Set OVS log levels (for debug purpose only, if needed):

```
ovs-appctl vlog/set ANY:ANY:dbg
ovs-appctl vlog/set poll_loop:ANY:OFF
ovs-appctl vlog/set netlink_socket:ANY:OFF
```

that idle flows remain cached in the datapath:

```
 idle flows will remain cached in the datapath
$(ovs-vsctl list open_vswitch | grep _uuid | cut -f2 -d ":" | tr -d ' ') other_config:max-idle=60000
 revalidator threads will wait for kernel statistics before executing flow revalidation
. other_config:max-revalidator=10000
tware datapaths to use for handling new flows. The default value is the number of online CPU cores m
. other_config:n-handler-threads=8
hreads for software datapaths to use for revalidating flows in the datapath.
. other_config:n-revalidator-threads=4
```

- Step 7. After adding the hardware offload policy, restart OVS:

```
ovs-ctl restart
```

- Step 8. Print a brief overview of the database contents:

```
ovs-vsctl show
```

- Step 9. Adding bridge to OVS:

```
ovs-vsctl add-br <bridge_name>
```

For Example:

```
ovs-vsctl add-br br0
```

***Note*:** For VM to VM or VM to Port or Port to VM Tunnel alone create two OVS bridges. For example: `ovs-vsctl add-br br0` and `ovs-vsctl add-br br1`.

4.2.11 Functionality Check

After adding the U25N network interfaces to the OVS bridge, the functionality can be verified using ping, iperf, and dpdk network performance tools.

Ping Test

- Step 1. Assign the IP address to the respective interface and do a ping using the following command:

```
ping <remote_ip>
```

- Step 2. After a successful ping test, iperf can be used:

  ***Note*:** For VXLAN and L2GRE, set the MTU size to 1400 before running iperf3 or pktgen on a particular interface.

```
ifconfig <interface> mtu 1400 [as root]
```

- Step 3. Run `iperf3 -s` on the host device [iperf server].

- Step 4. Run `iperf3 -c <ip address>` on a remote device [iperf client].

4.2.12 Uninstalling OVS Setup

To uninstall the setup, please follow the instructions below.

- Step 1. power off all running VMs.

  ```
  killall qemu-system-x86
  ```

- Step2. Delete the OVS bridge

  ```
  ovs-vsctl del-br < Bridge_nam >
  ```

- Step3. Delete the bond interface if U25N's interface is the slave interface of this bond interface.

  ```
  ip link del < bond_name >
  ```

- Step4. Make all PF interfaces up

  ```
  ifconfig < interface_name > up
  ```

- Step5. Change mode from switchdev to legacy

  ```
  devlink dev switch set pci/0000:<pci> mode legacy
  ```

- Step6. Remove all VFs using the echo command

  ```
  echo 0 > /sys/class/net/< interface_name >/device/sriov_numvfs
  ```

- Step7. uninstall the driver module

  ```
  rmmod sfc
  ```

***Note*:** Refer to DPDK on U25N to run dpdk-testpmd.

4.3 IPsec

4.3.1 Supported XFRM Parameters

IPsec tunnels are created between two servers. Because IPsec is in *transport mode*, L2GRE is used to create tunnels. The strongSwan application runs in user space. The charon plugin of strongSwan is used to offload rules on the U25N. Packets reaching the IPsec module should be L2GRE encapsulated.

- Encryption algorithm: AES-GCM 256 encryption/decryption
- IPsec mode: Transport mode.
- Maximum IPsec tunnel supported: 32

4.3.2 Classification Fields (Matches)

4.3.2.1 Encryption

```
Key

1. IPv4 source address

2. IPv4 destination address

3. IP4 protocol Action

Actions

1. Action flag

2. SPI

3. Key

4. IV
```

4.3.2.2 Decryption

```
Keys

1. IPv4 source address

2. IPv4 destination address

3. SPI (Security Parameter Index)

Actions

1. Decryption key

2. IV
```

4.3.3 strongSwan Installation

***Note*:** Before installing the Debian package for strongSwan, make sure all the dependencies are installed.

```
apt-get install aptitude opensc libgmp10 libgmp-dev libssl-dev
```

- Step 1. Refer to Basic Requirements and Component Versions Supported for the required OS/software version.
- Step 2. Use the following commands to validate the version of the strongSwan Debian package.

  The version can be found using the command `swanctl --version`. Remove the already installed package before installing the latest one:

  ```
  dpkg -r strongswan_5.8.4-1_amd64
  dpkg -i strongswan_5.8.4-1_amd64.deb
  ```

- Step 3. After installing the strongSwan package, create a CA certificate. CA certificate can be created in one server and can be copied to the other server.

4.3.3.1 IPSec Server 1 Configuration

- Step 1. Generating a self-signed CA certificate using the PKI utility of strongSwan:

```
-size 4096 --outform pem > private/strongswanKey.pem
etime 3650 --in private/strongswanKey.pem --type rsa --dn "C=CH, O=strongSwan, CN=Root CA" --outform
```

- Step 2. After the key and certificate are generated in server 1, copy them to server 2 in the same directory.

  Copy the file `strongswanKey.pem` present in `/etc/ipsec.d/private/` from the first server to the second server at the same location.

  Copy the file `strongswanCert.pem` presenti `n /etc/ipsec.d/cacerts/strongswanCert.pem` from the first server to the second server at the same location.

  After finishing the above, create a key pair and certificate for each server separately as root.

```
cacert cacerts/strongswanCert.pem --cakey private/strongswanKey.pem --dn "C=CH, O=strongSwan, CN=dev
```

- Step 4. Configure the .conf file and secret file in server 1:

  In /etc/ipsec.conf, add below text in the beginning of the file

```
conn hw_offload #
      left=10.16.0.2
      right=10.16.0.1
      ike=aes256gcm16-sha256-modp2048
      esp=aes256gcm16-modp2048
      keyingtries=%forever
      ikelifetime=8h
      lifetime=8h
      dpddelay=1h
      dpdtimeout=1h
      dpdaction=restart
      auto=route
      keyexchange=ikev2
      type=transport
      leftcert=client1Cert.pem
      leftsendcert=always
      hw_offload=yes
      leftid="C=CH, O=strongSwan, CN=device1"
      rightid="C=CH, O=strongSwan, CN=device2"
      leftprotoport=gre
      rightprotoport=gre
```

  In /etc/ipsec.secrets, add below line in the end of the file

```
: RSA client1Key.pem
```

  ***Note*:** There is a white space, present between : and RSA.

4.3.3.2 IPSec Server 2 Configuration

```
cacert cacerts/strongswanCert.pem --cakey private/strongswanKey.pem --dn "C=CH, O=strongSwan, CN=dev
```

- Step 2. Configure the conf file and secret file in server 2:

  In /etc/ipsec.conf, add below text in the beginning of the file

```
conn hw_offload #
      left=10.16.0.1
      right=10.16.0.2
      ike=aes256gcm16-sha256-modp2048
      esp=aes256gcm16-modp2048
      keyingtries=%forever
      ikelifetime=8h
      lifetime=8h
      dpddelay=1h
      dpdtimeout=1h
      dpdaction=restart
      auto=route
      keyexchange=ikev2
      type=transport
      leftcert=client2Cert.pem
      leftsendcert=always
      hw_offload=yes
      leftid="C=CH, O=strongSwan, CN=device2"
      rightid="C=CH, O=strongSwan, CN=device1"
      leftprotoport=gre
      rightprotoport=gre
```

In /etc/ipsec.secrets, add below line in the end of the file

```
: RSA client1Key.pem
```

***Note*:** There is a white space, present between : and RSA.

4.3.3.3 Server 1: Steps to Run IPsec

Follow the step 1 to 10 of L2GRE Server 1 Configuration to setup L2GRE on server 1.

Then enable IPSec as below.

- Step 1. Enable IPSec offload in the driver:

  ```
  echo 1 >> /sys/class/net/<PF0_interface>/device/ipsec_enable
  ```

  For example:

  ```
  echo 1 >> /sys/class/net/u25eth0/device/ipsec_enable
  ```

- Step 2. Start IPsec:

  ```
  ipsec restart
  ```

4.3.3.4 Server 2: Steps to Run IPsec

Follow the step 1 to 10 of L2GRE Server 2 Configuration to setup L2GRE on server 1.

Then enable IPSec as below.

- Step 1. Enable IPSec offload in the driver:

  ```
  echo 1 >> /sys/class/net/<PF0_interface>/device/ipsec_enable
  ```

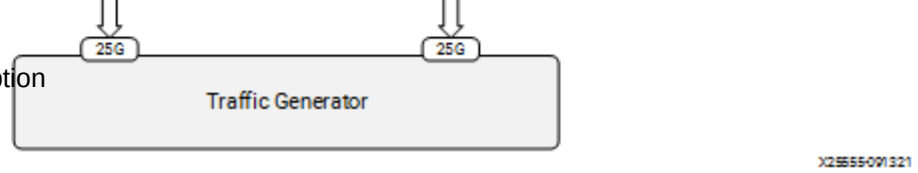
  For example:

  ```
  echo 1 >> /sys/class/net/u25eth0/device/ipsec_enable
  ```

- Step 2. Start IPsec:

  ```
  ipsec restart
  ```

Figure 13: **IPsec + OVS End to End setup Diagram**



4.4 Stateless Firewall

This section is about stateless firewall prerequisites, installation steps, and supported rules.

Kernel Upgrade

Run the command `uname -r` to get the kernel version. If the kernel version is v5.5 or higher, skip the following steps:

1. Install kernel > 5.5 for nftables offload support

```
install linux-image-<version>-generic linux-headers-<version>-generic  linux-modules-extra-<version
```

- Step 2. After the command is executed with no error, do a reboot:

  ```
  reboot
  ```

nftables

***Note*:** Refer Deployment Image Flashing for flashing images to check firewall functionality.

- Step 1. Refer to Basic Requirements and Component Versions Supported for the required OS/software version.
- Step 2. Check the nftables version. U25N supports nftables version ≥ v0.9.6. The nftables version can be found using the command `nft -v`.

  ***Note*:** Tested version 0.9.6 and 0.9.8.
- Step 3. The nftables only work in legacy mode. Ensure the U25N is in legacy mode using the following command:

  ```
  devlink dev eswitch show pci/0000:<pci_id>
  ```

  The output of the above command should be:

  ```
  pci/0000:<pci_id>: mode legacy
  ```

  In case not in legacy mode, change it to legacy mode using the following command:

  ```
  devlink dev eswitch set pci/0000:<pci_id> mode legacy
  ```

4.4.3 Classification Fields (Matches)

Maximum rules supported: 1K for each U25N PF interface.

```
Keys:

1. IPv4/IPv6 source address

2. IPv4/IPv6 destination address

3. Protocol (TCP/UDP)

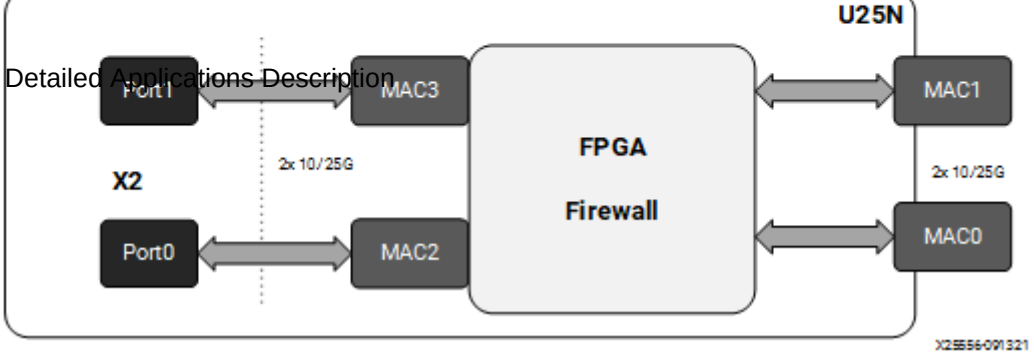
4. Src_port

5. Dst_port

6. Chain

7. Interface Action

Actions
```



Driver Installation

***Note*:** Log in as a root user before proceeding with the following steps.

The prerequisites for the driver installation are as follows:

- modprobe mtd (first time only)
- modprobe mdio (first time only)
- Step 1. Check the driver version of U25N interface using the following command:

  ***Note*:** Ignore this step, if the U25N driver version is latest.

  ```
  ethtool -i u25eth0 | grep version
  ```

  ***Note*:** To install the latest sfc driver, refer to U25N Driver.
- Step 2. Creating a table

  ```
  nft add table <family> <name>
  ```

  For example: `nft add table netdev filter`.

ep 3. Creating a chain:

r example:

```
ft add chain netdev filter input1 { type filter hook ingress device ens7f1np1 priority 1; flags off
```

ding a chain without specifying the policy leads to the default policy Accept.

- Step 4. Adding rules to the chain:

  ```
  nft add rule <family> <table name> <chain name> ip saddr <ip> drop
  ```

  For example:

  ```
  nft add rule netdev filter input1 ip saddr 1.1.1.1 drop
  ```
- Step 5. Commands for listing tables, chain, and rules:

  Listing a table of a netdev family:

  ```
  nft list tables <family>
  ```

  For example:

  ```
  nft list tables netdev
  ```

  Listing a particular chain from a table:

  ```
  nft list chain <family> <table name> <chain_name>
  ```

  For example:

  ```
  nft list chain netdev filter input1
  ```

  Listing a chain along with a handle:

  ```
  nft -a list chain <family> <table name> <chain_name>
  ```

For example:

```
nft -a list chain netdev filter input1
```

Listing all tables, chains, and rules with handle:

```
nft -a list ruleset
```

- Step 6. Commands for deleting tables, chains and rules:

  Deleting a table:

  ```
  nft delete table <family> <name>
  ```

  For example:

  ```
  nft delete table netdev filter
  ```

  Deleting a chain:

  ```
  nft delete chain <family> <table name> <chain name>
  ```

  For example:

  ```
  nft delete chain netdev filter input1
  ```

  Deleting a specific rule with a handle:

  ```
  nft delete rule <family> <table name> <chain_name> handle <handle_no>
  ```

  For example:

  ```
  nft delete rule netdev filter input1 handle 3
  ```

  ***Note*:** Here the handle number for a specific rule could be found using the `nft -a list ruleset` command.

4.5 Rate Limiting

Rate limiting can be applied on U25N's virtual function in ingress and egress directions.

4.5.1 Ingress rule:

4.5.1.1 Add Rate Limit Policy:

```
tc filter add dev <VF_rep_interface> parent ffff: protocol all prio 90 matchall action polic
```

Eg: Ingress rate limit of 4 Gbps at u25eth0_0

```
tc filter add dev u25eth0_0 parent ffff: protocol all prio 90 matchall action police rate 40
```

4.5.1.2 Delete Rate Limit Policy

```
tc filter del dev <VF_rep_interface> parent ffff: protocol all prio 90 matchall action polic
```

Eg:

```
tc filter del dev u25eth0_0 parent ffff: protocol all prio 90 matchall action police rate 40
```

4.5.2 Egress rule:

4.5.2.1 Add Rate Limit Policy:

```
tc qdisc add dev <VF_rep_interface> handle 1: root prio
tc filter add dev <VF_rep_interface> parent 1: protocol all prio 100 matchall action police
```

Eg: Egress rate limit of 2 Gbps at u25eth0_0

```
tc qdisc add dev u25eth0_0 handle 1: root prio
tc filter add dev u25eth0_0 parent 1: protocol all prio 100 matchall action police rate 2000
```

4.5.2.2 Delete Rate Limit Policy

```
tc filter del dev <VF_rep_interface> parent 1: protocol all prio 100 matchall action police
tc qdisc del dev <VF_rep_interface> handle 1: root prio
```

Eg:

```
tc qdisc del dev u25eth0_0 handle 1: root prio
tc filter del dev u25eth0_0 parent 1: protocol all prio 100 matchall action police rate 2000
```

4.6 Mirroring :

4.6.1 Local Mirroring

Local mirroring feature mirrors the traffic to VF interface on local server.

4.6.1.1 Add Local Ingress Mirroring Rule:

```
tc filter add dev <VF_rep_interface_0> parent ffff: matchall skip_sw action mirred ingress m
```

***Note**:* Here <VF_rep_interface_0> is the interface from which traffic will be mirrored and <VF_rep_interface_2> is the interface to which mirrored traffic will be sent.

Eg:

```
tc filter add dev u25eth0_0 parent ffff: matchall skip_sw action mirred ingress mirror dev u
```

4.6.1.2 Delete Local Ingress Mirroring Rule:

```
tc -f filter show dev <VF_rep_interface> parent ffff:
tc filter del dev <VF_rep_interface> parent ffff: handle 1 prio 49152 protocol all matchall
```

Eg:

```
tc -f filter show dev u25eth0_0 parent ffff:
tc filter del dev u25eth0_0 parent ffff: handle 1 prio 49152 protocol all matchall
```

4.6.1.3 Add Local Egress Mirroring Rule:

```
tc qdisc add dev <VF_rep_interface_0> handle 1: root prio
tc filter add dev <VF_rep_interface_0> parent 1: matchall skip_sw action mirred egress mirro
```

Eg:

```
tc qdisc add dev u25eth0_0 handle 1: root prio
tc filter add dev u25eth0_0 parent 1: matchall skip_sw action mirred egress mirror dev u25et
```

4.6.1.4 Delete Local Egress Mirroring Rule:

```
tc qdisc del dev <VF_rep_interface>
tc qdisc del dev <VF_rep_interface> root
```

Eg:

```
tc qdisc del dev u25eth0_0
tc qdisc del dev u25eth0_0 root
```

4.6.2 Remote Mirroring

4.6.2.1 Setup VxLAN tunnel between local and remote servers

On local server, create VxLAn tunnel to connect to remote server

```
ip addr add <TUNNEL_LOCAL_IP>/24 dev <VF_interface>
ip link set dev <VF_interface> mtu 9000
ip link set dev <VF_interface> up
ip link add vxlan0 type vxlan id 100 remote <TUNNEL_REMOTE_IP> dev <VF_interface> dstport 47
```

Eg: u25eth0n0 is the VF interface. It connects to remote server through the VxLAN tunnel for mirrored traffic.

```
ip addr add 10.0.0.2/24 dev u25eth0n0
ip link set dev u25eth0n0 mtu 9000
ip link set dev u25eth0n0 up
ip link add vxlan0 type vxlan id 100 remote 10.0.0.3 dev u25eth0n0 dstport 4789 ifconfig vxl
```

4.6.2.2 Add Remote Ingress Mirroring Rule:

```
tc filter add dev <VF_rep_interface> parent ffff: matchall skip_sw action tunnel_key set src
```

Eg:

```
tc filter add dev u25eth0_0 parent ffff: matchall skip_sw action tunnel_key set src_ip 10.0.
```

4.6.2.3 Delete Remote Ingress Mirroring Rule:

```
tc filter del dev <VF_rep_interface> parent ffff:
```

Eg:

```
tc filter del dev u25eth0_0 parent ffff:
```

4.6.2.4 Add Remote Egress Mirroring Rule:

```
tc qdisc add dev <VF_rep_interface> handle 1: root prio
tc filter add dev <VF_rep_interface> parent 1: matchall skip_sw action tunnel_key set src_ip
```

Eg:

```
tc qdisc add dev u25eth0_0 handle 1: root prio
tc filter add dev u25eth0_0 parent 1: matchall skip_sw action tunnel_key set src_ip 10.0.0.2
```

4.6.2.5 Delete Remote Egress Mirroring Rule:

```
tc filter del dev <VF_rep_interface> parent ffff:
tc qdisc del dev <<VF_rep_interface>> root
```

Eg:

```
tc filter del dev u25eth0_0 parent ffff:
tc qdisc del dev u25eth0_0 root
```

4.7 Statistics

This section outlines the commands used by different modules to check the statistics and packet counters.

4.7.1 OVS Commands

To print a brief overview of the database contents:

```
ovs-vsctl show
```

To show the datapath flow entries:

```
ovs-ofctl dump-flows <bridge_name>
```

To show the full OpenFlow flow table, including hidden flows, on the bridge:

```
ovs-appctl dpctl/dump-flows type=offloaded
```

To show the OVS datapath flows:

```
ovs-dpctl dump-flows
ovs-appctl dpctl/dump-flows
```

To show which flows are offloaded or not:

```
tc filter show dev <iface_name> ingress
```

4.7.2 MAE Rules

Use the following command to display the rules present in the match-action engine (MAE):

```
cat /sys/kernel/debug/sfc/<if_iface>/mae_rules
```

***Note*:** Here `if_iface` should be the corresponding PF interface.

For example:

```
cat /sys/kernel/debug/sfc/if_u25eth0/mae_rules
```

Use the following command to display the default rules present in the MAE:

```
cat /sys/kernel/debug/sfc/<iface/mae_default_rules
```

4.7.3 IPsec Statistics

Get IPsec stats:

```
swanctl --stats
```

Use the `ip xfrm show` command to display IPsec offload security association.

4.8 Debug Commands

***Note*:** The output of the following commands should be saved for debug purposes.

- `lsmod` - It displays which kernel modules are currently loaded. Whether the sfc driver is inserted or not can be verified by the below command.

  ```
  lsmod | grep sfc
  ```

  Expected result:

  ```
  sfc                           704512  0
  sfc_driverlink                16384   1  sfc
  virtual_bus                   20480   1  sfc
  mtd                           65536   14 cmdlinepart,sfc
  mdio                          16384   1  sfc
  ```

- `dmesg` - To get kernel logs, enter the following command. This command will help to understand all the actions performed by the driver and if there are any crashes happening due to the driver.
- `lspci` - To display information about all PCI buses and devices in the system. It will also show the network cards inserted in the system with details like driver in use, pci id etc. For Example:

```
grep Solarflare #Lists the info regarding solarflare devices in the system
k | grep Solarflare #Lists the subsystem information also.
vvv -s <BDF>
```

esult:

```
grep Solarflare:
Ethernet controller: Solarflare Communications XtremeScale SFC9250 10/25/40/50/100G Ethernet Contro
Ethernet controller: Solarflare Communications XtremeScale SFC9250 10/25/40/50/100G Ethernet Contro

k | grep Solarflare:
Ethernet controller: Solarflare Communications XtremeScale SFC9250 10/25/40/50/100G Ethernet Contro
em: Solarflare Communications XtremeScale SFC9250 10/25/40/50/100G Ethernet Controller
Ethernet controller: Solarflare Communications XtremeScale SFC9250 10/25/40/50/100G Ethernet Contro
em: Solarflare Communications XtremeScale SFC9250 10/25/40/50/100G Ethernet Controller
```

- Logs generated by U25N hardware are saved to a file.The logs are collected from the internal processing subsystem and exported to the host at frequent intervals. Currently these logs are populated when switchdev mode is enabled.

  Path to read the logs in host : `/var/log/ps_dmesg.txt`

- `sfreport` - A command line utility that generates a diagnostic log file providing diagnostic data about the server and Solarflare adapters. Please refer to SF-103837-CD Solarflare Server Adapter User Guide chapter 5.20 for more details.

- `top` - This command can be used to show the linux processes or threads. It provides a real time view of the running system. It can be used to detect memory leaks. The file in the linux path /proc/meminfo can also be used for detecting memory leaks.

  Watch out for the total memory already in use. Memory leak can be identified by running the command multiple times and checking whether the memory usage keeps on increasing.

- `ps` - This command displays relevant information about active processes.

  Example:

  ```
  ps -aux | grep sfc
  ```

- `ethtool` - This command can be used to understand the driver related information such as version, firmware info and the enabled features.

  ```
  ethtool -i <interface_name>
  ```

  Example Usage: ethtool -i u25eth0

  ```
  Expected result:
  driver: sfc
  version: 5.3.3.2000.1
  firmware-version: 7.8.7.1005 rx0 tx0
  expansion-rom-version:
  bus-info: 0000:3b:00.0
  supports-statistics: yes
  supports-test: yes
  supports-eeprom-access: no
  supports-register-dump: yes
  supports-priv-flags: yes
  ```

  To get the information of the state of protocol offload and other features

  ```
  ethtool -k <interface_name>
  ```

  Example Usage: ethtool -k u25eth0

  ```
  Expected result:
  Features for enp59s0f1np1:
  rx-checksumming: on
  ```

```
tx-checksumming: on
   tx-checksum-ipv4: on
   tx-checksum-ip-generic: off [fixed]
   tx-checksum-ipv6: on
   tx-checksum-fcoe-crc: off [fixed]
   tx-checksum-sctp: off [fixed]
scatter-gather: on
   tx-scatter-gather: on
   tx-scatter-gather-fraglist: off [fixed]
tcp-segmentation-offload: on
   tx-tcp-segmentation: on
   tx-tcp-ecn-segmentation: on
   tx-tcp-mangleid-segmentation: off
   tx-tcp6-segmentation: on
generic-segmentation-offload: on
generic-receive-offload: on
large-receive-offload: off
```

To change the offload parameters and other features of the network device

```
ethtool -K <interface_name> <feature> <on/off>
```

To get information about NIC Statistics

```
ethtool -S <interface_name>
```

Example usage: ethtool -S enp59s0f1np1

```
```bash
Expected result:
NIC statistics:
   rx_noskb_drops: 0
   rx_nodesc_trunc: 0
   port_tx_bytes: 59052
   port_tx_packets: 353
   port_tx_pause: 0
   port_tx_control: 0
   port_tx_unicast: 0
   port_tx_multicast: 273
   port_tx_broadcast: 80
   port_tx_lt64: 0
   port_tx_64: 0
   port_tx_65_to_127: 229
   port_tx_128_to_255: 44
   port_tx_256_to_511: 80
   port_tx_512_to_1023: 0
   port_tx_1024_to_15xx: 0
   port_tx_15xx_to_jumbo: 0
   port_rx_bytes: 0
   port_rx_good_bytes: 0
   port_rx_bad_bytes: 0
   port_rx_packets: 0
```
```

***NOTE*:** ethtool functionalities for sfc driver can also be realised using the sfctool utility also. For example:

```
sfctool -S <interface_name>
```

Example Usage:

```
sfctool -S u25eth0
```

- `u25n_update` Application: u25n_update utility can be used to read the U25N shell version. For example:

  ```
  ./utils/u25n_update get-version <PF0_interface>
  ```

- MCDI Logging - Mcdi request and response data will be visible in dmesg if we activate mcdi logs. To activate the logs in dmesg.

  ```
  echo 1 >> /sys/class/net/<interface_name>/device/mcdi_logging
  ```

- OVS log levels - It can be turned on/off using the ovs-appctl commands.

  The ovs-vswitchd accepts the option –log-file[=file] to enable logging to a specific file. The file argument is actually optional, so if it is specified, it is used as the exact name for the log file. The default is used if the file is not specified. Usually the default is /usr/local/var/log/openvswitch/ovs-vswitchd.log

  Setting OVS Log levels:

  ```
  ovs-appctl vlog/set ANY:ANY:dbg
  ovs-appctl vlog/set poll_loop:ANY:OFF
  ovs-appctl vlog/set netlink_socket:ANY:OFF
  ```

- MAE Rules - To see the offloaded rules available in the MAE, check the below file:

  ```
  cat /sys/kernel/debug/sfc/if_<interface_name>/mae_rules
  ```

  To check the default rules in MAE, check the below file:

  ```
  cat /sys/kernel/debug/sfc/if_<interface_name>/mae_default_rules
  ```

- `iperf3` - Iperf is a tool for network performance measurement and tuning. It is a cross-platform tool that can produce standardized performance measurements for any network. It has client and server functionality, and can create data streams to measure the throughput between the interfaces. After setting proper ip addresses:

  Server:

  ```
  iperf3 -s <options>
  ```

  Client :

  ```
  iperf -c <ip_addr_interface> <options>
  ```

- `tcpdump` - Tcpdump is a packet sniffing and packet analyzing tool meant for System Administrators to troubleshoot connectivity issues in Linux. For example it can capture the packets coming to the network interface using the below command.

  ```
  tcpdump -i <interface_name>
  ```

- `conntrack` - get connection status and track connection process

  Get connection status

  ```
  conntrack -L
  ```

  Track connection process

  ```
  conntrack -E
  ```

  Check U25N offloaded connection

  ```
  watch -n 10 "cat /sys/kernel/debug/sfc/if_<PF_interface>/tracked_conns | grep 0xfff | wc -l"
  ```

DPDK

DPDK compilation steps

Download the dpdk git repositories from http://www.dpdk.org/browse.

Extract the downloaded dpdk files and download the required packages:

```
apt-get install libnuma-dev liblua5.3-dev libpcap-dev [for ubuntu]
```

Create the DPDK build tree:

Follow the steps mentioned at Compilation Steps to do DPDK compilation.

oc.dpdk.org/guides/linux_gsg/linux_drivers.html.

```
io driver, the path for dpdk-kmod can be found at [http://git.dpdk.org/dpdk-kmods](http://git.dpdk.o
```

river:

```
tory]
```

Command to run testpmd:

Refer to https://doc.dpdk.org/guides/testpmd_app_ug/run_app.html for more information about the testpmd runtime command. The following is an example command to run testpmd:

***Note*:** Make sure the dpdk-testpmd for specific PCI devices is running on the same NUMA node.

Use the following command to get the numa node for a specific PCI device.

```
cat /sys/bus/pci/devices/0000\:<pci device id>/numa_node
```

The cores for the specific numa node could be found using the following command:

```
lscpu | grep NUMA
```

For example, `lscpu | grep NUMA`.

```
NUMA node(s): 2
NUMA node0 CPU(s): 0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30
NUMA node1 CPU(s): 1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31
```

APPENDIX A U25N Shell Programming

To program the U25N flash, an Alveo™ programming cable is required. Refer to *Alveo Programming Cable User Guide* (UG1377) to connect the Alveo programming cable to the U25N maintenance connector. The Vivado® tools must be installed on the server to which the Alveo programming cable's USB port is connected. For more information, see Vivado ML Overview.

Entering into U-Boot Mode

***Note*:** There is no need to performing the first step when flashing the shell image for the first time.

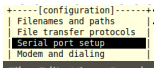
Connect a USB cable to the SmartNIC.

a. Open a command line terminal and run the minicom command with sudo.

b. Before running minicom, verify the serial port setup configuration using the following steps:

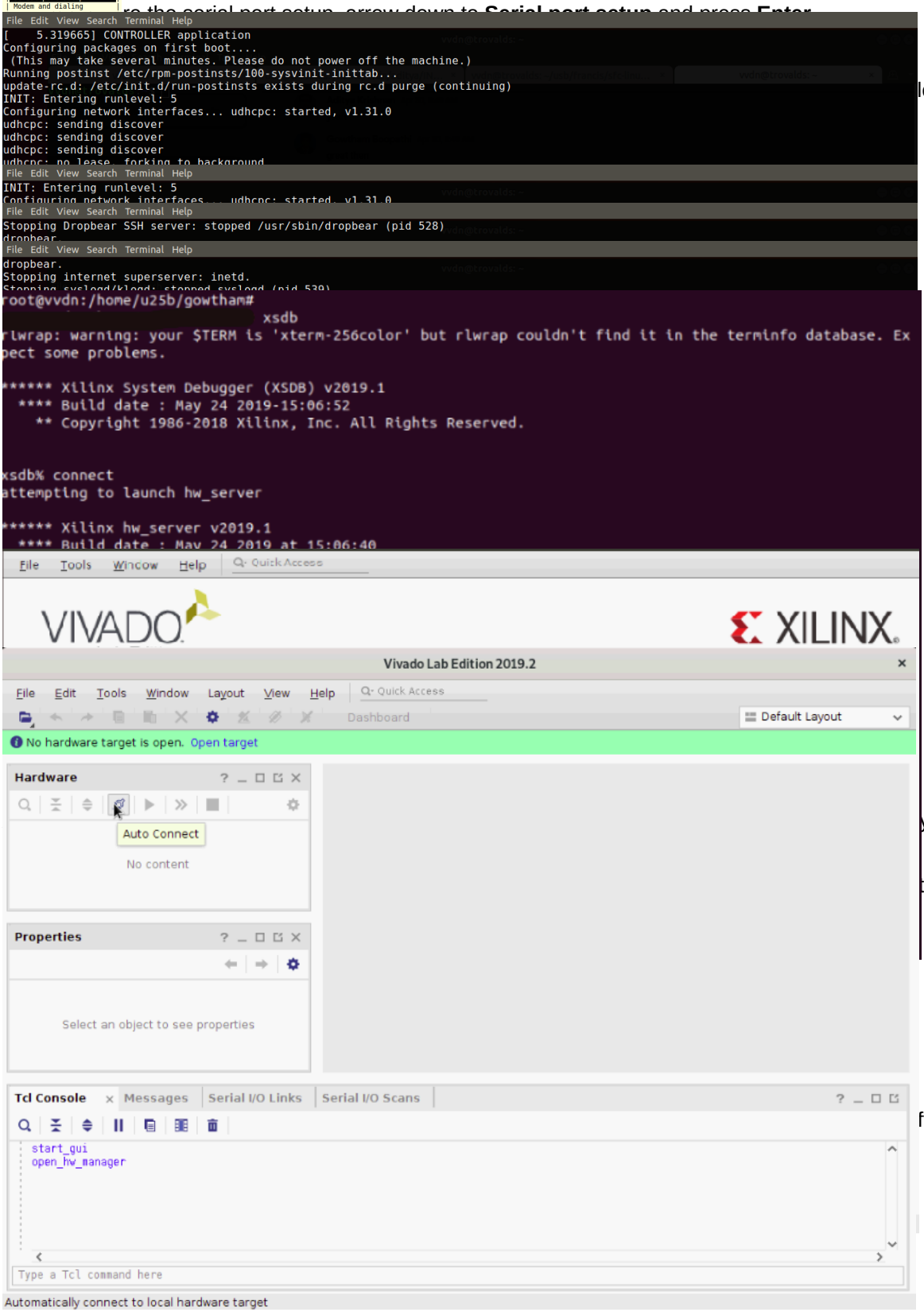
i. Pull up the settings using the -s option.

```
sudo minicom -s
```



oring up a colorful display listing the different settings.

re the serial port setup, arrow down to **Serial port setup** and press **Enter**.



le, press **A** to

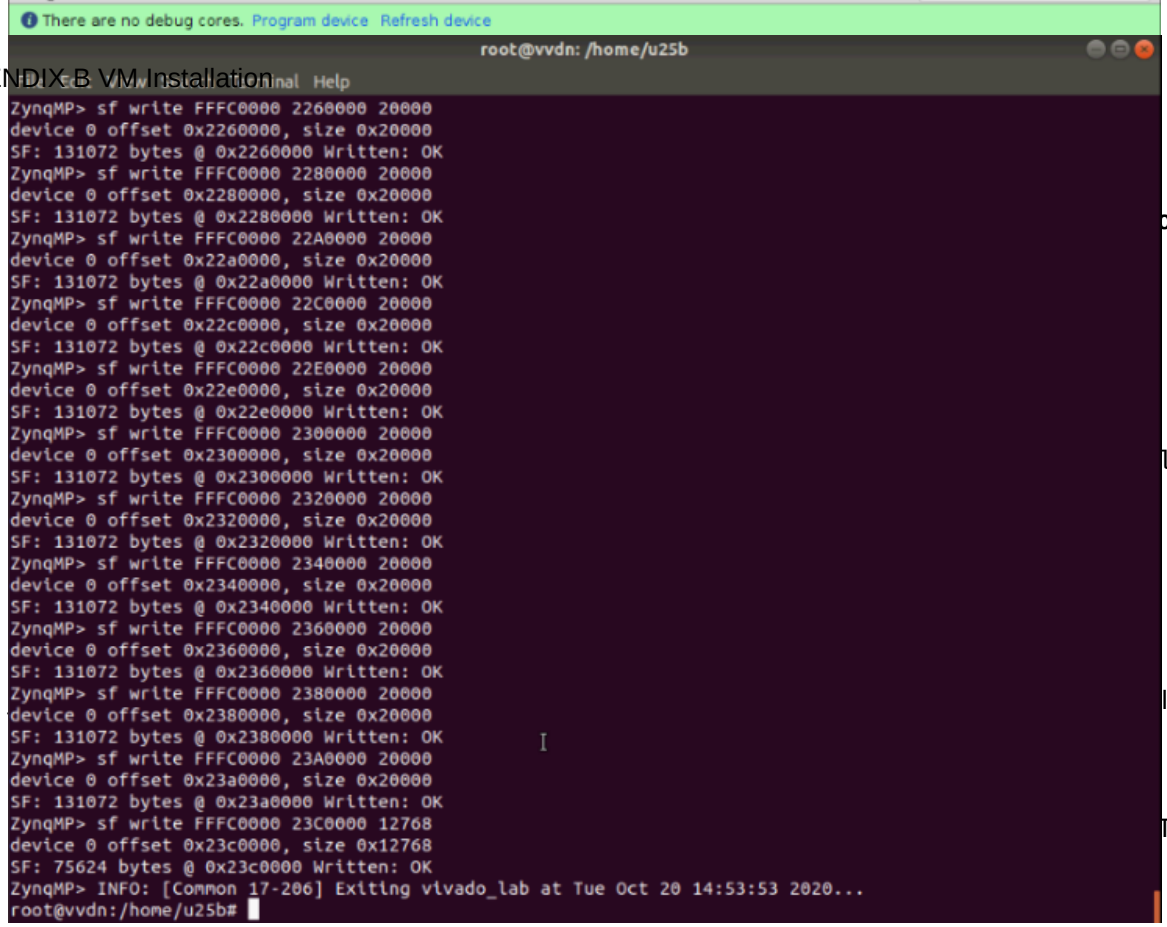
command as

y in the

t you get core

ftware opens,

c. Verify whether the U25N SmartNIC is detected properly.



ce.

lf).

lowing output

These can be

Power cycle the server (Power OFF and Power ON).

APPENDIX B VM Installation

***Note*:** This steps mentioned in this section needs to be performed only for VM cases. The following configuration steps are for Ubuntu OS.

Installation of libraries and dependencies is required for running VMs

Install qemu (IN HOST):

```
sudo apt-get install qemu-system-x86
```

Update the following command in the path `/etc/sysctl.conf` for huge page configuration:

```
sudo vim /etc/sysctl.conf
kernel.shmmax = 25769803776 [ it allocates 24G 1G hugepages ]
vm.hugetlb_shm_group = 0
```

Update Host Grub with the following command:

```
sudo vim /etc/default/grub
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash hugepagesz=1GB hugepages=24
default_hugepagesz=1GB iommu=pt intel_iommu=on pci=realloc"
```

After doing changes in Host Grub, update the Grub using the following command:

```
sudo update-grub
sudo reboot
```

Allocate the hugepage of size 8 Gb for a VM using this command:

```
mkdir /mnt/huge_vm
mount -t hugetlbfs -o size=8G none /mnt/huge_vm
```

Insert the VFIO driver using this command:

```
sudo modprobe vfio-pci
```

***Note*:** Make sure the next step is done only after the VF representor interfaces are added to the OVS bridge.

Unbind the VF PCIe® id from sfc before binding to the vfio-pci driver.

***Note*:** The VF PCIe device ID can be listed with the `lspci -d 1924:1b03` command. An example of the device ID is *af:00.2 Ethernet controller: Solarflare Communications XtremeScale SFC9250 10/25/40/50/100G Ethernet Controller (Virtual Function) (rev 01)*.

```
echo '0000:<VF PCIe id>' > /sys/bus/pci/drivers/sfc/unbind
```

For example:

```
echo '0000:af:00.2' > /sys/bus/pci/drivers/sfc/unbind
```

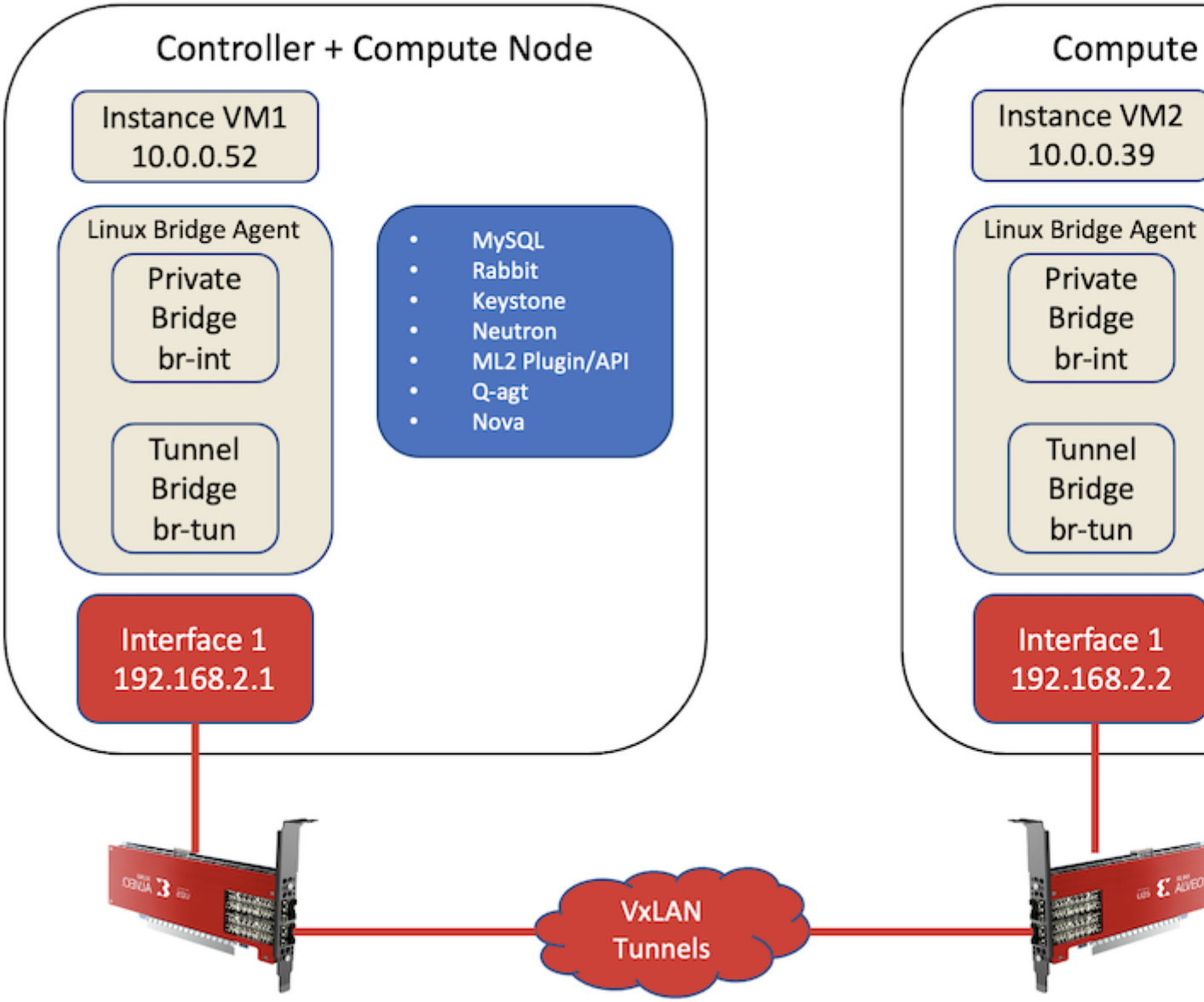
Bind the Vendor ID and device ID of the U25N X2 to VFIO driver:

```
echo "1924 1b03" > /sys/bus/pci/drivers/vfio-pci/new_id
```

Bind the respective PCIe device ID to the driver:

```
echo '0000:<VF PCIe id>' > /sys/bus/pci/drivers/vfio-pci/bind
```

For example:



Please refer to U25N Installation Guide for U25N and server configuration.

2. Software Requirements

2.1 OpenStack Releases

OpenStack has multiple releases from Austin to Zed. In this document, version Zed is used as an example and tested.

2.2 OS versions

Ubuntu 22.04 is used as an example and tested in this installation guide.

Below Linux kernel versions are tested.

- 5.15.0-52-generic
- 5.15.0-46-generic

2.3 OpenStack Deployment Tool

Due to the complexity of OpenStack installation, OpenStack deployment tools document lists some deployment tools. Some OS distributions also have their own deployment tools. For example, Ubuntu has Microstack.

In this section, DevStack is used for the OpenStack deployment. All the configurations for using U25N in OpenStack environment are through standard Nova, Neutron config files. After installation complete, the same configuration files, e.g. /etc/neutron/neutron.conf, /etc/neutron/plugins/ml2/ml2_conf.ini, /etc/nova/nova.conf, /etc/nova/nova-cpu.conf, etc will be used for the following configurations.

3. OpenStack Installation

3.1 Grub Configuration

***Note*:** Grub configuration should be done on both Controller + Compute node and Compute node

To enable the U25N VF passthrough into the VM instance, it is required to enable IOMMU.

Open /etc/default/grub in the editor

Modify the line of GRUB_CMDLINE_LINUX_DEFAULT to enable IOMMU

```
# For Intel platform:
GRUB_CMDLINE_LINUX_DEFAULT="iommu=pt intel_iommu=on pci=realloc”
# For AMD platform:
GRUB_CMDLINE_LINUX_DEFAULT="iommu=pt amd_iommu=on pci=realloc”
```

Update Grub

```
$ sudo update-grub2
```

3.2 Add Stack User (optional)

***Note*:** Adding stack user should be done on both Controller + Compute node and Compute node

Below setup uses non-root user **stack** with sudo enabled. The password is set to “nomoresecret” as shown in the following DevStack configuration file.

***Note*:** Please refer to DevStack Documentation for more details.

```
$ sudo useradd -s /bin/bash -d /opt/stack -m stack
$ sudo chmod +x /opt/stack
$ echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack
$ sudo -u stack -i
$ passwd
Enter new UNIX password: nomoresecret
Retype new UNIX password: nomoresecret
passwd: password updated successfully
```

3.3 Download DevStack

***Note*:** DevStack should be downloaded and installed on both Controller + Compute node and Compute node

Download DevStack release zed from Git repository.

```
git clone https://opendev.org/openstack/devstack -b stable/zed
```

3.4 Update DevStack Configuration

***Note*:** DevStack configuration should be modified on both Controller + Compute node and Compute node

Copy devstack/sample/local.conf to devstack/ and then add below lines

For server 1 (Controller + Compute Node),

- Modify below lines in the local.conf

```
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
```

- Add below lines in the local.conf

```
disable_service n-net
enable_service q-svc
enable_service q-agt
enable_service q-dhcp
enable_service q-meta
enable_service neutron

HOST_IP=<server 1's IP> # The IP you ssh to this node
PUBLIC_INTERFACE=u25eth0
PUBLIC_NETWORK_GATEWAY=192.168.2.1 # The network u25N attached
FLOATING_RANGE=192.168.2.0/24
Q_FLOATING_ALLOCATION_POOL=start=192.168.2.2,end=192.168.2.254

Q_USE_PROVIDERNET_FOR_PUBLIC=True
PUBLIC_PHYSICAL_NETWORK=public
OVS_BRIDGE_MAPPINGS=public:br-ex
OVS_ENABLE_TUNNELING=True

Q_PLUGIN=ml2
Q_AGENT=openvswitch
Q_L3_ROUTER_PER_TENANT=True
Q_ML2_TENANT_NETWORK_TYPE=vxlan
Q_ML2_PLUGIN_TYPE_DRIVERS=flat,vxlan
Q_ML2_PLUGIN_VXLAN_TYPE_OPTIONS=(vni_ranges=1001:2000)
Q_ML2_PLUGIN_MECHANISM_DRIVERS=openvswitch,l2population

IP_VERSION=4
```

For server 2 (Compute Node),

- Modify below lines in the local.conf

```
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
```

- Add below lines in the local.conf

```
RABBIT_PASSWORD=$ADMIN_PASSWORD

HOST_IP=<server 2's IP> # The IP you ssh to this node
SERVICE_HOST=<server 1's IP>

MYSQL_HOST=$SERVICE_HOST
```

```
RABBIT_HOST=$SERVICE_HOST
GLANCE_HOSTPORT=$SERVICE_HOST:9292
KEYSTONE_AUTH_HOST=$SERVICE_HOST
KEYSTONE_SERVICE_HOST=$SERVICE_HOST

PHYSICAL_NETWORK=default
OVS_PHYSICAL_BRIDGE=br-ex
PUBLIC_INTERFACE=u25eth0
Q_USE_PROVIDER_NETWORKING=True
PUBLIC_NETWORK_GATEWAY=192.168.2.2
Q_AGENT=openvswitch
NEUTRON_AGENT=$Q_AGENT
Q_ML2_PLUGIN_MECHANISM_DRIVERS=openvswitch,l2population
Q_ML2_TENANT_NETWORK_TYPE=vxlan
Q_ML2_PLUGIN_TYPE_DRIVERS=flat,vxlan
Q_ML2_PLUGIN_VXLAN_TYPE_OPTIONS=(vni_ranges=1001:2000)
ENABLED_SERVICES=n-cpu,rabbit,q-agt,placement-client,n-novnc
```

Copy devstack/sample/local.sh to devstack/ and comment out below lines

```
#    openstack security group rule create $default --protocol tcp --dst-port 22
#    openstack security group rule create $default --protocol icmp
```

Comment below lines in devstack/lib/neturon-legacy

```
#    ARP_CMD="sudo arping -A -c 3 -w 5 -I $to_intf $IP "
```

Assign tunnel Endpoint IP to U25N PF interface

***Note*:** Here using U25N's PF0 as a public interface.

- For server 1 (Controller + Compute Node), the TUNNEL_LOCAL_IP is 192.168.2.1 as configured in above local.conf.
- For server 2 (Compute Node), the TUNNEL_LOCAL_IP is 192.168.2.2 as configured in above local.conf.

```
$ ip addr add ${TUNNEL_LOCAL_IP}/24 dev u25eth0
$ ip link set dev u25eth0 up
```

Then, run stack.sh to install DevStack.

```
./stack.sh
```

3.5 U25N Open vSwitch Hardware Offload

***Note*:** U25N offload should be configured on both Controller + Compute node and Compute node

Before configuring OpenStack Nova and Neutron configuration files, Virtual Functions of U25N need to be created.

- Download OVS bit file on U25N Please refer to U25N Installation Guide for how to install the U25N driver, flash deployment image, load OVS partial Bitstream.

  Program the OVS bitstream.

```
# Set U25N in legacy mode
devlink dev eswitch set pci/${PCIE_BUS} mode legacy

# Make sure the network ports are up
ip link set dev u25eth0 up
ip link set dev u25eth1 up

# Program the deployment shell
U25N_Release/utils/u25n_update upgrade U25N_Release/bits/ovs.bit u25eth0

sleep 5

# Reset hardware to boot from updated deployment shell
```

```
U25N_Release/utils/u25n_update reset u25eth0

# Get deployment shell version
U25N_Release/utils/u25n_update get-version u25eth0
```

- Stop all service of OpenStack

  ***Note*:** This step is required only afer power cycle or reboot.

  ```
  $ systemctl stop devstack@*
  ```

- Stop Network Manager

  ```
  $ systemctl stop NetworkManager
  ```

- Enable U25N's Virtual Function, for this setup, NUM_VF is 1.

  ```
  # Enable 1 VFs on PF 1, they could be used for VM-to-port or VM-to-VM use cases.
  echo ${NUM_VF} > /sys/class/net/u25eth0/device/sriov_numvfs

  # Make sure the network ports are up
  ip link set dev u25eth0 up
  ip link set dev u25eth1 up

  # Set both ports in switchdev for OVS
  sleep 5
  devlink dev eswitch set pci/0000:${PCIE_BUS}:00.0 mode switchdev
  sleep 5
  devlink dev eswitch set pci/0000:${PCIE_BUS}:00.1 mode switchdev
  sleep 1
  ```

- Configure Open vSwitch hardware offload

  ```
  # Export paths
  export PATH=$PATH:/usr/local/share/openvswitch/scripts
  export PATH=$PATH:/usr/local/bin

  # Enable VFs
  ip link set dev ${VF_0} up

  # Enable VF Representors
  ip link set dev ${VF_RP_0} up

  # Enable hardware offload
  ovs-vsctl set Open_vSwitch . other_config:hw-offload=true
  ovs-vsctl set Open_vSwitch . other_config:tc-policy=none
  ```

- Modify local IP in /etc/neutron/plugin/ml2/ml2_conf.ini on each nodes

  ```
  # Server 1 (Controller + Compute Node)
  local_ip = 192.168.2.1
  # Server 2 (Compute Node)
  local_ip = 192.168.2.2
  ```

- Update /etc/nova/nova.conf on Controller node

  ```
  [filter_scheduler]
  enabled_filters = PciPassthroughFilter
  available_filters = nova.scheduler.filters.all_filters
  ```

- Update /etc/nova/nova.conf on Compute nodes.

  ***Note*:** If using single server for both Controller and Compute nodes, this config file is /etc/nova/nova-cpu.conf.

  ***Note*:** Below af:00 is the Buss:Device of the U25N.

  ```
  [pci]
  passthrough_whitelist = {"address":":af:00.", "physical_network":null}
  ```

- Restart Open vSwitch service

```
$ systemctl restart openvswitch-switch.service
```

- Start all OpenStack services

```
$ systemctl start devstack@* --all
$ systemctl status devstack@*  # Check the status of services
```

- Dump the OVS bridge information on both nodes and remove unused interface in br-ex

```
$ ovs-vsctl show
2f120f0a-fa48-4779-8e75-4915cf4044cc
Manager "ptcp:6640:127.0.0.1"
    is_connected: true
Bridge br-ex
    Controller "tcp:127.0.0.1:6633"
        is_connected: true
    fail_mode: secure
    datapath_type: system
    Port phy-br-ex
        Interface phy-br-ex
            type: patch
            options: {peer=int-br-ex}
    Port br-ex
        Interface br-ex
            type: internal
Bridge br-int
    Controller "tcp:127.0.0.1:6633"
        is_connected: true
    fail_mode: secure
    datapath_type: system
    Port int-br-ex
        Interface int-br-ex
            type: patch
            options: {peer=phy-br-ex}
    Port br-int
        Interface br-int
            type: internal
    Port u25eth0_1
        tag: 2
        Interface u25eth0_1
    Port patch-tun
        Interface patch-tun
            type: patch
            options: {peer=patch-int}
Bridge br-tun
    Controller "tcp:127.0.0.1:6633"
        is_connected: true
    fail_mode: secure
    datapath_type: system
    Port br-tun
        Interface br-tun
            type: internal
    Port vxlan-c0a80201
        Interface vxlan-c0a80201
            type: vxlan
            options: {df_default="true", in_key=flow, local_ip="192.168.2.2", out_key=fl
    Port patch-int
        Interface patch-int
            type: patch
```

```
            options: {peer=patch-tun}
    ovs_version: "2.17.3"
```

If u25eth is attached in Bridge br-ex, please remove it by below command.

```
ovs-vsctl del-port br-ex u25eth0
```

3.6 Launch VM instance

***Note*:** Launching VM instannce requires to use openstack command line tool and only need to run it from the Controller + Compute node.

- Source openrc to use OpenStack command line tool on Controller node

  Source openrc in the shell, and then use the openstack command line tool to manage your OpenStack.

  ```bash
  $ cd devstack/
  $ source openrc admin
  ```

ge

tu Cloud image is used as the VM image.

```
sary packeges for VM support. Most of them should already been installed during DevStack installatio
 libvirt-clients qemu-kvm libvirt-daemon-system bridge-utils virtinst cloud-image-utils libguestfs-

host for virtualzation support
date
tus libvirtd

ge and set root password as "root"
ubuntu-20.04 --install "linux-modules-extra-`uname -r`" --format qcow2 --output ubuntu.qcow2 --root-
```

- Add Cloud VM image to OpenStack

  ```
  # Create OpenStack image
  $ openstack image create --disk-format qcow2 --file ubuntu.qcow2 ubuntu
  ```

ect ports for U25N's VF on the private network

```
k port create --network private --vnic-type=direct --binding-profile '{"capabilities": ["switchdev"]
k port create --network private --vnic-type=direct --binding-profile '{"capabilities": ["switchdev"]
```

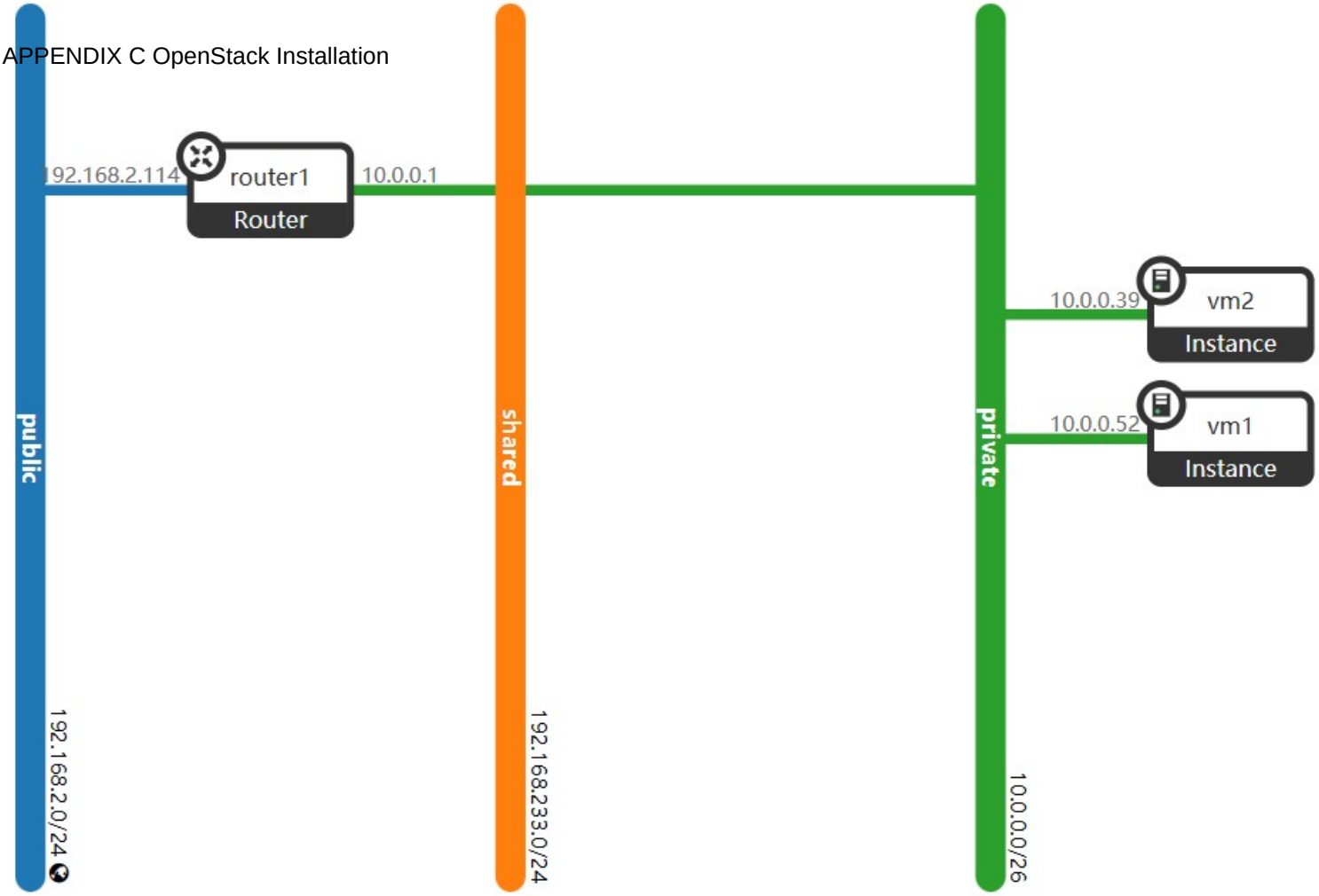
- Create two VM instances with direct port attached

  To make one VM instance per each node, host server is specified in the OpenStack server creation command.

  - Get host name by list available Compute host

    ```
    $ openstack compute service list
    +--------------------------------------+----------------+---------+----------+------
    | ID                                   | Binary         | Host    | Zone     | Statu
    +--------------------------------------+----------------+---------+----------+------
    | 53b216ed-3b68-4187-bc43-5d8156a666e8 | nova-scheduler | server1 | internal | enabl
    | 3e97f027-3320-4a73-aed4-3c08aa06b5f1 | nova-conductor | server1 | internal | enabl
    | 8afa5295-03f0-4de4-9f3f-0c63db719bc1 | nova-conductor | server1 | internal | enabl
    | 21b79147-ef60-4d6a-bd4c-0cc26e1b6420 | nova-compute   | server1 | nova     | enabl
    | 95502539-446e-4d7c-bda5-4c9796ef66d0 | nova-compute   | server2 | nova     | enabl
    +--------------------------------------+----------------+---------+----------+------
    ```

  - Besides the compute server list, check the hypervisor to make sure both server's hypervisor are detected by OpenStack



- Below is the configuration sample dumpped by OpenStack command line tool

```
(openstack) hypervisor list
+----+---------------------------+-----------------+---------------+-------+
| ID | Hypervisor Hostname       | Hypervisor Type | Host IP       | State |
+----+---------------------------+-----------------+---------------+-------+
|  1 | server1                   | QEMU            | <server 1 IP> | up    |
|  2 | server2                   | QEMU            | <server 2 IP> | up    |
+----+---------------------------+-----------------+---------------+-------+

(openstack) compute service list
+--------------------------------------+----------------+---------+----------+---------+
| ID                                   | Binary         | Host    | Zone     | Status  |
+--------------------------------------+----------------+---------+----------+---------+
| 53b216ed-3b68-4187-bc43-5d8156a666e8 | nova-scheduler | server1 | internal | enabled |
| 3e97f027-3320-4a73-aed4-3c08aa06b5f1 | nova-conductor | server1 | internal | enabled |
| 8afa5295-03f0-4de4-9f3f-0c63db719bc1 | nova-conductor | server1 | internal | enabled |
| 21b79147-ef60-4d6a-bd4c-0cc26e1b6420 | nova-compute   | server1 | nova     | enabled |
| 95502539-446e-4d7c-bda5-4c9796ef66d0 | nova-compute   | server2 | nova     | enabled |
+--------------------------------------+----------------+---------+----------+---------+

(openstack) server list
+--------------------------------------+------+--------+-------------------+--------+---
| ID                                   | Name | Status | Networks          | Image  | Fl
+--------------------------------------+------+--------+-------------------+--------+---
| 6b3356f3-5e23-450f-9a6e-14147990b0f2 | vm2  | ACTIVE | private=10.0.0.39 | ubuntu | m1
| 24b8231e-0ae8-4581-a350-76231ba30ea3 | vm1  | ACTIVE | private=10.0.0.52 | ubuntu | m1
+--------------------------------------+------+--------+-------------------+--------+---

(openstack) server show vm1
+-------------------------------------+-----------------------------------------------
| Field                               | Value
```

```
+--------------------------------------+------------------------------------------------
| OS-DCF:diskConfig                    | MANUAL
| OS-EXT-AZ:availability_zone          | nova
| OS-EXT-SRV-ATTR:host                 | server1
| OS-EXT-SRV-ATTR:hypervisor_hostname  | server1
| OS-EXT-SRV-ATTR:instance_name        | instance-00000008
| OS-EXT-STS:power_state               | Running
| OS-EXT-STS:task_state                | None
| OS-EXT-STS:vm_state                  | active
| OS-SRV-USG:launched_at               | 2023-01-11T09:32:25.000000
| OS-SRV-USG:terminated_at             | None
| accessIPv4                           |
| accessIPv6                           |
| addresses                            | private=10.0.0.52
| config_drive                         |
| created                              | 2023-01-11T09:31:53Z
| flavor                               | m1.xlarge (5)
| hostId                               | 87f48315a984ef8f0aad61eaec4435b5179468af66be9e98
| id                                   | 24b8231e-0ae8-4581-a350-76231ba30ea3
| image                                | ubuntu (5b4f747d-f778-430f-93fa-a4fff3f33223)
| key_name                             | None
| name                                 | vm1
| progress                             | 0
| project_id                           | b9d3608c81a74b4a953146a302142c26
| properties                           |
| security_groups                      | name='default'
| status                               | ACTIVE
| updated                              | 2023-01-11T09:32:26Z
| user_id                              | d92a44031cbf4ff7840bd16c04b0c569
| volumes_attached                     |
+--------------------------------------+------------------------------------------------
(openstack) server show vm2
+--------------------------------------+------------------------------------------------
| Field                                | Value
+--------------------------------------+------------------------------------------------
| OS-DCF:diskConfig                    | MANUAL
| OS-EXT-AZ:availability_zone          | nova
| OS-EXT-SRV-ATTR:host                 | server2
| OS-EXT-SRV-ATTR:hypervisor_hostname  | server2
| OS-EXT-SRV-ATTR:instance_name        | instance-0000000b
| OS-EXT-STS:power_state               | Running
| OS-EXT-STS:task_state                | None
| OS-EXT-STS:vm_state                  | active
| OS-SRV-USG:launched_at               | 2023-01-11T10:27:38.000000
| OS-SRV-USG:terminated_at             | None
| accessIPv4                           |
| accessIPv6                           |
| addresses                            | private=10.0.0.39
| config_drive                         |
| created                              | 2023-01-11T10:27:30Z
| flavor                               | m1.large (4)
| hostId                               | 5704a22af1b38b8ed0235f34e26cb050cd5fc0c63ab2964d
| id                                   | 6b3356f3-5e23-450f-9a6e-14147990b0f2
| image                                | ubuntu (5b4f747d-f778-430f-93fa-a4fff3f33223)
| key_name                             | None
| name                                 | vm2
| progress                             | 0
| project_id                           | b9d3608c81a74b4a953146a302142c26
| properties                           |
```

```
| security_groups                     | name='default'
| status                              | ACTIVE
| updated                             | 2023-01-11T10:27:39Z
| user_id                             | d92a44031cbf4ff7840bd16c04b0c569
| volumes_attached                    |
+-------------------------------------+-----------------------------------------

(openstack) network list
+--------------------------------------+---------+--------------------------------------
| ID                                   | Name    | Subnets
+--------------------------------------+---------+--------------------------------------
| 4b8396ce-f932-4f64-95d1-3b1274b606e6 | public  | 6a9ece87-cb33-4e92-b1cc-8335dbdf54c6
| ead66cda-b35a-4af0-bf90-5c3c1fca808f | shared  | 89bf2fcb-dc6b-4b13-811a-ee2a0cb145c5
| eb96370b-4063-4a4a-9151-39c941fffb83 | private | c9b0c2c8-5dd3-440f-b1d9-fed99d18849e
+--------------------------------------+---------+--------------------------------------

(openstack) network show private
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | UP                                   |
| availability_zone_hints   |                                      |
| availability_zones        | nova                                 |
| created_at                | 2023-01-11T05:29:39Z                 |
| description               |                                      |
| dns_domain                | None                                 |
| id                        | eb96370b-4063-4a4a-9151-39c941fffb83 |
| ipv4_address_scope        | None                                 |
| ipv6_address_scope        | None                                 |
| is_default                | None                                 |
| is_vlan_transparent       | None                                 |
| mtu                       | 1450                                 |
| name                      | private                              |
| port_security_enabled     | True                                 |
| project_id                | b9d3608c81a74b4a953146a302142c26     |
| provider:network_type     | vxlan                                |
| provider:physical_network | None                                 |
| provider:segmentation_id  | 1358                                 |
| qos_policy_id             | None                                 |
| revision_number           | 2                                    |
| router:external           | Internal                             |
| segments                  | None                                 |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   | c9b0c2c8-5dd3-440f-b1d9-fed99d18849e |
| tags                      |                                      |
| updated_at                | 2023-01-11T05:29:40Z                 |
+---------------------------+--------------------------------------+
(openstack) network show public
+---------------------------+--------------------------------------+
| Field                     | Value                                |
+---------------------------+--------------------------------------+
| admin_state_up            | UP                                   |
| availability_zone_hints   |                                      |
| availability_zones        | nova                                 |
| created_at                | 2023-01-11T05:29:49Z                 |
| description               |                                      |
| dns_domain                | None                                 |
| id                        | 4b8396ce-f932-4f64-95d1-3b1274b606e6 |
```

```
| ipv4_address_scope        | None                                 |
| ipv6_address_scope        | None                                 |
| is_default                | True                                 |
| is_vlan_transparent       | None                                 |
| mtu                       | 1500                                 |
| name                      | public                               |
| port_security_enabled     | True                                 |
| project_id                | c4dd2d393580489d9821e51f756814a4     |
| provider:network_type     | flat                                 |
| provider:physical_network | public                               |
| provider:segmentation_id  | None                                 |
| qos_policy_id             | None                                 |
| revision_number           | 2                                    |
| router:external           | External                             |
| segments                  | None                                 |
| shared                    | False                                |
| status                    | ACTIVE                               |
| subnets                   | 6a9ece87-cb33-4e92-b1cc-8335dbdf54c6 |
| tags                      |                                      |
| updated_at                | 2023-01-11T05:29:53Z                 |
+---------------------------+--------------------------------------+


(openstack) port list
+--------------------------------------+-------------+-------------------+-------------
| ID                                   | Name        | MAC Address       | Fixed IP Add
+--------------------------------------+-------------+-------------------+-------------
| 106930bf-7170-4d35-9a3f-e3c3b6596c4a |             | fa:16:3e:86:fb:66 | ip_address='
| 1acb4e56-4d2d-47ee-be9f-8cdcb21444b1 |             | fa:16:3e:99:3f:fe | ip_address='
| 2fd0af9f-92a7-4883-9ab1-322a25a122c8 | direct_port1 | fa:16:3e:64:61:db | ip_address='
| 845e5c57-74a3-409a-b072-fcbdcb2de989 | direct_port2 | fa:16:3e:b6:a6:11 | ip_address='
| b742df40-f426-4761-9acb-804e69f5b14f |             | fa:16:3e:f4:e2:f7 | ip_address='
| d809f1a1-2c3f-4cb7-8883-48049195ebe7 |             | fa:16:3e:08:09:ed | ip_address='
+--------------------------------------+-------------+-------------------+-------------
(openstack) port show direct_port1 -f table --fit-width
+-------------------------+------------------------------------------------------------
| Field                   | Value
+-------------------------+------------------------------------------------------------
| admin_state_up          | UP
| allowed_address_pairs   |
| binding_host_id         | server1
| binding_profile         | capabilities='['switchdev']', card_serial_number='XFL1Y0JRZM
|                         | pf_mac_address='00:0a:35:0d:b9:d0', physical_network=, vf_nu
| binding_vif_details     | bound_drivers.0='openvswitch', bridge_name='br-int', connect
|                         | port_filter='True'
| binding_vif_type        | ovs
| binding_vnic_type       | direct
| created_at              | 2023-01-11T07:28:24Z
| data_plane_status       | None
| description             |
| device_id               | 24b8231e-0ae8-4581-a350-76231ba30ea3
| device_owner            | compute:nova
| device_profile          | None
| dns_assignment          | None
| dns_domain              | None
| dns_name                | None
| extra_dhcp_opts         |
| fixed_ips               | ip_address='10.0.0.52', subnet_id='c9b0c2c8-5dd3-440f-b1d9-f
| id                      | 2fd0af9f-92a7-4883-9ab1-322a25a122c8
```

```
| ip_allocation           | None
| mac_address             | fa:16:3e:64:61:db
| name                    | direct_port1
| network_id              | eb96370b-4063-4a4a-9151-39c941fffb83
| numa_affinity_policy    | None
| port_security_enabled   | True
| project_id              | b9d3608c81a74b4a953146a302142c26
| propagate_uplink_status | None
| qos_network_policy_id   | None
| qos_policy_id           | None
| resource_request        | None
| revision_number         | 21
| security_group_ids      | d8577903-067b-4ea0-911d-ba20dabdafb9
| status                  | ACTIVE
| tags                    |
| trunk_details           | None
| updated_at              | 2023-01-11T09:32:19Z
+-------------------------+--------------------------------------------------------------------
(openstack) port show direct_port2 -f table --fit-width
+-------------------------+--------------------------------------------------------------------
| Field                   | Value
+-------------------------+--------------------------------------------------------------------
| admin_state_up          | UP
| allowed_address_pairs   |
| binding_host_id         | server2
| binding_profile         | capabilities='['switchdev']', card_serial_number='XFL11BYHGM
|                         | pf_mac_address='00:0a:35:0d:b1:60', physical_network=, vf_nu
| binding_vif_details     | bound_drivers.0='openvswitch', bridge_name='br-int', connect
|                         | port_filter='True'
| binding_vif_type        | ovs
| binding_vnic_type       | direct
| created_at              | 2023-01-11T08:07:16Z
| data_plane_status       | None
| description             |
| device_id               | 6b3356f3-5e23-450f-9a6e-14147990b0f2
| device_owner            | compute:nova
| device_profile          | None
| dns_assignment          | None
| dns_domain              | None
| dns_name                | None
| extra_dhcp_opts         |
| fixed_ips               | ip_address='10.0.0.39', subnet_id='c9b0c2c8-5dd3-440f-b1d9-f
| id                      | 845e5c57-74a3-409a-b072-fcbdcb2de989
| ip_allocation           | None
| mac_address             | fa:16:3e:b6:a6:11
| name                    | direct_port2
| network_id              | eb96370b-4063-4a4a-9151-39c941fffb83
| numa_affinity_policy    | None
| port_security_enabled   | True
| project_id              | b9d3608c81a74b4a953146a302142c26
| propagate_uplink_status | None
| qos_network_policy_id   | None
| qos_policy_id           | None
| resource_request        | None
| revision_number         | 18
| security_group_ids      | d8577903-067b-4ea0-911d-ba20dabdafb9
| status                  | ACTIVE
| tags                    |
| trunk_details           | None
```

```
| 2023-01-11T10:27:34Z
+-----------------------------------------------------------------------------------------------------------------------------
```

The Alveo™ U25N is a 2x10/25G SmartNIC. The half-height, half-length (HHHL) Alveo U25N SmartNIC is compliant with the PCI Express® Gen3 x8 (x16 connector). It features the Zynq® UltraScale+™ XCU25 MPSoC and XtremeScale X2 Ethernet controller. The Alveo U25N SmartNIC platform is based on a powerful FPGA, enabling hardware acceleration and offload to happen inline with maximum efficiency while avoiding unnecessary data movements and CPU processing. It is composed of multiple software modules that contain Ethernet drivers and control demons such as vswitchd and strongSwan.

This user guide describes installation, configuration, and operation of the Alveo U25N SmartNIC, as well as its features, performance, and diagnostic tools. For the U25N SmartNIC feature list, refer to the Alveo U25N page.

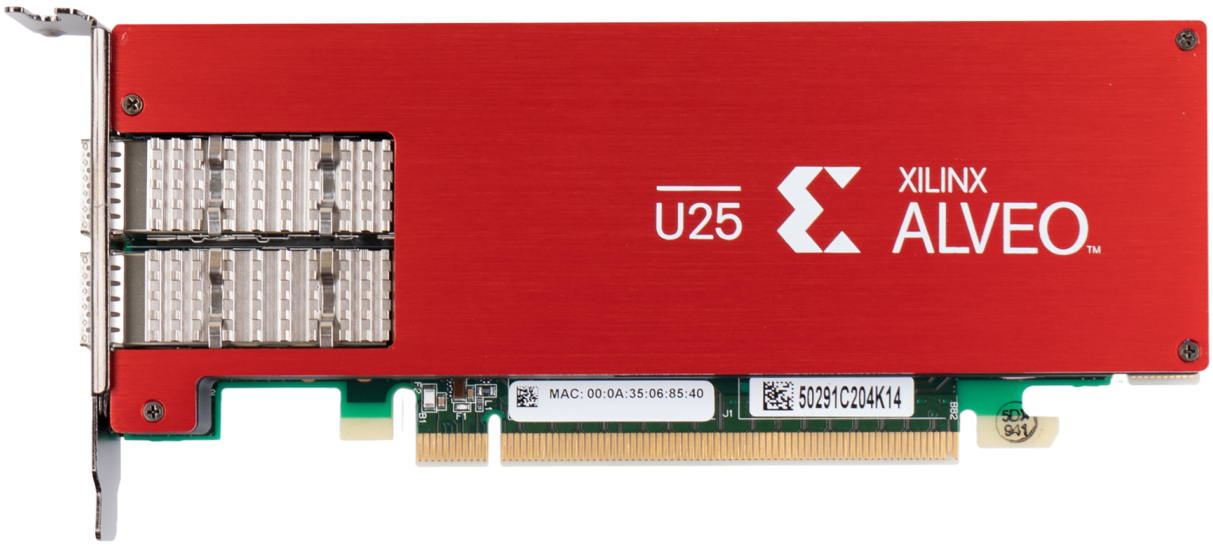


Figure 1: ***Alveo U25N SmartNIC***

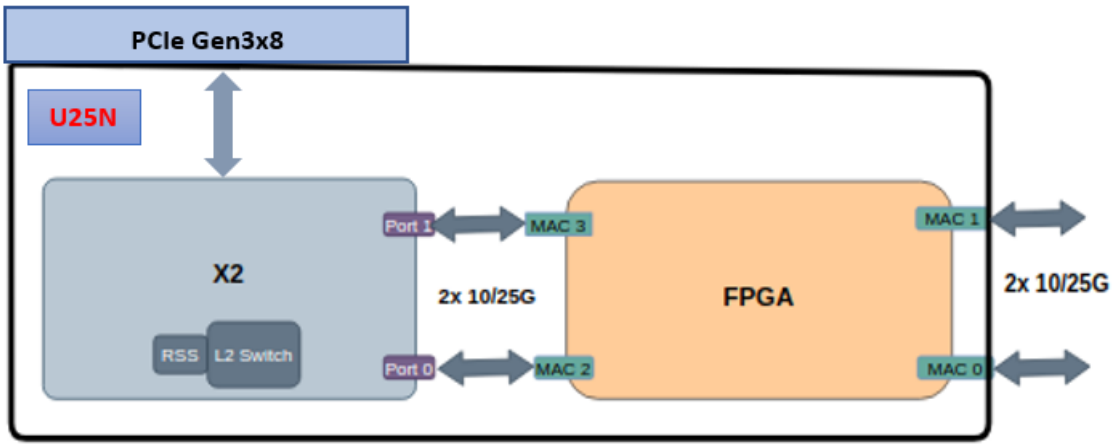


Figure 2: ***U25N Architecture***