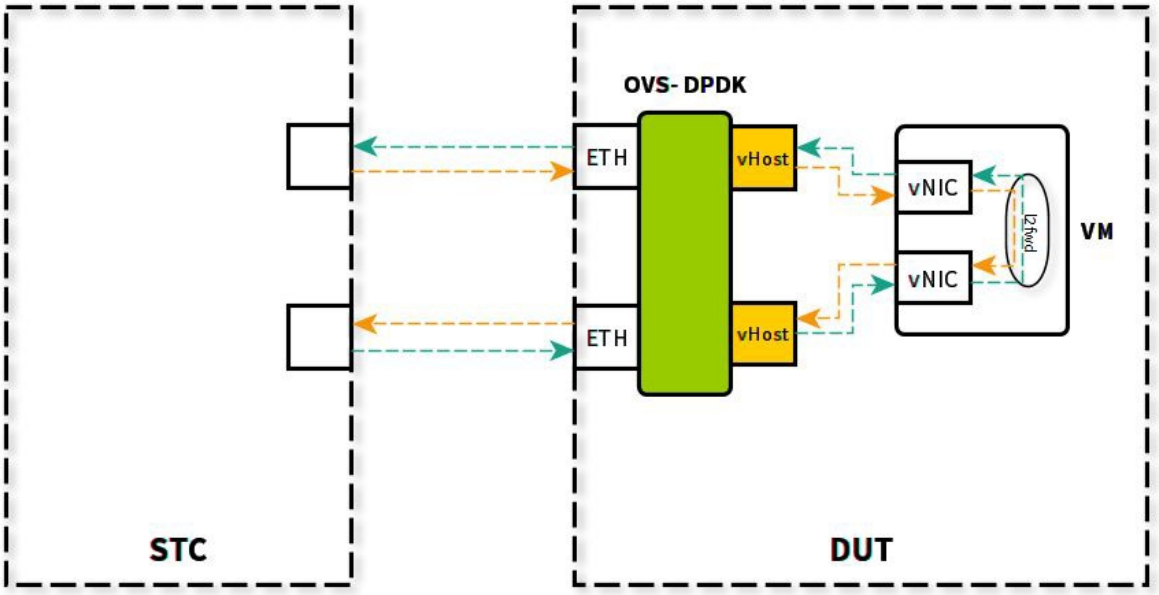
OVS-DPDK VM test models

Model 1: 1 VM



Step 1. Export environment:

```
export OVS_PREFIX=/home/chitam/workspace/code/ovs_workspace/install/ovs
export br_name=nctam-dpdk-br1
export vport0=nctam_vhost_port0
export vport1=nctam_vhost_port1
export phy_port0=nctam_phy_port0
export phy_port1=nctam_phy_port1
export pcie_phy_port0="0000:04:00.0"
export pcie_phy_port1="0000:04:00.1"
export mem_s0="4096"
export mem_s1="4096"
export per_port_memory="true"
export dpdk_prefix=${br_name}-pref
export pmd_cpu_mask="0x003C0"
export dpdk_lcore_mask="0xC0030000"
```

Step 2. Start qemu. QEMU will wait for client connection for it actually starts at the first time.

```
sudo qemu-system-x86_64 -enable-kvm -nographic -cpu host -smp 2 -m 4096 -hda core-ubuntu.qcow \
-device virtio-net-pci,netdev=net0 -netdev user,id=net0,hostfwd=tcp::5559-:22 \
-chardev socket,id=char1,path=${OVS_PREFIX}/var/run/openvswitch/${vport0},server \
-netdev type=vhost-user,id=${vport0},chardev=char1,vhostforce \
-object memory-backend-file,id=mem1,size=2G,mem-path=/mnt/huge,share=on \
-numa node,memdev=mem1 -mem-prealloc \
-device virtio-net-pci,mac=A0:01:00:00:00:00,netdev=${vport0} \
-chardev socket,id=char2,path=${OVS_PREFIX}/var/run/openvswitch/${vport1},server \
-netdev type=vhost-user,id=${vport1},chardev=char2,vhostforce \
-object memory-backend-file,id=mem2,size=2G,mem-path=/mnt/huge,share=on \
-numa node,memdev=mem2 \
-mem-prealloc \
-device virtio-net-pci,mac=A0:02:00:00:00:00,netdev=${vport1} \
-name vm1
```

Step 3. Open new terminal and login with root user:

```
sudo su
```

Step 4. Rerun step 1 (export environment variables):

Step 5. Close current instance OVS and clear all data:

```
ovs-ctl stop
rm -rf ${OVS_PREFIX}/etc/openvswitch/conf.db
```

Step 6. Start ovsdb-server only:

```
ovs-ctl --no-ovs-vswitchd start
```

Step 7. Configure ovs-dpdk:

```
ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-init=true \
-- set Open_vSwitch . other_config:dpdk-file-prefix="${dpdk_prefix}" \
-- set Open_vSwitch . other_config:dpdk-lcore-mask="${dpdk_lcore_mask}" \
-- set Open_vSwitch . other_config:pmd-cpu-mask="${pmd_cpu_mask}" \
-- set Open_vSwitch . other_config:dpdk-socket-mem="${mem_s0},${mem_s1}" \
-- set Open_vSwitch . other_config:dpdk-pci-whitelist="${pcie_phy_port0},${pcie_phy_port1}" \
-- set Open_vSwitch . other_config:per-port-memory="${per_port_memory}"
```

Step 8. Add bridge:

```
ovs-vsctl --no-wait  add-br ${br_name} -- \
set bridge ${br_name} datapath_type=netdev fail-mode=secure
```

Step 9. Add port ovs:

```
ovs-vsctl --no-wait add-port ${br_name} ${vport0} \
    -- set Interface ${vport0} type=dpdkvhostuserclient \
        options:vhost-server-path=${OVS_PREFIX}/var/run/openvswitch/${vport0}  ofport=1

ovs-vsctl --no-wait add-port ${br_name} ${vport1} \
    -- set Interface ${vport1} type=dpdkvhostuserclient \
        options:vhost-server-path=${OVS_PREFIX}/var/run/openvswitch/${vport1}  ofport=2

ovs-vsctl --no-wait add-port ${br_name} ${phy_port0} -- set Interface ${phy_port0} \
    type=dpdk options:dpdk-devargs=${pcie_phy_port0} ofport=3

ovs-vsctl --no-wait add-port ${br_name} ${phy_port1} -- set Interface ${phy_port1} \
    type=dpdk options:dpdk-devargs=${pcie_phy_port1} ofport=4
```

Step 10. Start open-vswitch:

```
ovs-ctl start
```

Step 11. Start `l2fwd` application in VM

Step 11.1. SSH to VM **`bash ssh -p 5559 linux@127.0.0.1`**

Step 11.2. Load dpdk kernel module:

```
cd /path/to/dpdk
export RTE_SDK=`pwd`
export RTE_TARGET=x86_64-native-linuxapp-gcc
sudo rmmod igb_uio
sudo modprobe uio
sudo insmod ${RTE_TARGET}/kmod/igb_uio.ko
sudo ./usertools/dpdk-devbind.py -b igb_uio 0000:00:04.0 0000:00:05.0
```

Step 11.3. Build **`l2fwd`** application:

```
cd examples/l2fwd
make
cd $RTE_SDK
```

Step 11.4. Start **`l2pwd`**:

```
sudo examples/l2fwd/build/app/l2fwd -l 0-1 -n 4 -w 0000:00:04.0 -w 0000:00:05.0  -- -p 3 -T 2
```

Note: This model dont setup any flow for packet

Test Cases:

Test Case 1: 1 flow packets, 1 VM without HQoS feature

In this test case, we use **Model 1** in section *Models* above.

Please folow the step in **Model 1** to setup **OVS-DPDK** and VM machine. Then run below step to finish test case setup.

Step 1: Add flow for OVS-DPDK bridge.

```
ovs-ofctl add-flow ${br_name} cookie=0x1,table=0,in_port=3,action=output:1
ovs-ofctl add-flow ${br_name} cookie=0x2,table=0,in_port=2,action=output:4
ovs-ofctl add-flow ${br_name} cookie=0x3,table=0,in_port=4,action=output:2
ovs-ofctl add-flow ${br_name} cookie=0x4,table=0,in_port=1,action=output:3
```

Now you setup STC to transmit packet and review return packet.

Test Case 2: 1 flows packets, 1 VM with HQoS feature

In this test case, we use **Model 1** in section *Models* above. Add HQoS to 2 port *vhost-user* with following hierarchy:

- 1 subport per port, bandwidth = port rate (10G).
- 1 pipe per subport, bandwidth = subport's bandwidth (10G).
- 4 TCs per pipe, bandwidth = pipe's bandwidth (10G).
- 4 queues per TC.

Please folow the step in **Model 1** to setup **OVS-DPDK** and VM machine. Then run below step to finish test case setup.

Step 1: Add flow for OVS-DPDK bridge.

```
ovs-ofctl add-flow ${br_name} cookie=0x1,table=0,in_port=3,action=output:1
ovs-ofctl add-flow ${br_name} cookie=0x2,table=0,in_port=2,action=output:4
ovs-ofctl add-flow ${br_name} cookie=0x3,table=0,in_port=4,action=output:2
ovs-ofctl add-flow ${br_name} cookie=0x4,table=0,in_port=1,action=output:3
```

Step 2. Add HQoS for vport0:

```
ovs-vsctl --no-wait set port ${vport0} \
    qos=@newqos -- --id=@newqos create QOS type=egress_hqos \
    other-config:frame-overhead=24 other-config:n-subports-per-port=${n_subport} \
    other-config:n-pipes-per-subport=${n_pipe} other-config:qsize="64 64 64 64" \
    other-config:subport-profile-map="0" \
    other-config:pipe_profiles_subport_0="0_${n_pipe}:0" \
    other-config:tc-0-wred-min="48 40 32" \
    other-config:tc-0-wred-max="64 64 64" \
    other-config:tc-0-wred-inv-prob="10 10 10" \
    other-config:tc-0-wred-weight="9 9 9" \
    other-config:tc-1-wred-min="48 40 32" \
    other-config:tc-1-wred-max="64 64 64" \
    other-config:tc-1-wred-inv-prob="10 10 10" \
    other-config:tc-1-wred-weight="9 9 9" \
    other-config:tc-2-wred-min="48 40 32" \
    other-config:tc-2-wred-max="64 64 64" \
    other-config:tc-2-wred-inv-prob="10 10 10" \
    other-config:tc-2-wred-weight="9 9 9" \
    other-config:tc-3-wred-min="48 40 32" \
    other-config:tc-3-wred-max="64 64 64" \
    other-config:tc-3-wred-inv-prob="10 10 10" \
    other-config:tc-3-wred-weight="9 9 9" \
    subports:0=@subport_profile0 \
    pipes:0=@pipe_profile0 -- --id=@subport_profile0 create qos type=subport_param \
    other-config:tb-rate=${subport_rate} other-config:tb-size=1000000 \
    other-config:tc-0-rate=${subport_rate} \
    other-config:tc-1-rate=${subport_rate} \
    other-config:tc-2-rate=${subport_rate} \
    other-config:tc-3-rate=${subport_rate} \
    other-config:tc-period=10 -- --id=@pipe_profile0 create qos type=pipe_param \
    other-config:tb-rate=${pipe_rate} other-config:tb-size=1000000 \
    other-config:tc-0-rate=${pipe_rate} \
    other-config:tc-1-rate=${pipe_rate} other-config:tc-2-rate=${pipe_rate} \
    other-config:tc-3-rate=${pipe_rate} other-config:tc-period=40 \
    other-config:tc-0-wrr="1 1 1 1" other-config:tc-1-wrr="1 1 1 1" \
    other-config:tc-2-wrr="1 1 1 1" other-config:tc-3-wrr="1 1 1 1"
```

Step 13. Add HQoS for vport1:

```
ovs-vsctl --no-wait set port ${vport1} \
    qos=@newqos -- --id=@newqos create QOS type=egress_hqos \
    other-config:frame-overhead=24 other-config:n-subports-per-port=${n_subport} \
    other-config:n-pipes-per-subport=${n_pipe} other-config:qsize="64 64 64 64" \
    other-config:subport-profile-map="0" \
    other-config:pipe_profiles_subport_0="0_${n_pipe}:0" \
    other-config:tc-0-wred-min="48 40 32" \
    other-config:tc-0-wred-max="64 64 64" \
    other-config:tc-0-wred-inv-prob="10 10 10" \
    other-config:tc-0-wred-weight="9 9 9" \
    other-config:tc-1-wred-min="48 40 32" \
    other-config:tc-1-wred-max="64 64 64" \
    other-config:tc-1-wred-inv-prob="10 10 10" \
    other-config:tc-1-wred-weight="9 9 9" \
    other-config:tc-2-wred-min="48 40 32" \
    other-config:tc-2-wred-max="64 64 64" \
    other-config:tc-2-wred-inv-prob="10 10 10" \
    other-config:tc-2-wred-weight="9 9 9" \
    other-config:tc-3-wred-min="48 40 32" \
    other-config:tc-3-wred-max="64 64 64" \
    other-config:tc-3-wred-inv-prob="10 10 10" \
    other-config:tc-3-wred-weight="9 9 9" \
    subports:0=@subport_profile0 \
    pipes:0=@pipe_profile0 -- --id=@subport_profile0 create qos type=subport_param \
    other-config:tb-rate=${subport_rate} other-config:tb-size=1000000 \
    other-config:tc-0-rate=${subport_rate} \
    other-config:tc-1-rate=${subport_rate} \
    other-config:tc-2-rate=${subport_rate} \
    other-config:tc-3-rate=${subport_rate} \
    other-config:tc-period=10 -- --id=@pipe_profile0 create qos type=pipe_param \
    other-config:tb-rate=${pipe_rate} other-config:tb-size=1000000 \
    other-config:tc-0-rate=${pipe_rate} \
    other-config:tc-1-rate=${pipe_rate} other-config:tc-2-rate=${pipe_rate} \
    other-config:tc-3-rate=${pipe_rate} other-config:tc-period=40 \
    other-config:tc-0-wrr="1 1 1 1" other-config:tc-1-wrr="1 1 1 1" \
    other-config:tc-2-wrr="1 1 1 1" other-config:tc-3-wrr="1 1 1 1"
```

Now you setup STC to transmit packet and review return packet.