AT OVS

Tasks Note

Contents

Introduction

```
This is used to note task & solve it information (implementation & other information)
```

Abbreviation and Terminologies

SLA Service-level agreement

MTU A maximum transmission unit (MTU) is the largest packet or frame size, specified in octets (eight-bit bytes) that can be sent in a packet- or frame-based network such as the internet

CoS Class of Service

CIR/EIR committed information rate (CIR)/Excess Information Rate (EIR) defines how fast token will be refilled.

CBS/EBS Committed Burst Size (CBS)/Excess Burst Size (EBS) defines maximum size of token buckets, CBS is burst size of CIR & EBS is burst size of EIR. CBS and/or EBS have to greater than 0 and should be larger than MTU

1. Task Note

1.1. Investigate “Configuration of dpdkvhostuserclient type” (Nov 21, 2019)

```
XXX: Update this information to correctly part

# OVS side: Set socket path that will be created by QEMU
$ sudo ovs-vsctl set Interface ovs-br0-vport1
options:vhost-server-path=${HOME}/ovs_prj/ovs-software/var/run/openvswitch/dpdkvhostuserclient/ovs-vport1-s

# Add & set port
$ sudo ovs-vsctl add-port ovs-dpdk-br0 ovs-br0-vport2 -- set Interface ovs-br0-vport2 \
type=dpdkvhostuserclient
options:vhost-server-path=${HOME}/ovs_prj/ovs-software/var/run/openvswitch/dpdkvhostuserclient/ovs-vport2-s


# QEMU side: Start VM1
$ sudo qemu-system-x86_64 -m 1024 -smp 1 -cpu host -hda ${HOME}/qemu/lubuntu01.qcow2 -boot c
-enable-kvm -no-reboot \
-net none -chardev
socket,id=char1,path=${HOME}/ovs_prj/ovs-software/var/run/openvswitch/dpdkvhostuserclient/ovs-vport1-socket
\
-netdev type=vhost-user,id=ovs-br0-vport1,chardev=char1,vhostforce      \
-device virtio-net-pci,mac=A0:01:00:00:00:00,netdev=ovs-br0-vport1      \
-object memory-backend-file,id=mem,size=1G,mem-path=/mnt/huge,share=on -numa node,memdev=mem
-mem-prealloc      \
-virtfs local,path=${HOME}/iperf_debs,mount_tag=host0,security_model=none,id=vm1_dev
# QEMU side: Start VM2
sudo qemu-system-x86_64 -m 1024 -smp 1 -cpu host -hda ${HOME}/qemu/lubuntu02.qcow2 -boot c
-enable-kvm -no-reboot      \
-net none -chardev
socket,id=char2,path=${HOME}/ovs_prj/ovs-software/var/run/openvswitch/dpdkvhostuserclient/ovs-vport2-socket
\
-netdev type=vhost-user,id=ovs-br0-vport2,chardev=char2,vhostforce      \
-device virtio-net-pci,mac=A0:02:00:00:00:00,netdev=ovs-br0-vport2      \
-object memory-backend-file,id=mem,size=1G,mem-path=/mnt/huge,share=on -numa node,memdev=mem
-mem-prealloc      \
-virtfs local,path=${HOME}/iperf_debs,mount_tag=host0,security_model=none,id=vm2_dev
```

1.2. OVS Status & Statistics (Oct 31, 2019)

```
Investigate & collect all command to show status & statistics of OVS. List out all information that
is not supported to show out, make the plan to do it.

Note:
    + Total packages & total data (size) go through PORT/INTERFACE?
```

```
    + How many packages of every package size (dropted & sent) at PORT/INTERFACE.
    + Total packages & total data (size) go through Queue (or QoS object: policer, shaper)?
    + How many packages of every package size (dropted & sent) at Queue (or QoS object: policer,
shaper).
    + Show bandwidth of Queue/PORT/INTERFACE (using to check the bandwidth inside OVS)
    + Package flow & detail about Thread that handled it.
    + Detail about metadata (What can we do with it?)
```

1.2.1. Current status

```
Final: support to show Queue's statistics
    + Support "ovs-ofctl queue-stats switch [Interface [Queue ID]] (follow Open Flow standards)
    + Add command to show detail queue's statistics, refer to __Task Note/Egress HQoS 2 level
policer(Sept 19, 2019)__
```

1.2.2. Issue

- **How to show status & statistic of OVS ?** refer to __OVS Commands/OVS Status & Statistic Commands
- **Show information of Queue (Should be implemented if it is not exist)?** Not supported so we will do it, refer to **Task Note/Egress HQoS 2 level policer(Sept 19, 2019)**
- **Get any statistics information from database?** Not started yet

1.3. Egress HQoS 2 level policer(Sept 19, 2019)

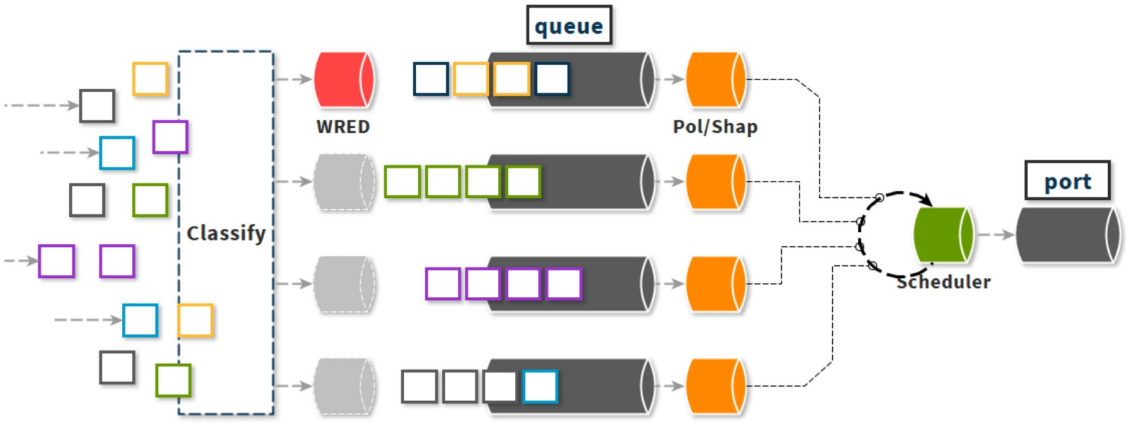


Figure 1-1 - HQoS2L Policer

Add feature: show statistics for QoS Queue (the result is the same format with statistic of interface) - Nov 4,2019

```
statistics          : {"rx_1024_to_1522_packets"=0, "rx_128_to_255_packets"=15,
"rx_1523_to_max_packets"=0, "rx_1_to_64_packets"=7, "rx_256_to_511_packets"=5,
"rx_512_to_1023_packets"=0, "rx_65_to_127_packets"=8, rx_bytes=5469, rx_dropped=0, rx_errors=0,
rx_packets=35, tx_bytes=2327, tx_dropped=21, tx_packets=22}
```

Note: gitlab tracking task: **Add 2 level egress policer in OVS&DPDK**

Add features: Update configuration of hqos2l_policer

1.3.1. Current status

- Investigate about *.ovsschema & add more column: done

- Define configuration command for testing: done
- Investiage & coding: Done
- Testing: refer to **${This Task}/Egress HQoS 2 level policer Testing**
- Investigate statistic in OVS: Done (Refer to **Task Note/OVS Status & Statistics (Oct 31, 2019)**)
- Investigate & Coding: Done
- Openflow classifies (OVS without DPDK): Refer to **${This Task}/Openflow classifies (OVS without DPDK)**
- Show Queue Statistics:
  - Support command ovs-ofctl queue-stats (OpenFlow 1.5.1): Done
  - Add command to show more detail Queue's statistics: Done
- STC Testing: Done (for more information about this once, please check the comment in gitlab)
- Documentation: On going

1.3.2. Issue

- **Detail about n traffic class (n Queue) & n+1 Policer ?**
  - PL1: srTCM rfc2698 (should be send or drop only)
  - PL2x: trTCM rfc2698 or rfc4115 (currently, compatible DPDK version just supports rfc2698 so will use it first.)
  - Scheduler: Not Included
  - Classifier will be handle by Open Flow
- **What information should be got from user ?**
  - PL1: cir, cbs, pir & pbs (rfc2698) (cir == 0 mean unlimited)
  - PL2x: cir, cbs, pir & pbs (rfc2698) OR cir, cbs, eir & ebs (rfc4115) & color actions
- **Detail about Scheduler algorithm of scheduler)?** No thing to do
- **How to classifier ?** refer to **${this task}/Open Flow classifier (OVS + DPDK)**
- **How many process or core will be used ?** For the first version, try to process on one core (many thread if neaded)
- **What is configuration command ?** Refer to **${THIS TASK}/Configuration Command** session.
- **How to add new column in ovsdb ?** refer to ***.ovsschema file (Json format)/How to del or add columns in ovsdb (update vswitch.ovsschema)** session.
- **How to test this qos type ?** Refer to session **OVS Commands/QoS user command line** of this document
- **Which library in DPDK can be use?**
  - QoS library
  - Ring buffer (No need to prepare Queue for policer)
- **error "ovs-vsctl: Queue does not contain a column whose name matches "qos""** We have to define new "ovsschema" for new column or base on old "ovsschema", define correctly command.
  - solution 1: define new "ovsschema": Refer **${*.ovsschema file (Json format)}/How to del/add columns in ovsdb (update vswitch.ovsschema)** session.
  - solution 2: base one current *.ovsschema, define a correctly command (TBD)

- **Which command should we use to show the Queue Statistics ?** refer to **${this task}/Which command should be used to show Queue statistic**

1.3.3. Openflow classifies (OVS without DPDK)

- **Openflow classifies (OVS without DPDK): package –> (openflow classifies) –> set(skb_priority(Queue ID)):**

```
/* user space */
static void
        xlate_set_queue_action(struct xlate_ctx *ctx, uint32_t queue_id)
{
    uint32_t skb_priority;

    if (!dpif_queue_to_priority(ctx->xbridge->dpif, queue_id, &skb_priority)) {
        ctx->xin->flow.skb_priority = skb_priority;
    } else {
        /* Couldn't translate queue to a priority.  Nothing to do.  A warning
         * has already been logged. */
    }
}


/* Kernel data path */
static int execute_masked_set_action(struct sk_buff *skb,
        struct sw_flow_key *flow_key,
        const struct nlattr *a)
{
    int err = 0;
    switch (nla_type(a)) {
    case OVS_KEY_ATTR_PRIORITY:
        OVS_SET_MASKED(skb->priority, nla_get_u32(a),
        *get_mask(a, u32 *));
        flow_key->phy.priority = skb->priority;
        break;
```

- **QoS of Tc htb (kernel linux) will run base on skb->priority information.**

1.3.4. Which command should be used to show Queue statistic

1.3.5. Show detail queue's information: sudo ovs-appctl qos/show-queue-stats INTERFACE [Queue ID] (defined by ATVN)

```
`$ sudo ovs-appctl qos/show-queue-stats ovs-br0-vport2`
Queues's Statistics of interface ovs-br0-vport2 (4 queues):
Queue 8: tx_packets:0, tx_bytes:0,tx_dropped:0, tx_errors:0, duration=-9223372036854775808,
        tx_1_to_64_packets:0, tx_65_to_127_packets:0, tx_128_to_255_packets:0,
        tx_256_to_511_packets:0,tx_512_to_1023_packets:0, tx_1024_to_1522_packets:0,
tx_1523_to_max_packets:0.
Queue 1: tx_packets:0, tx_bytes:0,tx_dropped:0, tx_errors:0, duration=-9223372036854775808,
        tx_1_to_64_packets:0, tx_65_to_127_packets:0, tx_128_to_255_packets:0,
        tx_256_to_511_packets:0,tx_512_to_1023_packets:0, tx_1024_to_1522_packets:0,
tx_1523_to_max_packets:0.
Queue 2: tx_packets:0, tx_bytes:0,tx_dropped:0, tx_errors:0, duration=-9223372036854775808,
        tx_1_to_64_packets:0, tx_65_to_127_packets:0, tx_128_to_255_packets:0,
        tx_256_to_511_packets:0,tx_512_to_1023_packets:0, tx_1024_to_1522_packets:0,
tx_1523_to_max_packets:0.
Queue 4: tx_packets:0, tx_bytes:0,tx_dropped:0, tx_errors:0, duration=-9223372036854775808,
        tx_1_to_64_packets:0, tx_65_to_127_packets:0, tx_128_to_255_packets:0,
        tx_256_to_511_packets:0,tx_512_to_1023_packets:0, tx_1024_to_1522_packets:0,
tx_1523_to_max_packets:0.

`$ sudo ovs-appctl qos/show-queue-stats ovs-br0-vport2 1`
Queue's Statistics of interface ovs-br0-vport2 (Queue 1):
Queue 1: tx_packets:0, tx_bytes:0,tx_dropped:0, tx_errors:0, duration=-9223372036854775808,
        tx_1_to_64_packets:0, tx_65_to_127_packets:0, tx_128_to_255_packets:0,
```

```
        tx_256_to_511_packets:0,tx_512_to_1023_packets:0,  tx_1024_to_1522_packets:0,
tx_1523_to_max_packets:0.
```

1.3.6. ovs-ofctl queue-stats switch [port [queue]]

Final: We can use this command to show Queue's statistics (have to follow open flow so we cannot show too many information here)

```
This one has not implemented yet, so we will do it (current status: done)
`$ sudo ovs-ofctl queue-stats ovs-dpdk-br0 ovs-br0-vport2`
OFPST_QUEUE reply (xid=0x4): 4 queues
port "ovs-br0-vport2" queue 4: bytes=0, pkts=0, errors=0, duration=?
port "ovs-br0-vport2" queue 8: bytes=0, pkts=0, errors=0, duration=?
port "ovs-br0-vport2" queue 2: bytes=0, pkts=0, errors=0, duration=?
port "ovs-br0-vport2" queue 1: bytes=0, pkts=0, errors=0, duration=?

`$ sudo ovs-ofctl queue-stats ovs-dpdk-br0 ovs-br0-vport2 1`
OFPST_QUEUE reply (xid=0x4): 1 queues
port "ovs-br0-vport2" queue 1: bytes=0, pkts=0, errors=0, duration=?
```

1.3.6.1. ovs-ofctl dump-ports [BRIDGE_NAME][PORT_NAME]

Final: We cannot used this command to show Queue Statistic, the replies message has to follow Open Flow Standard.__ Now, the OpenFlow14,15&16 support to show more detail of size of package, Will update code to show TX package for PORT. Refer below for more information about basic of ovs-ofctl command:

- **ovs-ofctl side, supported commands list** ``` /* Supported commands list */ static const struct ovs_cmdl_command all_commands[] = { { “show”, “switch”, 1, 1, ofctl_show, OVS_RO }, { “monitor”, “switch [misslen][invalid_ttl] [watch:[…]]”, 1, 3, ofctl_monitor, OVS_RO }, { “snoop”, “switch”, 1, 1, ofctl_snoop, OVS_RO }, { “dump-desc”, “switch”, 1, 1, ofctl_dump_desc, OVS_RO }, { “dump-tables”, “switch”, 1, 1, ofctl_dump_tables, OVS_RO }, { “dump-table-features”, “switch”, 1, 1, ofctl_dump_table_features, OVS_RO }, { “dump-table-desc”, “switch”, 1, 1, ofctl_dump_table_desc, OVS_RO }, { “dump-flows”, “switch”, 1, 2, ofctl_dump_flows, OVS_RO }, { “dump-aggregate”, “switch”, 1, 2, ofctl_dump_aggregate, OVS_RO }, { “queue-stats”, “switch [port [queue]]”, 1, 3, ofctl_queue_stats, OVS_RO }, { “queue-get-config”, “switch [port [queue]]”, 1, 3, ofctl_queue_get_config, OVS_RO }, { “add-flow”, “switch flow”, 2, 2, ofctl_add_flow, OVS_RW }, { “add-flows”, “switch file”, 2, 2, ofctl_add_flows, OVS_RW }, { “mod-flows”, “switch flow”, 2, 2, ofctl_mod_flows, OVS_RW }, { “del-flows”, “switch [flow]”, 1, 2, ofctl_del_flows, OVS_RW }, { “replace-flows”, “switch file”, 2, 2, ofctl_replace_flows, OVS_RW }, { “diff-flows”, “source1 source2”, 2, 2, ofctl_diff_flows, OVS_RW }, { “add-meter”, “switch meter”, 2, 2, ofctl_add_meter, OVS_RW }, { “mod-meter”, “switch meter”, 2, 2, ofctl_mod_meter, OVS_RW }, { “del-meter”, “switch meter”, 1, 2, ofctl_del_meters, OVS_RW }, { “del-meters”, “switch”, 1, 2, ofctl_del_meters, OVS_RW }, { “dump-meter”, “switch meter”, 1, 2, ofctl_dump_meters, OVS_RO }, { “dump-meters”, “switch”, 1, 2, ofctl_dump_meters, OVS_RO }, { “meter-stats”, “switch [meter]”, 1, 2, ofctl_meter_stats, OVS_RO }, { “meter-features”, “switch”, 1, 1, ofctl_meter_features, OVS_RO }, { “packet-out”, “switch "in_port= packet= actions=…"”, 2, INT_MAX, ofctl_packet_out, OVS_RW }, { “dump-ports”, “switch [port]”, 1, 2, ofctl_dump_ports, OVS_RO }, { “dump-ports-desc”, “switch [port]”, 1, 2, ofctl_dump_ports_desc, OVS_RO }, { “mod-port”, “switch iface act”, 3, 3, ofctl_mod_port, OVS_RW }, { “mod-table”, “switch mod”, 3, 3, ofctl_mod_table, OVS_RW }, { “get-frags”, “switch”, 1, 1, ofctl_get_frags, OVS_RO }, { “set-frags”, “switch frag_mode”, 2, 2, ofctl_set_frags, OVS_RW }, { “probe”, “target”, 1, 1, ofctl_probe, OVS_RO }, { “ping”, “target [n]”, 1, 2, ofctl_ping, OVS_RO }, { “benchmark”, “target n count”, 3, 3, ofctl_benchmark, OVS_RO }, { “dump-ipfix-bridge”, “switch”, 1, 1, ofctl_dump_ipfix_bridge, OVS_RO }, { “dump-ipfix-flow”, “switch”, 1, 1, ofctl_dump_ipfix_flow, OVS_RO }, { “ct-flush-zone”, “switch zone”, 2, 2, ofctl_ct_flush_zone, OVS_RO }, { “ofp-parse”, “file”, 1, 1, ofctl_ofp_parse, OVS_RW }, { “ofp-parse-pcap”, “pcap”, 1, INT_MAX, ofctl_ofp_parse_pcap, OVS_RW }, { “add-group”, “switch group”, 1, 2, ofctl_add_group, OVS_RW }, { “add-groups”, “switch file”, 1, 2, ofctl_add_groups, OVS_RW }, { “mod-group”, “switch group”, 1, 2, ofctl_mod_group, OVS_RW }, { “del-groups”, “switch [group]”, 1, 2, ofctl_del_groups, OVS_RW }, { “insert-buckets”, “switch [group]”, 1, 2, ofctl_insert_bucket, OVS_RW }, { “remove-buckets”, “switch [group]”, 1, 2, ofctl_remove_bucket, OVS_RW }, { “dump-groups”, “switch [group]”, 1, 2, ofctl_dump_group_desc, OVS_RO }, { “dump-group-stats”, “switch [group]”, 1, 2,

ofctl_dump_group_stats, OVS_RO }, { “dump-group-features”, “switch”, 1, 1, ofctl_dump_group_features, OVS_RO }, { “bundle”, “switch file”, 2, 2, ofctl_bundle, OVS_RW }, { “add-tlv-map”, “switch map”, 2, 2, ofctl_add_tlv_map, OVS_RO }, { “del-tlv-map”, “switch [map]”, 1, 2, ofctl_del_tlv_map, OVS_RO }, { “dump-tlv-map”, “switch”, 1, 1, ofctl_dump_tlv_map, OVS_RO }, { “help”, NULL, 0, INT_MAX, ofctl_help, OVS_RO }, { “list-commands”, NULL, 0, INT_MAX, ofctl_list_commands, OVS_RO }, /* Undocumented commands for testing. */ { “parse-flow”, NULL, 1, 1, ofctl_parse_flow, OVS_RW }, { “parse-flows”, NULL, 1, 1, ofctl_parse_flows, OVS_RW }, { “parse-nx-match”, NULL, 0, 0, ofctl_parse_nxm, OVS_RW }, { “parse-nxm”, NULL, 0, 0, ofctl_parse_nxm, OVS_RW }, { “parse-oxm”, NULL, 1, 1, ofctl_parse_oxm, OVS_RW }, { “parse-actions”, NULL, 1, 1, ofctl_parse_actions, OVS_RW }, { “parse-instructions”, NULL, 1, 1, ofctl_parse_instructions, OVS_RW }, { “parse-ofp10-match”, NULL, 0, 0, ofctl_parse_ofp10_match, OVS_RW }, { “parse-ofp11-match”, NULL, 0, 0, ofctl_parse_ofp11_match, OVS_RW }, { “parse-pcap”, NULL, 1, INT_MAX, ofctl_parse_pcap, OVS_RW }, { “check-vlan”, NULL, 2, 2, ofctl_check_vlan, OVS_RW }, { “print-error”, NULL, 1, 1, ofctl_print_error, OVS_RW }, { “encode-error-reply”, NULL, 2, 2, ofctl_encode_error_reply, OVS_RW }, { “ofp-print”, NULL, 1, 2, ofctl_ofp_print, OVS_RW }, { “encode-hello”, NULL, 1, 1, ofctl_encode_hello, OVS_RW }, { “parse-key-value”, NULL, 1, INT_MAX, ofctl_parse_key_value, OVS_RW }, { “compose-packet”, NULL, 1, 2, ofctl_compose_packet, OVS_RO }, { “parse-packet”, NULL, 0, 0, ofctl_parse_packet, OVS_RO }, { NULL, NULL, 0, 0, NULL, OVS_RO }, };

/* Connection with ovs-vswitchd */ *struct vconn_class* { */* Prefix for connection names, e.g. “nl”, “tcp”.*/ const char* name;

```
/* Attempts to connect to an OpenFlow device.  'name' is the full
 * connection name provided by the user, e.g. "tcp:1.2.3.4".  This name is
 * useful for error messages but must not be modified.
 *
 * 'allowed_versions' is the OpenFlow versions that may be
 * negotiated for a connection.
 *
 * 'suffix' is a copy of 'name' following the colon and may be modified.
 * 'dscp' is the DSCP value that the new connection should use in the IP
 * packets it sends.
 *
 * Returns 0 if successful, otherwise a positive errno value.  If
 * successful, stores a pointer to the new connection in '*vconnp'.
 *
 * The open function must not block waiting for a connection to complete.
 * If the connection cannot be completed immediately, it should return
 * EAGAIN (not EINPROGRESS, as returned by the connect system call) and
 * continue the connection in the background. */
int (*open)(const char *name, uint32_t allowed_versions,
            char *suffix, struct vconn **vconnp, uint8_t dscp);

/* Closes 'vconn' and frees associated memory. */
void (*close)(struct vconn *vconn);

/* Tries to complete the connection on 'vconn'.  If 'vconn''s connection is
 * complete, returns 0 if the connection was successful or a positive errno
 * value if it failed.  If the connection is still in progress, returns
 * EAGAIN.
 *
 * The connect function must not block waiting for the connection to
 * complete; instead, it should return EAGAIN immediately. */
int (*connect)(struct vconn *vconn);

/* Tries to receive an OpenFlow message from 'vconn'.  If successful,
 * stores the received message into '*msgp' and returns 0.  The caller is
 * responsible for destroying the message with ofpbuf_delete().  On
 * failure, returns a positive errno value and stores a null pointer into
 * '*msgp'.
 *
 * If the connection has been closed in the normal fashion, returns EOF.
 *
 * The recv function must not block waiting for a packet to arrive.  If no
 * packets have been received, it should return EAGAIN. */
int (*recv)(struct vconn *vconn, struct ofpbuf **msgp);

/* Tries to queue 'msg' for transmission on 'vconn'.  If successful,
```

```
 * returns 0, in which case ownership of 'msg' is transferred to the vconn.
 * Success does not guarantee that 'msg' has been or ever will be delivered
 * to the peer, only that it has been queued for transmission.
 *
 * Returns a positive errno value on failure, in which case the caller
 * retains ownership of 'msg'.
 *
 * The send function must not block.  If 'msg' cannot be immediately
 * accepted for transmission, it should return EAGAIN. */
int (*send)(struct vconn *vconn, struct ofpbuf *msg);

/* Allows 'vconn' to perform maintenance activities, such as flushing
 * output buffers.
 *
 * May be null if 'vconn' doesn't have anything to do here. */
void (*run)(struct vconn *vconn);

/* Arranges for the poll loop to wake up when 'vconn' needs to perform
 * maintenance activities.
 *
 * May be null if 'vconn' doesn't have anything to do here. */
void (*run_wait)(struct vconn *vconn);

/* Arranges for the poll loop to wake up when 'vconn' is ready to take an
 * action of the given 'type'. */
void (*wait)(struct vconn *vconn, enum vconn_wait_type type);
```

}; ```

- **ovs-vswitchd side** **/* Process open flow message */ bridge_run(void)->bridge_run__(void)->ofproto_run(struct ofproto *p)->handle_openflow(struct ofconn *ofconn, const struct ovs_list *msgs) Note: for dump statistics, handle_openflow()->handle_single_part_openflow(ofconn, msg->data, type)->handle_port_stats_request()->handle_port_request()->... ->cb(port, &replies) == append_port_stat()**

About Statistic of open flow: ``` /* bridge_run()->bridge_reconfigure(), configure */ *bridge_reconfigure(const struct ovsrec_open_vswitch* ovs_cfg) { struct sockaddr_in *managers; struct bridge* br, *next; int sflow_bridge_number; size_t n_managers;

```
COVERAGE_INC(bridge_reconfigure);

ofproto_set_flow_limit(smap_get_int(&ovs_cfg->other_config, "flow-limit",
                                    OFPROTO_FLOW_LIMIT_DEFAULT));
ofproto_set_max_idle(smap_get_int(&ovs_cfg->other_config, "max-idle",
                                  OFPROTO_MAX_IDLE_DEFAULT));
ofproto_set_vlan_limit(smap_get_int(&ovs_cfg->other_config, "vlan-limit",
                                  LEGACY_MAX_VLAN_HEADERS));
ofproto_set_bundle_idle_timeout(smap_get_int(&ovs_cfg->other_config,
                                             "bundle-idle-timeout", 0));
ofproto_set_threads(
    smap_get_int(&ovs_cfg->other_config, "n-handler-threads", 0),
    smap_get_int(&ovs_cfg->other_config, "n-revalidator-threads", 0));

/* Destroy "struct bridge"s, "struct port"s, and "struct iface"s according
 * to 'ovs_cfg', with only very minimal configuration otherwise.
 *
 * This is mostly an update to bridge data structures. Nothing is pushed
 * down to ofproto or lower layers. */
add_del_bridges(ovs_cfg);
HMAP_FOR_EACH (br, node, &all_bridges) {
    bridge_collect_wanted_ports(br, &br->wanted_ports);
    bridge_del_ports(br, &br->wanted_ports);
}

/* Start pushing configuration changes down to the ofproto layer:
 *
 *   - Delete ofprotos that are no longer configured.
 *
```

```
*   - Delete ports that are no longer configured.
*
*   - Reconfigure existing ports to their desired configurations, or
*     delete them if not possible.
*
* We have to do all the deletions before we can do any additions, because
* the ports to be added might require resources that will be freed up by
* deletions (they might especially overlap in name). */
bridge_delete_ofprotos();
HMAP_FOR_EACH (br, node, &all_bridges) {
    if (br->ofproto) {
        bridge_delete_or_reconfigure_ports(br);
    }
}

/* Finish pushing configuration changes to the ofproto layer:
*
*     - Create ofprotos that are missing.
*
*     - Add ports that are missing. */
HMAP_FOR_EACH_SAFE (br, next, node, &all_bridges) {
    if (!br->ofproto) {
        int error;

        error = ofproto_create(br->name, br->type, &br->ofproto);   //ofproto look like is created
for every bridge
        if (error) {
            VLOG_ERR("failed to create bridge %s: %s", br->name,
                    ovs_strerror(error));
            shash_destroy(&br->wanted_ports);
            bridge_destroy(br, true);
        } else {
            /* Trigger storing datapath version. */
            seq_change(connectivity_seq_get());
        }
    }
}

config_ofproto_types(&ovs_cfg->other_config);

HMAP_FOR_EACH (br, node, &all_bridges) {
    bridge_add_ports(br, &br->wanted_ports);
    shash_destroy(&br->wanted_ports);
}

reconfigure_system_stats(ovs_cfg);

/* Complete the configuration. */
sflow_bridge_number = 0;
collect_in_band_managers(ovs_cfg, &managers, &n_managers);
HMAP_FOR_EACH (br, node, &all_bridges) {
    struct port *port;

    /* We need the datapath ID early to allow LACP ports to use it as the
    * default system ID. */
    bridge_configure_datapath_id(br);

    HMAP_FOR_EACH (port, hmap_node, &br->ports) {
        struct iface *iface;

        port_configure(port);

        LIST_FOR_EACH (iface, port_elem, &port->ifaces) {
            iface_set_ofport(iface->cfg, iface->ofp_port);
            /* Clear eventual previous errors */
            ovsrec_interface_set_error(iface->cfg, NULL);
            iface_configure_cfm(iface);
            iface_configure_qos(iface, port->cfg->qos);
            iface_set_mac(br, port, iface);
            ofproto_port_set_bfd(br->ofproto, iface->ofp_port,
                                 &iface->cfg->bfd);
```

```
            ofproto_port_set_lldp(br->ofproto, iface->ofp_port,
                                  &iface->cfg->lldp);
            ofproto_port_set_config(br->ofproto, iface->ofp_port,
                                    &iface->cfg->other_config);
        }
    }
    bridge_configure_mirrors(br);
    bridge_configure_forward_bpdu(br);
    bridge_configure_mac_table(br);
    bridge_configure_mcast_snooping(br);
    bridge_configure_remotes(br, managers, n_managers);
    bridge_configure_netflow(br);
    bridge_configure_sflow(br, &sflow_bridge_number);
    bridge_configure_ipfix(br);
    bridge_configure_spanning_tree(br);
    bridge_configure_tables(br);
    bridge_configure_dp_desc(br);
    bridge_configure_aa(br);
}
free(managers);

/* The ofproto-dpif provider does some final reconfiguration in its
 * ->type_run() function.  We have to call it before notifying the database
 * client that reconfiguration is complete, otherwise there is a very
 * narrow race window in which e.g. ofproto/trace will not recognize the
 * new configuration (sometimes this causes unit test failures). */
bridge_run__();
```

}

/* Create ofproto object: contain statistic (struct ofproto) & API of ofproto (struct ofproto)->(struct ofproto_class) */ *ofproto_create(const char* datapath_name, const char *datapath_type, struct ofproto ofprotop) OVS_EXCLUDED(ofproto_mutex) { const struct ofproto_class* class; struct ofproto *ofproto; int error; int i;

```
*ofprotop = NULL;
datapath_type = ofproto_normalize_type(datapath_type);
ALOG_INFO("ofproto_create()/datapath_name:%s - datapath_type:%s\n",datapath_name,datapath_type);
class = ofproto_class_find__(datapath_type);
if (!class) {
    VLOG_WARN("could not create datapath %s of unknown type %s",
            datapath_name, datapath_type);
    return EAFNOSUPPORT;
}

ofproto = class->alloc();
if (!ofproto) {
    VLOG_ERR("failed to allocate datapath %s of type %s",
            datapath_name, datapath_type);
    return ENOMEM;
}

/* Initialize. */
ovs_mutex_lock(&ofproto_mutex);
memset(ofproto, 0, sizeof *ofproto);
ofproto->ofproto_class = class; //Methods of ofproto
ofproto->name = xstrdup(datapath_name);
ofproto->type = xstrdup(datapath_type);
hmap_insert(&all_ofprotos, &ofproto->hmap_node,
            hash_string(ofproto->name, 0));
ofproto->datapath_id = 0;
ofproto->forward_bpdu = false;
ofproto->fallback_dpid = pick_fallback_dpid();
ofproto->mfr_desc = NULL;
ofproto->hw_desc = NULL;
ofproto->sw_desc = NULL;
ofproto->serial_desc = NULL;
ofproto->dp_desc = NULL;
ofproto->frag_handling = OFPUTIL_FRAG_NORMAL;
hmap_init(&ofproto->ports);
```

```
    hmap_init(&ofproto->ofport_usage);
    shash_init(&ofproto->port_by_name);
    simap_init(&ofproto->ofp_requests);
    ofproto->max_ports = ofp_to_u16(OFPP_MAX);
    ofproto->eviction_group_timer = LLONG_MIN;
    ofproto->tables = NULL;
    ofproto->n_tables = 0;
    ofproto->tables_version = OVS_VERSION_MIN;
    hindex_init(&ofproto->cookies);
    hmap_init(&ofproto->learned_cookies);
    ovs_list_init(&ofproto->expirable);
    ofproto->connmgr = connmgr_create(ofproto, datapath_name, datapath_name);
    ofproto->min_mtu = INT_MAX;
    cmap_init(&ofproto->groups);
    ovs_mutex_unlock(&ofproto_mutex);
    ofproto->ogf.types = 0xf;
    ofproto->ogf.capabilities = OFPGFC_CHAINING | OFPGFC_SELECT_LIVENESS |
                                OFPGFC_SELECT_WEIGHT;
    for (i = 0; i < 4; i++) {
        ofproto->ogf.max_groups[i] = OFPG_MAX;
        ofproto->ogf.ofpacts[i] = (UINT64_C(1) << N_OFPACTS) - 1;
    }
    ovsrcu_set(&ofproto->metadata_tab, tun_metadata_alloc(NULL));

    ovs_mutex_init(&ofproto->vl_mff_map.mutex);
    cmap_init(&ofproto->vl_mff_map.cmap);

    error = ofproto->ofproto_class->construct(ofproto);
    if (error) {
        VLOG_ERR("failed to open datapath %s: %s",
                 datapath_name, ovs_strerror(error));
        ovs_mutex_lock(&ofproto_mutex);
        connmgr_destroy(ofproto->connmgr);
        ofproto->connmgr = NULL;
        ovs_mutex_unlock(&ofproto_mutex);
        ofproto_destroy__(ofproto);
        return error;
    }

    /* Check that hidden tables, if any, are at the end. */
    ovs_assert(ofproto->n_tables);
    for (i = 0; i + 1 < ofproto->n_tables; i++) {
        enum oftable_flags flags = ofproto->tables[i].flags;
        enum oftable_flags next_flags = ofproto->tables[i + 1].flags;

        ovs_assert(!(flags & OFTABLE_HIDDEN) || next_flags & OFTABLE_HIDDEN);
    }

    ofproto->datapath_id = pick_datapath_id(ofproto);
    init_ports(ofproto);

    /* Initialize meters table. */
    if (ofproto->ofproto_class->meter_get_features) {
        ofproto->ofproto_class->meter_get_features(ofproto,
                                                   &ofproto->meter_features);
    } else {
        memset(&ofproto->meter_features, 0, sizeof ofproto->meter_features);
    }
    hmap_init(&ofproto->meters);
    ofproto->slowpath_meter_id = UINT32_MAX;
    ofproto->controller_meter_id = UINT32_MAX;

    /* Set the initial tables version. */
    ofproto_bump_tables_version(ofproto);

    *ofprotop = ofproto;
    return 0;
```

}

Note: in **struct ofproto** contain **struct hmap ports** , this one look like contain statistic of port ```

```
/* Get statistic of port */
netdev|INFO|[ATVN]netdev_get_stats()->(struct netdev)netdev->(struct netdev_class)netdev_class->get_stats()
netdev_dpdk|INFO|[ATVN](struct netdev_class)netdev_class->get_stats()==netdev_dpdk_vhost_get_stats()
//base on port type

Note: netdev|INFO|[ATVN]netdev_get_custom_stats()->(struct netdev)netdev->(struct netdev_class)netdev_class->get_custom_stats(): also support this API but until now, just dpdk or dpdkr support this one.
```

```
/* Update Source Code to prepare statistics data of TX package of dpdk port
__netdev_dpdk_vhost_send() ->__netdev_dpdk_vhost_send()->netdev_dpdk_vhost_update_tx_counters() */
netdev_dpdk_vhost_update_tx_counters(struct netdev_stats *stats, struct dp_packet **packets, int attempted, int dropped)
{
    int i,packet_size;
    int sent = attempted - dropped;

    stats->tx_packets += sent;
    stats->tx_dropped += dropped;

    for (i = 0; i < sent; i++) {
        packet_size = dp_packet_size(packets[i]);
        stats->tx_bytes += packet_size;
        /* ATVN NOTE: Update TX package here */
        if (packet_size < 256) {
            if (packet_size >= 128) {
                stats->tx_128_to_255_packets++;
            } else if (packet_size <= 64) {
                stats->tx_1_to_64_packets++;
            } else {
                stats->tx_65_to_127_packets++;
            }
        } else {
            if (packet_size >= 1523) {
                stats->tx_1523_to_max_packets++;
            } else if (packet_size >= 1024) {
                stats->tx_1024_to_1522_packets++;
            } else if (packet_size < 512) {
                stats->tx_256_to_511_packets++;
            } else {
                stats->tx_512_to_1023_packets++;
            }
        }
    }
}

/* Prepare data to replies (struct netdev_class)netdev_class->get_stats()==netdev_dpdk_vhost_get_stats() */
netdev_dpdk_vhost_get_stats(const struct netdev *netdev,
                            struct netdev_stats *stats)
{
    struct netdev_dpdk *dev = netdev_dpdk_cast(netdev);
    ALOG_INFO("(struct netdev_class)netdev_class->get_stats()==netdev_dpdk_vhost_get_stats()\n");
    ovs_mutex_lock(&dev->mutex);

    rte_spinlock_lock(&dev->stats_lock);
    /* Supported Stats */
    stats->rx_packets = dev->stats.rx_packets;
    stats->tx_packets = dev->stats.tx_packets;
    stats->rx_dropped = dev->stats.rx_dropped;
    stats->tx_dropped = dev->stats.tx_dropped;
    stats->multicast = dev->stats.multicast;
    stats->rx_bytes = dev->stats.rx_bytes;
    stats->tx_bytes = dev->stats.tx_bytes;
    stats->rx_errors = dev->stats.rx_errors;
    stats->rx_length_errors = dev->stats.rx_length_errors;
```

```
    stats->rx_1_to_64_packets = dev->stats.rx_1_to_64_packets;
    stats->rx_65_to_127_packets = dev->stats.rx_65_to_127_packets;
    stats->rx_128_to_255_packets = dev->stats.rx_128_to_255_packets;
    stats->rx_256_to_511_packets = dev->stats.rx_256_to_511_packets;
    stats->rx_512_to_1023_packets = dev->stats.rx_512_to_1023_packets;
    stats->rx_1024_to_1522_packets = dev->stats.rx_1024_to_1522_packets;
    stats->rx_1523_to_max_packets = dev->stats.rx_1523_to_max_packets;

    /* ATVN NOTE:get statistics information*/
    stats->tx_1_to_64_packets = dev->stats.tx_1_to_64_packets;
    stats->tx_65_to_127_packets = dev->stats.tx_65_to_127_packets;
    stats->tx_128_to_255_packets = dev->stats.tx_128_to_255_packets;
    stats->tx_256_to_511_packets = dev->stats.tx_256_to_511_packets;
    stats->tx_512_to_1023_packets = dev->stats.tx_512_to_1023_packets;
    stats->tx_1024_to_1522_packets = dev->stats.tx_1024_to_1522_packets;
    stats->tx_1523_to_max_packets = dev->stats.tx_1523_to_max_packets;

    rte_spinlock_unlock(&dev->stats_lock);

    ovs_mutex_unlock(&dev->mutex);

    return 0;
}
```

```
/* Prepare replies message for ovs-ofctl : append_port_stat()->ofputil_append_port_stat() */
void
ofputil_append_port_stat(struct ovs_list *replies,
                         const struct ofputil_port_stats *ops)
{
    ALOG_INFO("ofputil_append_port_stat()/ofpmp_version(replies):%d\n",ofpmp_version(replies));
    switch (ofpmp_version(replies)) {
    case OFP13_VERSION: {
        struct ofp13_port_stats *reply = ofpmp_append(replies, sizeof *reply);
        ofputil_port_stats_to_ofp13(ops, reply);
        break;
    }
    case OFP12_VERSION:
    case OFP11_VERSION: {
        struct ofp11_port_stats *reply = ofpmp_append(replies, sizeof *reply);
        ofputil_port_stats_to_ofp11(ops, reply);
        break;
    }

    case OFP10_VERSION: {
        struct ofp10_port_stats *reply = ofpmp_append(replies, sizeof *reply);
        ofputil_port_stats_to_ofp10(ops, reply);
        break;
    }

    case OFP14_VERSION:
    case OFP15_VERSION:
    case OFP16_VERSION:
        ofputil_append_ofp14_port_stats(ops, replies);
        break;

    default:
        OVS_NOT_REACHED();
    }
}

Note: base on OFPXX_VERSION, the replies message will difference:

`$ sudo ovs-ofctl --protocols=OpenFlow14 dump-ports ovs-dpdk-br0 ovs-br0-vport2`
`$ sudo ovs-ofctl --protocols=OpenFlow14 dump-ports ovs-dpdk-br0 ovs-br0-pport2`
```

1.3.7. Open Flow classifier (OVS + DPDK)

1.3.7.1. Open Flow Classifies command

Example:

`$ sudo ovs-ofctl add-flow ovs-dpdk-br0 cookie=0x11,in_port="ovs-br0-vport1",actions=set_queue:123,output:ovs-br0-vport2`

`$ sudo ovs-ofctl add-flow ovs-dpdk-br0 cookie=0x12,in_port="ovs-br0-vport2",actions=set_queue:234,normal`

1.3.7.2. Classifies Results

After classifies, the queue id will be updated to '''uint32_t skb_priority''' **in** '''struct pkt_metadata''' of '''struct dp_packet'''

The source code:

```
static int
netdev_dpdk_vhost_send(struct netdev *netdev, int qid, struct dp_packet_batch *batch,bool concurrent_txq OVS_UNUSED)

{

    if (OVS_UNLIKELY(batch->packets[0]->source != DPBUF_DPDK)) {

        dpdk_do_tx_copy(netdev, qid, batch);

        dp_packet_delete_batch(batch, true);

    } else {

#define ATVN_DEBUG

#ifdef ATVN_DEBUG  //just for debug

        int i = 0;

        struct dp_packet * package;

        for(i = 0; i < batch->count; i++ ){

            package = batch->packets[i];

            ALOG_INFO("netdev_dpdk_vhost_send/dp_packet/md/skb_priority: %d\n",package->md.skb_priority);

        }

#endif

        __netdev_dpdk_vhost_send(netdev, qid, batch->packets, batch->count);

    }

    return 0;

}
```

1.3.8. Configuration Command

Configuration HQoS command, **Refer** to __OVS **Commands/HQoS Configuration** (**Defined by** ATVN)__

1.3.9. Egress HQoS 2 level policer Update configuration testing

Environment Testing

```
+ Testing on SERVER 9
+ MAX Queues: 4000
+ Testing with ATVN_DEBUG flag to known detail of data structures
```

Note: below configuration command is used for “update configuration testing”

```
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- \
--id=@newqos create qos type=egress-hqos2l-policer other-config:cir=2048 other-config:cbs=2048 other-
config:pir=31457280 other-config:pbs=2048  other-config:ydropt=true \
qos=@qospl11,@qospl12,@qospl13,@qospl14,@qospl15,@qospl16,@qospl17,@qospl18,@qospl19,@qospl1a -- \
--id=@qospl11 create qos type=egress-policer-rfc2698 other-config:cir=8192 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:1=@queue1 -- \
--id=@qospl12 create qos type=egress-policer-rfc2698 other-config:cir=2048 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:2=@queue2 -- \
--id=@qospl13 create qos type=egress-policer-rfc2698 other-config:cir=4096 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:4=@queue4 -- \
--id=@qospl14 create qos type=egress-policer-rfc2698 other-config:cir=8192 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:8=@queue8 -- \
--id=@qospl15 create qos type=egress-policer-rfc2698 other-config:cir=8192 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:4001=@queue1 -- \
--id=@qospl16 create qos type=egress-policer-rfc2698 other-config:cir=2048 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:4002=@queue2 -- \
--id=@qospl17 create qos type=egress-policer-rfc2698 other-config:cir=4096 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:4004=@queue4 -- \
--id=@qospl18 create qos type=egress-policer-rfc2698 other-config:cir=8192 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:4008=@queue8 -- \
--id=@qospl19 create qos type=egress-policer-rfc2698 other-config:cir=4096 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:4000=@queue4 -- \
--id=@qospl1a create qos type=egress-policer-rfc2698 other-config:cir=8192 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:8000=@queue8 -- \
--id=@queue1 create queue other-config:desc=rfu -- \
--id=@queue2 create queue other-config:desc=rfu -- \
--id=@queue4 create queue other-config:desc=rfu -- \
--id=@queue8 create queue other-config:desc=rfu`

#To list out all queues, please use command `$ sudo ovs-appctl qos/show-queue-stats ovs-br0-vport2`.
Example of this testing:
`$ sudo ovs-appctl qos/show-queue-stats ovs-br0-vport2`
Queues's Statistics of interface ovs-br0-vport2 (10 queues):
Queue 4008: tx_packets:0, tx_bytes:0,tx_dropped:0, tx_errors:0, duration=-9223372036854775808,
        tx_1_to_64_packets:0, tx_65_to_127_packets:0, tx_128_to_255_packets:0,
        tx_256_to_511_packets:0,tx_512_to_1023_packets:0, tx_1024_to_1522_packets:0,
tx_1523_to_max_packets:0.
Queue 8000: tx_packets:0, tx_bytes:0,tx_dropped:0, tx_errors:0, duration=-9223372036854775808,
        tx_1_to_64_packets:0, tx_65_to_127_packets:0, tx_128_to_255_packets:0,
        tx_256_to_511_packets:0,tx_512_to_1023_packets:0, tx_1024_to_1522_packets:0,
tx_1523_to_max_packets:0.
Queue 2: tx_packets:0, tx_bytes:0,tx_dropped:0, tx_errors:0, duration=-9223372036854775808,
        tx_1_to_64_packets:0, tx_65_to_127_packets:0, tx_128_to_255_packets:0,
        tx_256_to_511_packets:0,tx_512_to_1023_packets:0, tx_1024_to_1522_packets:0,
tx_1523_to_max_packets:0.
Queue 4001: tx_packets:0, tx_bytes:0,tx_dropped:0, tx_errors:0, duration=-9223372036854775808,
        tx_1_to_64_packets:0, tx_65_to_127_packets:0, tx_128_to_255_packets:0,
        tx_256_to_511_packets:0,tx_512_to_1023_packets:0, tx_1024_to_1522_packets:0,
```

```
tx_1523_to_max_packets:0.
Queue 4002: tx_packets:0, tx_bytes:0,tx_dropped:0, tx_errors:0, duration=-9223372036854775808,
        tx_1_to_64_packets:0, tx_65_to_127_packets:0, tx_128_to_255_packets:0,
        tx_256_to_511_packets:0,tx_512_to_1023_packets:0, tx_1024_to_1522_packets:0,
tx_1523_to_max_packets:0.
Queue 8: tx_packets:0, tx_bytes:0,tx_dropped:0, tx_errors:0, duration=-9223372036854775808,
        tx_1_to_64_packets:0, tx_65_to_127_packets:0, tx_128_to_255_packets:0,
        tx_256_to_511_packets:0,tx_512_to_1023_packets:0, tx_1024_to_1522_packets:0,
tx_1523_to_max_packets:0.
Queue 4004: tx_packets:0, tx_bytes:0,tx_dropped:0, tx_errors:0, duration=-9223372036854775808,
        tx_1_to_64_packets:0, tx_65_to_127_packets:0, tx_128_to_255_packets:0,
        tx_256_to_511_packets:0,tx_512_to_1023_packets:0, tx_1024_to_1522_packets:0,
tx_1523_to_max_packets:0.
Queue 1: tx_packets:0, tx_bytes:0,tx_dropped:0, tx_errors:0, duration=-9223372036854775808,
        tx_1_to_64_packets:0, tx_65_to_127_packets:0, tx_128_to_255_packets:0,
        tx_256_to_511_packets:0,tx_512_to_1023_packets:0, tx_1024_to_1522_packets:0,
tx_1523_to_max_packets:0.
Queue 4: tx_packets:0, tx_bytes:0,tx_dropped:0, tx_errors:0, duration=-9223372036854775808,
        tx_1_to_64_packets:0, tx_65_to_127_packets:0, tx_128_to_255_packets:0,
        tx_256_to_511_packets:0,tx_512_to_1023_packets:0, tx_1024_to_1522_packets:0,
tx_1523_to_max_packets:0.
Queue 4000: tx_packets:41, tx_bytes:3906,tx_dropped:0, tx_errors:0, duration=-9223372036854775808,
        tx_1_to_64_packets:2, tx_65_to_127_packets:39, tx_128_to_255_packets:0,
        tx_256_to_511_packets:0,tx_512_to_1023_packets:0, tx_1024_to_1522_packets:0,
tx_1523_to_max_packets:0.

#Result: base on information from, the array of all queues is
Index:  |   0   |   1   |   2   |   3   |   4   |   5   |   6   |   7   |   8   |   9   |   10  |
11  |...
Value:  |8000    |4001  |2      |1      |4004   |4      |4000   |4002   |4008   |8      |       |
|... (Configure result)
```

1.3.9.1. Test Case 1

Description

```
+ Remove queue 8000 in HQoS configuration
+ Queue 8000 with uuid: 9594e96c-c21e-4544-8f07-96b048876f12 (should check by yourself: `$ sudo ovs-
vsctl list qos`)
+ Root Node HQoS's uuid: 5048827e-93c1-4263-8ffd-c3c86d5aef0d
```

Execute remove command

```
`$ sudo ovs-vsctl remove qos 5048827e-93c1-4263-8ffd-c3c86d5aef0d qos 9594e96c-c21e-4544-8f07-
96b048876f12`
```

Result

```
Index:      |   0   |   1   |   2   |   3   |   4   |   5   |   6   |   7   |   8   |   9   |
10  |   11  |...
expected:   |4000   |4001   |2      |1      |4004   |4      |4002   |       |4008   |8      |
|       |...
Result:     |4000   |4001   |2      |1      |4004   |4      |4002   |       |4008   |8      |
|       |... (OK)
```

1.3.9.2. Test Case 2

Description

```
+ add queue 8000 in HQoS configuration
+ Queue 8000 with uuid: 9594e96c-c21e-4544-8f07-96b048876f12 (should check by yourself: `$ sudo ovs-
vsctl list qos`)
+ Root Node HQoS's uuid: 5048827e-93c1-4263-8ffd-c3c86d5aef0d
```

Execute remove command

```
`$ sudo ovs-vsctl add qos 5048827e-93c1-4263-8ffd-c3c86d5aef0d qos 9594e96c-c21e-4544-8f07-
96b048876f12`
```

Result

```
Index:      |   0   |   1   |   2   |   3   |   4   |   5   |   6   |   7   |   8   |   9   |
10  |   11  |...
expected:   |4000   |4001   |2      |1      |4004   |4      |4002   |8000   |4008   |8      |
|       |...
Result:     |4000   |4001   |2      |1      |4004   |4      |4002   |8000   |4008   |8      |
|       |... (OK)
```

1.3.9.3. Test Case 3

Description

```
+ Root Node HQoS's uuid: 5048827e-93c1-4263-8ffd-c3c86d5aef0d
+ remove queue 4000 in HQoS configuration
+ Queue 4000 with uuid: d7889d58-4d5e-4fd7-aad7-7522bdb6ab43 (should check by yourself: `$ sudo ovs-
vsctl list qos`)
+ create && add queue 8004 to HQoS
+ Add queue 4000 with uuid: d7889d58-4d5e-4fd7-aad7-7522bdb6ab43
+ create && add queue 8005 to HQoS
```

Execute command

```
#Remove QoS
`$ sudo ovs-vsctl remove qos 5048827e-93c1-4263-8ffd-c3c86d5aef0d qos d7889d58-4d5e-4fd7-aad7-
7522bdb6ab43`

#Create a qos leaf (Queue 8004)
`$ sudo ovs-vsctl create qos type=egress-policer-rfc2698 other-config:cir=8192 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:8004=@queue118 --  --id=@queue118 create queue other-config:desc=rfu`
36c2e054-d906-49db-8991-b0f0e801ea63    /*QoS node that have just created */
b052f06c-92bf-4011-b8e4-2706107a2ba6

#Add new QoS node to HQoS tree
`$ sudo ovs-vsctl add qos 5048827e-93c1-4263-8ffd-c3c86d5aef0d qos 36c2e054-d906-49db-8991-
b0f0e801ea63`

#Add QoS node 4000 to HQoS tree
`$ sudo ovs-vsctl add qos 5048827e-93c1-4263-8ffd-c3c86d5aef0d qos d7889d58-4d5e-4fd7-aad7-
7522bdb6ab43`

#Create a qos leaf (Queue 8005)
`$ sudo ovs-vsctl create qos type=egress-policer-rfc2698 other-config:cir=8192 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:8005=@queue11 --  --id=@queue119 create queue other-config:desc=rfu`
dc0e2fa4-3e3b-4230-bc6b-d82085f5ce8b    /*QoS node that have just created */
3e7780e9-aeb4-4d50-859a-eefb5348ff12

#Add QoS node 8005 to HQoS tree
`$ sudo ovs-vsctl add qos 5048827e-93c1-4263-8ffd-c3c86d5aef0d qos dc0e2fa4-3e3b-4230-bc6b-
d82085f5ce8b`
```

Result

```
Index:      |   0   |   1   |   2   |   3   |   4   |   5   |   6   |   7   |   8   |   9   |
10  |   11  |...
expected:   |8000   |4001   |2      |1      |4004   |4      |4002   |       |4008   |8      |
```

```
|       |... (remove queue 4)
Result:     |8000   |4001   |2      |1      |4004   |4      |4002   |       |4008   |8      |
|       |... (OK)
expected:   |8000   |4001   |2      |1      |4004   |4      |4002   |8004   |4008   |8      |
|       |... (Add qos node 8004)
Result:     |8000   |4001   |2      |1      |4004   |4      |4002   |8004   |4008   |8      |
|       |... (OK)
expected:   |8000   |4001   |2      |1      |4004   |4      |4002   |8004   |4008   |8      |4000
|       |... (Add QoS node 4000)
Result:     |8000   |4001   |2      |1      |4004   |4      |4002   |8004   |4008   |8      |4000
|       |... (OK)
expected:   |8000   |4001   |2      |1      |4004   |4      |4002   |8004   |4008   |8      |4000
|8005   |... (Add qos node 8005)
Result:     |8000   |4001   |2      |1      |4004   |4      |4002   |8004   |4008   |8      |4000
|8005   |... (OK)

> Note: checked POLICER feature is OK after all configuration. refer to __${This Task}/Egress HQoS
2 level policer Testing__ for more information about how to test it.
```

1.3.9.4. Test Case 4

Description

```
+ Update Port Node configuration
+ Port Node UUID: 5048827e-93c1-4263-8ffd-c3c86d5aef0d
+ Current Status: {cbs="2048", cir="2048", pbs="2048", pir="31457280", ydropt="true"}
+ Update it's cir to 4096
```

Execute command

```
`$ sudo ovs-vsctl set qos 5048827e-93c1-4263-8ffd-c3c86d5aef0d other_config:cir=4096`
```

Result

```
Tested with iperf. it is OK. refer to __${This Task}/Egress HQoS 2 level policer Testing__ for more
information about how to test it.
```

1.3.10. Egress HQoS 2 level policer Testing

how to configure & testing with iperf ? Please refer to **${This documents}/Testing tools/iperf** documents

Environment Testing

```
+ Testing on SERVER 9
+ VM1,VM2 & VM3: RAM:1G, Number of core: 1
```

Note: below configuration command is used for test case from 1 to 3

```
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- \
--id=@newqos create qos type=egress-hqos2l-policer other-config:cir=0 other-config:cbs=2048 other-
config:pir=31457280 other-config:pbs=2048  other-config:ydropt=true \
qos=@qospl11,@qospl12,@qospl13,@qospl14 -- \
--id=@qospl11 create qos type=egress-policer-rfc2698 other-config:cir=1024 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:1=@queue1 -- \
--id=@qospl12 create qos type=egress-policer-rfc2698 other-config:cir=2048 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:2=@queue2 -- \
--id=@qospl13 create qos type=egress-policer-rfc2698 other-config:cir=4096 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:4=@queue4 -- \
--id=@qospl14 create qos type=egress-policer-rfc2698 other-config:cir=8192 other-config:cbs=2048
```

```
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:8=@queue8 -- \
--id=@queue1 create queue other-config:desc=rfu -- \
--id=@queue2 create queue other-config:desc=rfu -- \
--id=@queue4 create queue other-config:desc=rfu -- \
--id=@queue8 create queue other-config:desc=rfu`
```

1.3.10.1. Test Case 1 (From 1 to 8: Basic testing for Policer Queue (Port's CIR is 0)

Description

```
+ generate speed: 15 kbits/s
+ Configuration:
> Port: cir = 0 (unlimited)
> Queue: cir = 32 kbits/s & Yelow color is dropted (details: cbs="2048", cir="4096", pbs="2048",
pir="10240", ydropt="true")

+ Expected bandwidth: 15 kbits/s
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2: refer to __${This Task}/Note: below configuration command is used
for test case from 1 to 3__
# Using Queue 4 by configure openflow: `$ sudo ovs-ofctl add-flow ovs-dpdk-br0
cookie=0x14,in_port="ovs-br0-vport1",actions=set_queue:4,output:ovs-br0-vport2`

# Check the result command: `$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2`
```

Test result

```
# Start iperf server (On VM2 - ovs-br0-vport2 port)
$ iperf -s -u -p 8080

# Start iperf client
$ iperf -c 192.168.2.12 -u -p 8080 -b 15k

Server Report (Tested with 3 times):
1. Transfer: 20.1 KBytes & Bandwidth: 15.0 Kbits/s
2. Transfer: 20.1 KBytes & Bandwidth: 15.0 Kbits/s
3. Transfer: 20.1 KBytes & Bandwidth: 15.0 Kbits/s
```

1.3.10.2. Test Case 2

Description

```
+ generate speed: 32 kbits/s
+ Configuration:
> Port: cir = 0 (unlimited)
> Queue: cir = 32 kbits/s & Yelow color is dropted (details: cbs="2048", cir="4096", pbs="2048",
pir="10240", ydropt="true")

+ Expected bandwidth: 32 kbits/s
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2: refer to __${This Task}/Note: below configuration command is used
for test case from 1 to 3__
# Using Queue 4 by configure openflow: `$ sudo ovs-ofctl add-flow ovs-dpdk-br0
cookie=0x14,in_port="ovs-br0-vport1",actions=set_queue:4,output:ovs-br0-vport2`

# Check the result command: `$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2`
```

Test result

```
# Start iperf server (On VM2 - ovs-br0-vport2 port)
$ iperf -s -u -p 8080

# Start iperf client
$ iperf -c 192.168.2.12 -u -p 8080 -b 32k

Server Report (Tested with 3 times):
1. Transfer: 41.6 KBytes & Bandwidth: 32.0 Kbits/s
2. Transfer: 41.6 KBytes & Bandwidth: 32.0 Kbits/s
3. Transfer: 41.6 KBytes & Bandwidth: 32.0 Kbits/s
```

1.3.10.3. Test Case 3

Description

```
+ generate speed: 1000 kbits/s
+ Configuration:
> Port: cir = 0 (unlimited)
> Queue: cir = 32 kbits/s & Yelow color is dropted (details: cbs="2048", cir="4096", pbs="2048",
pir="10240", ydropt="true")

+ Expected bandwidth: 32 kbits/s
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2: refer to __${This Task}/Note: below configuration command is used
for test case from 1 to 3__
# Using Queue 4 by configure openflow: `$ sudo ovs-ofctl add-flow ovs-dpdk-br0
cookie=0x14,in_port="ovs-br0-vport1",actions=set_queue:4,output:ovs-br0-vport2`

# Check the result command: `$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2`
```

Test result

```
# Start iperf server (On VM2 - ovs-br0-vport2 port)
$ iperf -s -u -p 8080

# Start iperf client
$ iperf -c 192.168.2.12 -u -p 8080 -b 1m

Server Report (Tested with 3 times):
1. Transfer: 40.2 KBytes & Bandwidth: 32.1 Kbits/s
2. Transfer: 40.2 KBytes & Bandwidth: 32.1 Kbits/s
3. Transfer: 40.2 KBytes & Bandwidth: 32.1 Kbits/s
```

Note: below configuration command is used for test case from 4 to 8

```
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- \
--id=@newqos create qos type=egress-hqos2l-policer other-config:cir=0 other-config:cbs=2048 other-
config:pir=31457280 other-config:pbs=2048  other-config:ydropt=false \
qos=@qospl11,@qospl12,@qospl13,@qospl14 -- \
--id=@qospl11 create qos type=egress-policer-rfc2698 other-config:cir=1024 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=false \
queues:1=@queue1 -- \
--id=@qospl12 create qos type=egress-policer-rfc2698 other-config:cir=2048 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=false \
queues:2=@queue2 -- \
--id=@qospl13 create qos type=egress-policer-rfc2698 other-config:cir=4096 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=false \
queues:4=@queue4 -- \
--id=@qospl14 create qos type=egress-policer-rfc2698 other-config:cir=8192 other-config:cbs=2048
```

```
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=false \
queues:8=@queue8 -- \
--id=@queue1 create queue other-config:desc=rfu -- \
--id=@queue2 create queue other-config:desc=rfu -- \
--id=@queue4 create queue other-config:desc=rfu -- \
--id=@queue8 create queue other-config:desc=rfu`
```

1.3.10.4. Test Case 4

Description

```
+ generate speed: 20 kbits/s
+ Configuration:
> Port: cir = 0 (unlimited)
> Queue: cir = 32 kbits/s, pir = 80 kbit/s & Yelow color is sent
(details: cbs="2048", cir="4096", pbs="2048", pir="10240", ydropt="false")

+ Expected bandwidth: 20 kbits/s
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2: refer to __${This Task}/Note: below configuration command is used
for test case from 4 to 8__
# Using Queue 4 by configure openflow: `$ sudo ovs-ofctl add-flow ovs-dpdk-br0
cookie=0x14,in_port="ovs-br0-vport1",actions=set_queue:4,output:ovs-br0-vport2`

# Check the result command: `$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2`
```

Test result

```
# Start iperf server (On VM2 - ovs-br0-vport2 port)
$ iperf -s -u -p 8080

# Start iperf client
$ iperf -c 192.168.2.12 -u -p 8080 -b 20k

Server Report (Tested with 3 times):
1. Transfer: 27.3 KBytes & Bandwidth: 20.0 Kbits/s
2. Transfer: 27.3 KBytes & Bandwidth: 20.0 Kbits/s
3. Transfer: 27.3 KBytes & Bandwidth: 20.0 Kbits/s
```

1.3.10.5. Test Case 5

Description

```
+ generate speed: 32 kbits/s
+ Configuration:
> Port: cir = 0 (unlimited)
> Queue: cir = 32 kbits/s, pir = 80 kbit/s & Yelow color is sent
(details: cbs="2048", cir="4096", pbs="2048", pir="10240", ydropt="false")

+ Expected bandwidth: 32 kbits/s
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2: refer to __${This Task}/Note: below configuration command is used
for test case from 4 to 8__
# Using Queue 4 by configure openflow: `$ sudo ovs-ofctl add-flow ovs-dpdk-br0
cookie=0x14,in_port="ovs-br0-vport1",actions=set_queue:4,output:ovs-br0-vport2`

# Check the result command: `$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2`
```

Test result

```
# Start iperf server (On VM2 - ovs-br0-vport2 port)
$ iperf -s -u -p 8080

# Start iperf client
$ iperf -c 192.168.2.12 -u -p 8080 -b 32k

Server Report (Tested with 3 times):
1. Transfer: 41.6 KBytes & Bandwidth: 32.0 Kbits/s
2. Transfer: 41.6 KBytes & Bandwidth: 32.0 Kbits/s
3. Transfer: 41.6 KBytes & Bandwidth: 32.0 Kbits/s
```

1.3.10.6. Test Case 6

Description

```
+ generate speed: 60 kbits/s
+ Configuration:
> Port: cir = 0 (unlimited)
> Queue: cir = 32 kbits/s, pir = 80 kbit/s & Yelow color is sent
(details: cbs="2048", cir="4096", pbs="2048", pir="10240", ydropt="false")

+ Expected bandwidth: 60 kbits/s
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2: refer to __${This Task}/Note: below configuration command is used
for test case from 4 to 8__
# Using Queue 4 by configure openflow: `$ sudo ovs-ofctl add-flow ovs-dpdk-br0
cookie=0x14,in_port="ovs-br0-vport1",actions=set_queue:4,output:ovs-br0-vport2`

# Check the result command: `$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2`
```

Test result

```
# Start iperf server (On VM2 - ovs-br0-vport2 port)
$ iperf -s -u -p 8080

# Start iperf client
$ iperf -c 192.168.2.12 -u -p 8080 -b 60k

Server Report (Tested with 3 times):
1. Transfer: 76.1 KBytes & Bandwidth: 60.0 Kbits/s
2. Transfer: 76.1 KBytes & Bandwidth: 60.0 Kbits/s
3. Transfer: 76.1 KBytes & Bandwidth: 60.0 Kbits/s
```

1.3.10.7. Test Case 7

Description

```
+ generate speed: 80 kbits/s
+ Configuration:
> Port: cir = 0 (unlimited)
> Queue: cir = 32 kbits/s, pir = 80 kbit/s & Yelow color is sent
(details: cbs="2048", cir="4096", pbs="2048", pir="10240", ydropt="false")

+ Expected bandwidth: 80 kbits/s
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2: refer to __${This Task}/Note: below configuration command is used
```

```
for test case from 4 to 8__
# Using Queue 4 by configure openflow: `$ sudo ovs-ofctl add-flow ovs-dpdk-br0
cookie=0x14,in_port="ovs-br0-vport1",actions=set_queue:4,output:ovs-br0-vport2`

# Check the result command: `$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2`
```

Test result

```
# Start iperf server (On VM2 - ovs-br0-vport2 port)
$ iperf -s -u -p 8080

# Start iperf client
$ iperf -c 192.168.2.12 -u -p 8080 -b 80k

Server Report (Tested with 3 times):
1. Transfer: 100.0 KBytes & Bandwidth: 80.0 Kbits/s
2. Transfer: 100.0 KBytes & Bandwidth: 80.0 Kbits/s
3. Transfer: 100.0 KBytes & Bandwidth: 80.0 Kbits/s
```

1.3.10.8. Test Case 8

Description

```
+ generate speed: 1000 kbits/s
+ Configuration:
> Port: cir = 0 (unlimited)
> Queue: cir = 32 kbits/s, pir = 80 kbit/s & Yelow color is sent
(details: cbs="2048", cir="4096", pbs="2048", pir="10240", ydropt="false")

+ Expected bandwidth: 80 kbits/s
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2: refer to __${This Task}/Note: below configuration command is used
for test case from 4 to 8__
# Using Queue 4 by configure openflow: `$ sudo ovs-ofctl add-flow ovs-dpdk-br0
cookie=0x14,in_port="ovs-br0-vport1",actions=set_queue:4,output:ovs-br0-vport2`

# Check the result command: `$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2`
```

Test result

```
# Start iperf server (On VM2 - ovs-br0-vport2 port)
$ iperf -s -u -p 8080

# Start iperf client
$ iperf -c 192.168.2.12 -u -p 8080 -b 1m

Server Report (Tested with 3 times):
1. Transfer: 99.1 KBytes & Bandwidth: 79.0 Kbits/s
2. Transfer: 99.1 KBytes & Bandwidth: 79.0 Kbits/s
3. Transfer: 99.1 KBytes & Bandwidth: 79.0 Kbits/s
```

Note: below configuration command is used for test case from 9 to 11

```
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- \
--id=@newqos create qos type=egress-hqos2l-policer other-config:cir=4096 other-config:cbs=2048 other-
config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
qos=@qospl11,@qospl12,@qospl13,@qospl14 -- \
--id=@qospl11 create qos type=egress-policer-rfc2698 other-config:cir=1024 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:1=@queue1 -- \
```

```
--id=@qospl12 create qos type=egress-policer-rfc2698 other-config:cir=2048 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:2=@queue2 -- \
--id=@qospl13 create qos type=egress-policer-rfc2698 other-config:cir=4096 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:4=@queue4 -- \
--id=@qospl14 create qos type=egress-policer-rfc2698 other-config:cir=8192 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:8=@queue8 -- \
--id=@queue1 create queue other-config:desc=rfu -- \
--id=@queue2 create queue other-config:desc=rfu -- \
--id=@queue4 create queue other-config:desc=rfu -- \
--id=@queue8 create queue other-config:desc=rfu`
```

1.3.10.9. Test Case 9 (From 9 to 16: Basic testing for Policer Port (Use Queue default - Not configure for any queue)

Description

```
+ generate speed: 15 kbits/s
+ Configuration:
> Port: cir = 32 kbits/s & Yelow color is dropted (details: cbs="2048", cir="4096", pbs="2048",
pir="10240", ydropt="true")

+ Expected bandwidth: 15 kbits/s
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2: refer to __${This Task}/Note: below configuration command is used
for test case from 1 to 3__
# Delete open flow to use default queue: `$ sudo ovs-ofctl del-flows ovs-dpdk-br0 cookie=0x14/-1`

# Check the result command: `$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2`
```

Test result

```
# Start iperf server (On VM2 - ovs-br0-vport2 port)
$ iperf -s -u -p 8080

# Start iperf client
$ iperf -c 192.168.2.12 -u -p 8080 -b 15k

Server Report (Tested with 3 times):
1. Transfer: 20.1 KBytes & Bandwidth: 15.0 Kbits/s
2. Transfer: 20.1 KBytes & Bandwidth: 15.0 Kbits/s
3. Transfer: 20.1 KBytes & Bandwidth: 15.0 Kbits/s
```

1.3.10.10. Test Case 10

Description

```
+ generate speed: 32 kbits/s
+ Configuration:
> Port: cir = 32 kbits/s & Yelow color is dropted (details: cbs="2048", cir="4096", pbs="2048",
pir="10240", ydropt="true")

+ Expected bandwidth: 32 kbits/s
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2: refer to __${This Task}/Note: below configuration command is used
for test case from 1 to 3__
```

```
# Delete open flow to use default queue: `$ sudo ovs-ofctl del-flows ovs-dpdk-br0 cookie=0x14/-1`

# Check the result command: `$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2`
```

Test result

```
# Start iperf server (On VM2 - ovs-br0-vport2 port)
$ iperf -s -u -p 8080

# Start iperf client
$ iperf -c 192.168.2.12 -u -p 8080 -b 32k

Server Report (Tested with 3 times):
1. Transfer: 41.6 KBytes & Bandwidth: 32.0 Kbits/s
2. Transfer: 41.6 KBytes & Bandwidth: 32.0 Kbits/s
3. Transfer: 41.6 KBytes & Bandwidth: 32.0 Kbits/s
```

1.3.10.11. Test Case 11

Description

```
+ generate speed: 1000 kbits/s
+ Configuration:
> Port: cir = 32 kbits/s & Yelow color is dropted (details: cbs="2048", cir="4096", pbs="2048",
pir="10240", ydropt="true")

+ Expected bandwidth: 32 kbits/s
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2: refer to __${This Task}/Note: below configuration command is used
for test case from 1 to 3__
# Delete open flow to use default queue: `$ sudo ovs-ofctl del-flows ovs-dpdk-br0 cookie=0x14/-1`

# Check the result command: `$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2`
```

Test result

```
# Start iperf server (On VM2 - ovs-br0-vport2 port)
$ iperf -s -u -p 8080

# Start iperf client
$ iperf -c 192.168.2.12 -u -p 8080 -b 1m

Server Report (Tested with 3 times):
1. Transfer: 40.2 KBytes & Bandwidth: 32.1 Kbits/s
2. Transfer: 40.2 KBytes & Bandwidth: 32.1 Kbits/s
3. Transfer: 40.2 KBytes & Bandwidth: 32.1 Kbits/s
```

Note: below configuration command is used for test case from 12 to 16

```
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- \
--id=@newqos create qos type=egress-hqos2l-policer other-config:cir=4096 other-config:cbs=2048 other-
config:pir=10240 other-config:pbs=2048  other-config:ydropt=false \
qos=@qospl11,@qospl12,@qospl13,@qospl14 -- \
--id=@qospl11 create qos type=egress-policer-rfc2698 other-config:cir=1024 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:1=@queue1 -- \
--id=@qospl12 create qos type=egress-policer-rfc2698 other-config:cir=2048 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:2=@queue2 -- \
--id=@qospl13 create qos type=egress-policer-rfc2698 other-config:cir=4096 other-config:cbs=2048
```

```
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:4=@queue4 -- \
--id=@qospl14 create qos type=egress-policer-rfc2698 other-config:cir=8192 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:8=@queue8 -- \
--id=@queue1 create queue other-config:desc=rfu -- \
--id=@queue2 create queue other-config:desc=rfu -- \
--id=@queue4 create queue other-config:desc=rfu -- \
--id=@queue8 create queue other-config:desc=rfu`
```

1.3.10.12. Test Case 12

Description

```
+ generate speed: 20 kbits/s
+ Configuration:
> Port: cir = 32 kbits/s, pir = 80 kbit/s & Yelow color is sent
(details: cbs="2048", cir="4096", pbs="2048", pir="10240", ydropt="false")

+ Expected bandwidth: 20 kbits/s
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2: refer to __${This Task}/Note: below configuration command is used
for test case from 4 to 8__
# Delete open flow to use default queue: `$ sudo ovs-ofctl del-flows ovs-dpdk-br0 cookie=0x14/-1`

# Check the result command: `$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2`
```

Test result

```
# Start iperf server (On VM2 - ovs-br0-vport2 port)
$ iperf -s -u -p 8080

# Start iperf client
$ iperf -c 192.168.2.12 -u -p 8080 -b 20k

Server Report (Tested with 3 times):
1. Transfer: 27.3 KBytes & Bandwidth: 20.0 Kbits/s
2. Transfer: 27.3 KBytes & Bandwidth: 20.0 Kbits/s
3. Transfer: 27.3 KBytes & Bandwidth: 20.0 Kbits/s
```

1.3.10.13. Test Case 13

Description

```
+ generate speed: 32 kbits/s
+ Configuration:
> Port: cir = 32 kbits/s, pir = 80 kbit/s & Yelow color is sent
(details: cbs="2048", cir="4096", pbs="2048", pir="10240", ydropt="false")

+ Expected bandwidth: 32 kbits/s
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2: refer to __${This Task}/Note: below configuration command is used
for test case from 4 to 8__
# Delete open flow to use default queue: `$ sudo ovs-ofctl del-flows ovs-dpdk-br0 cookie=0x14/-1`

# Check the result command: `$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2`
```

Test result

```
# Start iperf server (On VM2 - ovs-br0-vport2 port)
$ iperf -s -u -p 8080

# Start iperf client
$ iperf -c 192.168.2.12 -u -p 8080 -b 32k

Server Report (Tested with 3 times):
1. Transfer: 41.6 KBytes & Bandwidth: 32.0 Kbits/s
2. Transfer: 41.6 KBytes & Bandwidth: 32.0 Kbits/s
3. Transfer: 41.6 KBytes & Bandwidth: 32.0 Kbits/s
```

1.3.10.14. Test Case 14

Description

```
+ generate speed: 60 kbits/s
+ Configuration:
> Port: cir = 32 kbits/s, pir = 80 kbit/s & Yelow color is sent
(details: cbs="2048", cir="4096", pbs="2048", pir="10240", ydropt="false")

+ Expected bandwidth: 60 kbits/s
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2: refer to __${This Task}/Note: below configuration command is used
for test case from 4 to 8__
# Delete open flow to use default queue: `$ sudo ovs-ofctl del-flows ovs-dpdk-br0 cookie=0x14/-1`

# Check the result command: `$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2`
```

Test result

```
# Start iperf server (On VM2 - ovs-br0-vport2 port)
$ iperf -s -u -p 8080

# Start iperf client
$ iperf -c 192.168.2.12 -u -p 8080 -b 60k

Server Report (Tested with 3 times):
1. Transfer: 76.1 KBytes & Bandwidth: 60.0 Kbits/s
2. Transfer: 76.1 KBytes & Bandwidth: 60.0 Kbits/s
3. Transfer: 76.1 KBytes & Bandwidth: 60.0 Kbits/s
```

1.3.10.15. Test Case 15

Description

```
+ generate speed: 80 kbits/s
+ Configuration:
> Port: cir = 32 kbits/s, pir = 80 kbit/s & Yelow color is sent
(details: cbs="2048", cir="4096", pbs="2048", pir="10240", ydropt="false")

+ Expected bandwidth: 80 kbits/s
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2: refer to __${This Task}/Note: below configuration command is used
for test case from 4 to 8__
# Delete open flow to use default queue: `$ sudo ovs-ofctl del-flows ovs-dpdk-br0 cookie=0x14/-1`
```

```
# Check the result command: `$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2`
```

Test result

```
# Start iperf server (On VM2 - ovs-br0-vport2 port)
$ iperf -s -u -p 8080

# Start iperf client
$ iperf -c 192.168.2.12 -u -p 8080 -b 80k

Server Report (Tested with 3 times):
1. Transfer: 100.0 KBytes & Bandwidth: 80.0 Kbits/s
2. Transfer: 100.0 KBytes & Bandwidth: 80.0 Kbits/s
3. Transfer: 100.0 KBytes & Bandwidth: 80.0 Kbits/s
```

1.3.10.16. Test Case 16

Description

```
+ generate speed: 1000 kbits/s
+ Configuration:
> Port: cir = 32 kbits/s, pir = 80 kbit/s & Yelow color is sent
(details: cbs="2048", cir="4096", pbs="2048", pir="10240", ydropt="false")

+ Expected bandwidth: 80 kbits/s
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2: refer to __${This Task}/Note: below configuration command is used
for test case from 4 to 8__
# Delete open flow to use default queue: `$ sudo ovs-ofctl del-flows ovs-dpdk-br0 cookie=0x14/-1`

# Check the result command: `$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2`
```

Test result

```
# Start iperf server (On VM2 - ovs-br0-vport2 port)
$ iperf -s -u -p 8080

# Start iperf client
$ iperf -c 192.168.2.12 -u -p 8080 -b 1m

Server Report (Tested with 3 times):
1. Transfer: 99.1 KBytes & Bandwidth: 79.0 Kbits/s
2. Transfer: 99.1 KBytes & Bandwidth: 79.0 Kbits/s
3. Transfer: 99.1 KBytes & Bandwidth: 79.0 Kbits/s
```

Note: below configuration command is used for test case from 17 to 22

```
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- \
--id=@newqos create qos type=egress-hqos2l-policer other-config:cir=4096 other-config:cbs=2048 other-
config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
qos=@qospl11,@qospl12,@qospl13,@qospl14 -- \
--id=@qospl11 create qos type=egress-policer-rfc2698 other-config:cir=1024 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:1=@queue1 -- \
--id=@qospl12 create qos type=egress-policer-rfc2698 other-config:cir=2048 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:2=@queue2 -- \
--id=@qospl13 create qos type=egress-policer-rfc2698 other-config:cir=4096 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:4=@queue4 -- \
```

```
--id=@qospl14 create qos type=egress-policer-rfc2698 other-config:cir=8192 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:8=@queue8 -- \
--id=@queue1 create queue other-config:desc=rfu -- \
--id=@queue2 create queue other-config:desc=rfu -- \
--id=@queue4 create queue other-config:desc=rfu -- \
--id=@queue8 create queue other-config:desc=rfu`
```

1.3.10.17. Test Case 17 (From 17 to 20: Basic testing for Policer Port + Policer Queue)

Description

```
+ generate speed: 8 kbits/s
+ Configuration: Port's CIR > Queue's CIR
> Port: cir = 32 kbits/s, pir = 80 kbit/s & Yelow color is dropted
(details: cbs="2048", cir="4096", pbs="2048", pir="10240", ydropt="true")
> Queue: cir = 8 kbits/s, pir = 80 kbit/s & Yelow color is dropted (Queue 1)
(details: cbs="2048", cir="1024", pbs="2048", pir="10240", ydropt="true")

+ Expected bandwidth: 8 kbits/s
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2: refer to __${This Task}/Note: below configuration command is used
for test case from 17 to 20__
# Using Queue 4 by configure openflow: `$ sudo ovs-ofctl add-flow ovs-dpdk-br0
cookie=0x14,in_port="ovs-br0-vport1",actions=set_queue:1,output:ovs-br0-vport2`

# Check the result command: `$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2`
```

Test result

```
# Start iperf server (On VM2 - ovs-br0-vport2 port)
$ iperf -s -u -p 8080

# Start iperf client
$ iperf -c 192.168.2.12 -u -p 8080 -b 8k

Server Report (Tested with 3 times):
1. Transfer: 11.5 KBytes & Bandwidth: 8.18 Kbits/s
2. Transfer: 11.5 KBytes & Bandwidth: 8.18 Kbits/s
3. Transfer: 11.5 KBytes & Bandwidth: 8.18 Kbits/s
```

1.3.10.18. Test Case 18

Description

```
+ generate speed: 60 kbits/s
+ Configuration: Port's CIR > Queue's CIR
> Port: cir = 32 kbits/s, pir = 80 kbit/s & Yelow color is dropted
(details: cbs="2048", cir="4096", pbs="2048", pir="10240", ydropt="true")
> Queue: cir = 8 kbits/s, pir = 80 kbit/s & Yelow color is dropted (Queue 1)
(details: cbs="2048", cir="1024", pbs="2048", pir="10240", ydropt="true")

+ Expected bandwidth: 8 kbits/s
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2: refer to __${This Task}/Note: below configuration command is used
for test case from 17 to 20__
# Using Queue 4 by configure openflow: `$ sudo ovs-ofctl add-flow ovs-dpdk-br0
cookie=0x14,in_port="ovs-br0-vport1",actions=set_queue:1,output:ovs-br0-vport2`
```

```
# Check the result command: `$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2`
```

Test result

```
# Start iperf server (On VM2 - ovs-br0-vport2 port)
$ iperf -s -u -p 8080

# Start iperf client
$ iperf -c 192.168.2.12 -u -p 8080 -b 60k

Server Report (Tested with 3 times):
1. Transfer: 11.5 KBytes & Bandwidth: 8.26 Kbits/s
2. Transfer: 11.5 KBytes & Bandwidth: 8.26 Kbits/s
3. Transfer: 11.5 KBytes & Bandwidth: 8.26 Kbits/s
```

1.3.10.19. Test Case 19

Description

```
+ generate speed: 32 kbits/s
+ Configuration: Port's CIR < Queue's CIR
> Port: cir = 32 kbits/s, pir = 80 kbit/s & Yelow color is dropted
(details: cbs="2048", cir="4096", pbs="2048", pir="10240", ydropt="true")
> Queue: cir = 64 kbits/s, pir = 80 kbit/s & Yelow color is dropted (Queue 8)
(details: cbs="2048", cir="8192", pbs="2048", pir="10240", ydropt="true")

+ Expected bandwidth: 32 kbits/s
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2: refer to __${This Task}/Note: below configuration command is used
for test case from 17 to 20__
# Using Queue 4 by configure openflow: `$ sudo ovs-ofctl add-flow ovs-dpdk-br0
cookie=0x14,in_port="ovs-br0-vport1",actions=set_queue:8,output:ovs-br0-vport2`

# Check the result command: `$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2`
```

Test result

```
# Start iperf server (On VM2 - ovs-br0-vport2 port)
$ iperf -s -u -p 8080

# Start iperf client
$ iperf -c 192.168.2.12 -u -p 8080 -b 32k

Server Report (Tested with 3 times):
1. Transfer: 41.6 KBytes & Bandwidth: 32.0 Kbits/s
2. Transfer: 41.6 KBytes & Bandwidth: 32.0 Kbits/s
3. Transfer: 41.6 KBytes & Bandwidth: 32.0 Kbits/s
```

1.3.10.20. Test Case 20

Description

```
+ generate speed: 80 kbits/s
+ Configuration: Port's CIR < Queue's CIR
> Port: cir = 32 kbits/s, pir = 80 kbit/s & Yelow color is dropted
(details: cbs="2048", cir="4096", pbs="2048", pir="10240", ydropt="true")
> Queue: cir = 64 kbits/s, pir = 80 kbit/s & Yelow color is dropted (Queue 8)
(details: cbs="2048", cir="8192", pbs="2048", pir="10240", ydropt="true")
```

```
+ Expected bandwidth: 32 kbits/s
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2: refer to __${This Task}/Note: below configuration command is used
for test case from 17 to 20__
# Using Queue 4 by configure openflow: `$ sudo ovs-ofctl add-flow ovs-dpdk-br0
cookie=0x14,in_port="ovs-br0-vport1",actions=set_queue:8,output:ovs-br0-vport2`

# Check the result command: `$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2`
```

Test result

```
# Start iperf server (On VM2 - ovs-br0-vport2 port)
$ iperf -s -u -p 8080

# Start iperf client
$ iperf -c 192.168.2.12 -u -p 8080 -b 80k

Server Report (Tested with 3 times):
1. Transfer: 34.5 KBytes & Bandwidth: 26.8 Kbits/s
2. Transfer: 34.5 KBytes & Bandwidth: 26.8 Kbits/s
3. Transfer: 34.5 KBytes & Bandwidth: 26.8 Kbits/s
```

1.3.10.21. Test Case 21 (From 21 to 22: testing for Policer Port + 2 Policer Queue)

Description

```
+ generate speed: 100 kbits/s
+ Configuration: Port's CIR (32 kbits/s) > total Queue's CIR (8 + 16 kbits/s)
> Port: cir = 32 kbits/s, pir = 80 kbit/s & Yelow color is dropted
(details: cbs="2048", cir="4096", pbs="2048", pir="10240", ydropt="true")
> Queue: cir = 8 kbits/s, pir = 80 kbit/s & Yelow color is dropted (Queue 1)
(details: cbs="2048", cir="1024", pbs="2048", pir="10240", ydropt="true")
> Queue: cir = 16 kbits/s, pir = 80 kbit/s & Yelow color is dropted (Queue 2)
(details: cbs="2048", cir="2048", pbs="2048", pir="10240", ydropt="true")

+ Expected bandwidth: 24 kbits/s (total)
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2: refer to __${This Task}/Note: below configuration command is used
for test case from 17 to 20__
# Using Queue 4 by configure openflow:
> `$ sudo ovs-ofctl add-flow ovs-dpdk-br0 cookie=0x14,in_port="ovs-br0-
vport1",actions=set_queue:1,output:ovs-br0-vport2`
> `$ sudo ovs-ofctl add-flow ovs-dpdk-br0 cookie=0x15,in_port="ovs-br0-
vport3",actions=set_queue:2,output:ovs-br0-vport2`

# Check the result command: `$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2`
```

Test result

```
# Start iperf server (On VM2 - ovs-br0-vport2 port)
$ iperf -s -u -p 8080
$ iperf -s -u -p 8088

# Start iperf client (VM1 - ovs-br0-vport1 & VM3 - ovs-br0-vport3)
$ iperf -c 192.168.2.12 -u -p 8080 -b 100k
$ iperf -c 192.168.2.12 -u -p 8088 -b 100k

Server Report (Tested with 3 times):
```

```
1. Transfer: 11.5 KBytes & Bandwidth: 8.38 Kbits/s (VM1) & 17.2 KBytes & Bandwidth: 13.8 Kbits/s
(VM3)
2. Transfer: 10.0 KBytes & Bandwidth: 8.09 Kbits/s (VM1) & 17.2 KBytes & Bandwidth: 13.1 Kbits/s
(VM3)
3. Transfer: 11.5 KBytes & Bandwidth: 8.38 Kbits/s (VM1) & 17.2 KBytes & Bandwidth: 13.8 Kbits/s
(VM3)
```

1.3.10.22. Test Case 22

Description

```
+ generate speed: 100 kbits/s
+ Configuration: Port's CIR (32 kbits/s) < total Queue's CIR (32 + 16 kbits/s)
> Port: cir = 32 kbits/s, pir = 80 kbit/s & Yelow color is dropted
(details: cbs="2048", cir="4096", pbs="2048", pir="10240", ydropt="true")
> Queue: cir = 32 kbits/s, pir = 80 kbit/s & Yelow color is dropted (Queue 4)
(details: cbs="2048", cir="4096", pbs="2048", pir="10240", ydropt="true")
> Queue: cir = 16 kbits/s, pir = 80 kbit/s & Yelow color is dropted (Queue 2)
(details: cbs="2048", cir="2048", pbs="2048", pir="10240", ydropt="true")

+ Expected bandwidth: 32 kbits/s (total)
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2: refer to __${This Task}/Note: below configuration command is used
for test case from 17 to 20__
# Using Queue 4 by configure openflow:
> `$ sudo ovs-ofctl add-flow ovs-dpdk-br0 cookie=0x14,in_port="ovs-br0-
vport1",actions=set_queue:4,output:ovs-br0-vport2`
> `$ sudo ovs-ofctl add-flow ovs-dpdk-br0 cookie=0x15,in_port="ovs-br0-
vport3",actions=set_queue:2,output:ovs-br0-vport2`

# Check the result command: `$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2`
```

Test result

```
# Start iperf server (On VM2 - ovs-br0-vport2 port)
$ iperf -s -u -p 8080
$ iperf -s -u -p 8088

# Start iperf client (VM1 - ovs-br0-vport1 & VM3 - ovs-br0-vport3)
$ iperf -c 192.168.2.12 -u -p 8080 -b 100k
$ iperf -c 192.168.2.12 -u -p 8088 -b 100k

Server Report (Tested with 3 times):
1. Transfer: 34.5 KBytes & Bandwidth: 28.6 Kbits/s (VM1) & 4.31 KBytes & Bandwidth: 3.29 Kbits/s
(VM3)
2. Transfer: 33.0 KBytes & Bandwidth: 26.1 Kbits/s (VM1) & 8.61 KBytes & Bandwidth: 5.77 Kbits/s
(VM3)
3. Transfer: 37.3 KBytes & Bandwidth: 29.5 Kbits/s (VM1) & 4.31 KBytes & Bandwidth: 2.88 Kbits/s
(VM3)
```

1.4. Egress policer 2 rate Three Color Marker (trTCM) - (Sept 12, 2019)

- About single/two rate three color marker (srTCM/trTCM), refer **Introduce srTCM/trTCM**
- About srTCM, refer RFC2697
- About trTCM, refer RFC2698 or RFC4115 (have a little difference)

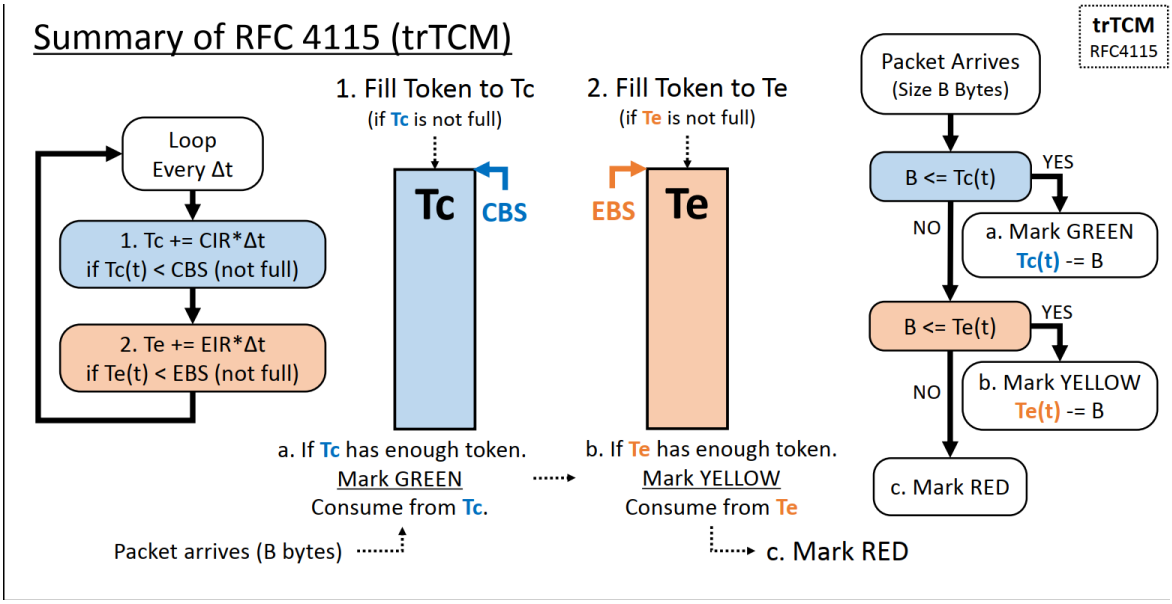


Figure 1-2 - RFC4115 - trTCM

1.4.1. Current status

- Implementation: done
- basic testing by iperf: refer to **${This Task}/Egress policer trTCM rfc2698 testing** session
- basic testing with egress srTCM rfc2697: refer to **${this task}/Egress policer srTCM rfc2697** session
- detail testing & find the way to measurement uncertainty: TODO

1.4.2. Expectation of this task

Add egress policer trTCM in OVS*DPDK (rfc2698 & rfc4115)

1.4.3. Issue Note

Detail implementation of current egress policer of OVS&DPDK ??

```
'''
#Register a Policer QoS type: bridge_run() --> dpdk_init() --> dpdk_init__() -->
netdev_dpdk_register():
void
netdev_dpdk_register(void)
{
    netdev_register_provider(&dpdk_class);
    netdev_register_provider(&dpdk_ring_class);
    netdev_register_provider(&dpdk_vhost_class); //dpdkvhostuser registed here
    netdev_register_provider(&dpdk_vhost_client_class);
}

static const struct netdev_class dpdk_vhost_class = {
    .type = "dpdkvhostuser",
    NETDEV_DPDK_CLASS_COMMON,    //register here

#define NETDEV_DPDK_CLASS_COMMON                            \
    .is_pmd = true,                                         \
    .alloc = netdev_dpdk_alloc,                             \
    ...                                                     \
    .get_qos_types = netdev_dpdk_get_qos_types,             \  /*used here*/
    .get_qos = netdev_dpdk_get_qos,                         \
    .set_qos = netdev_dpdk_set_qos,                         \

static int
    netdev_dpdk_get_qos_types(const struct netdev *netdev OVS_UNUSED,
                        struct sset *types)
```

```
{
    const struct dpdk_qos_ops *const *opsp;

    for (opsp = qos_confs /*qos_confs is used here*/; *opsp != NULL; opsp++) {
        const struct dpdk_qos_ops *ops = *opsp;
        if (ops->qos_construct && ops->qos_name[0] != '\0') {
            sset_add(types, ops->qos_name);
        }
    }
    return 0;
}

static const struct dpdk_qos_ops *const qos_confs[] = {
    &egress_policer_ops,    //

static const struct dpdk_qos_ops egress_policer_ops = {
    "egress-policer",    /* qos_name */
    egress_policer_qos_construct,
    egress_policer_qos_destruct,
    egress_policer_qos_get,
    egress_policer_qos_is_equal,
    egress_policer_run
};
'''
```

How to test them srTCM in OVS&DPDK???

- package will be handle & marker color here netdev_dpdk_policer_run() –> netdev_dpdk_policer_pkt_handle() –> rte_meter_srtcm_color_blind_check() //for srTCM
- In dpdk, we also have api for trTCM for RFC4115: rte_meter_trtcm_rfc4115_color_blind_check() /* blind mode */ *or rte_meter_trtcm_rfc4115_color_aware_check* / aware mode */
- Define test model to testing this one //TODO

Configure & checking command of OVS&DPDK about egress policer

```
Refer to session __OVS Commands/QoS user command line__ of this document
```

Update for trTCM RFC4115 in OVS&DPDK??? refer to the session **update for trTCM RFC4115 in OVS&DPDK**

Update for trTCM RFC2698 in OVS&DPDK??? refer to the session **update for trTCM RFC2698 in OVS&DPDK**

1.4.4. update for trTCM RFC4115 in OVS&DPDK

1.4.4.1. Define egress policer trCTM (rfc4115) configuration & testing command

```
# Configure QoS egress policing trTCM (rfc4115)
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- --id=@newqos create qos type=egress-policer-rfc4115 other-config:cir=46000000 other-config:cbs=2048 other-config:eir=23000000 other-config:ebs=1024`
```

HIGHLIGHT Current OVS (2.11.x & 2.12.x) don't compatible with supported rfc4115 of dpdk (19.XX).

1.4.5. update for trTCM RFC2698 in OVS&DPDK

1.4.5.1. Define egress policer trCTM (rfc2698) configuration & testing command

```
# Configure QoS egress policing trTCM (rfc2698)
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- --id=@newqos create qos type=egress-policer-rfc2698 other-config:cir=104857600 other-config:cbs=2048 other-config:pir=157286400 other-config:pbs=2048  other-config:ydropt=false other-config:rdropt=true`

# Edit QoS params
`$ sudo ovs-vsctl set qos __uuid__ other-config:....`
```

```
Note:
+ PIR have to equal or greater CIR
+ Default actions of yellow & red color is dropt (ydropt=true & rdropt=true)
```

1.4.6. Egress policer srTCM rfc2697 testing

how to configure & testing with iperf ? Please refer to **atsdk_ovs_note.md** documents

Environment Testing

```
+ Testing on SERVER 9
+ VM1 & VM2: RAM:1G, Number of core: 1
```

1.4.6.1. Test Case 1

Description

```
+ generate speed: 5 MBytes/s (40 Mbits/s)
+ Configuration: cir = 10 MBytes/s, cbs = 2048

+ Expected bandwidth: 5 MBytes/s (or 40 Mbits/s)
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- --id=@newqos create qos type=egress-policer
other-config:cir=10485760 other-config:cbs=2048`

# Check the result
~/ws/ovs_log$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2
QoS: ovs-br0-vport2 egress-policer
cir: 10485760
cbs: 2048
```

Test result

```
# Start iperf server
$ iperf -s -u -p 8080

# Start iperf client (40 Mbits/s)
$ iperf -c 192.168.2.12 -u -p 8080 -b 40m

Server Report (Tested with 3 times):
1. Transfer: 47.1 MBytes & Bandwidth: 39.5 Mbits/s
2. Transfer: 46.7 MBytes & Bandwidth: 38.2 Mbits/s
3. Transfer: 47.1 MBytes & Bandwidth: 39.5 Mbits/s
```

1.4.6.2. Test Case 2

Description

```
+ generate speed: 10 MBytes/s (80 Mbits/s)
+ Configuration: cir = 10 MBytes/s, cbs = 2048

+ Expected bandwidth: 10 MBytes/s (or 80 Mbits/s)
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2
```

```
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- --id=@newqos create qos type=egress-policer
other-config:cir=10485760 other-config:cbs=2048`

# Check the result
~/ws/ovs_log$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2
QoS: ovs-br0-vport2 egress-policer
cir: 10485760
cbs: 2048
```

Test result

```
# Start iperf server
$ iperf -s -u -p 8080

# Start iperf client (80 Mbits/s)
$ iperf -c 192.168.2.12 -u -p 8080 -b 80m

Server Report (Tested with 3 times):
1. Transfer: 94.5 MBytes & Bandwidth: 79.3 Mbits/s
2. Transfer: 94.5 MBytes & Bandwidth: 79.3 Mbits/s
3. Transfer: 94.5 MBytes & Bandwidth: 79.0 Mbits/s
```

1.4.6.3. Test Case 3

Description

```
+ generate speed: 12 MBytes/s (96 Mbits/s)
+ Configuration: cir = 10 MBytes/s, cbs = 2048

+ Expected bandwidth: 10 MBytes/s (or 80 Mbits/s)
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- --id=@newqos create qos type=egress-policer
other-config:cir=10485760 other-config:cbs=2048`

# Check the result
~/ws/ovs_log$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2
QoS: ovs-br0-vport2 egress-policer
cir: 10485760
cbs: 2048
```

Test result

```
# Start iperf server
$ iperf -s -u -p 8080

# Start iperf client (96 Mbits/s)
$ iperf -c 192.168.2.12 -u -p 8080 -b 96m

Server Report (Tested with 3 times):
1. Transfer: 85.0 MBytes & Bandwidth: 71.3 Mbits/s
2. Transfer: 85.2 MBytes & Bandwidth: 71.5 Mbits/s
3. Transfer: 85.1 MBytes & Bandwidth: 71.4 Mbits/s
```

1.4.6.4. Test Case 4

Description

```
+ generate speed: 20 MBytes/s (160 Mbits/s)
+ Configuration: cir = 10 MBytes/s, cbs = 2048
```

```
+ Expected bandwidth: 10 MBytes/s (or 80 Mbits/s)
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- --id=@newqos create qos type=egress-policer
other-config:cir=10485760 other-config:cbs=2048`

# Check the result
~/ws/ovs_log$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2
QoS: ovs-br0-vport2 egress-policer
cir: 10485760
cbs: 2048
```

Test result

```
# Start iperf server
$ iperf -s -u -p 8080

# Start iperf client (160 Mbits/s)
$ iperf -c 192.168.2.12 -u -p 8080 -b 160m

Server Report (Tested with 3 times):
1. Transfer: 94.2 MBytes & Bandwidth: 77.1 Mbits/s (log: received out-of-order)
2. Transfer: 93.8 MBytes & Bandwidth: 76.7 Mbits/s (log: received out-of-order)
3. Transfer:94.1 MBytes & Bandwidth: 78.9 Mbits/s
```

1.4.7. Egress policer trTCM rfc2698 testing

how to configure & testing with iperf ? Please refer to **atsdk_ovs_note.md** documents

Environment Testing

```
+ Testing on SERVER 9
+ VM1 & VM2: RAM:1G, Number of core: 1
```

1.4.7.1. Test Case 1

Description

```
+ generate speed: 10 MBytes/s (80 Mbits/s)
+ Configuration: cir = 20 MBytes/s, pir = 30 MBytes/s, pbs & cbs = 2048
+ Yelow color is dropted.

+ Expected bandwidth: 10 MBytes/s (or 80 Mbits/s)
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- --id=@newqos create qos type=egress-
policer-rfc2698 other-config:cir=20971520 other-config:cbs=2048 other-config:pir=31457280 other-
config:pbs=2048  other-config:ydropt=true`

# Check the result
~/ws/ovs_log$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2
QoS: ovs-br0-vport2 egress-policer-rfc2698
cir: 20971520
pbs: 2048
rdropt: true
cbs: 2048
ydropt: true
```

```
pir: 31457280
```

Test result

```
# Start iperf server
$ iperf -s -u -p 8080

# Start iperf client (400 Mbits/s)
$ iperf -c 192.168.2.12 -u -p 8080 -b 80m

Server Report (Tested with 3 times):
1. Transfer: 94.6 MBytes & Bandwidth: 79.4 Mbits/s
2. Transfer: 94.7 MBytes & Bandwidth: 79.4 Mbits/s
3. Transfer: 95   MBytes & Bandwidth: 79.7 Mbits/s
```

1.4.7.2. Test Case 2

Description

```
+ generate speed: 20 MBytes/s (160 Mbits/s)
+ Configuration: cir = 20 MBytes/s, pir = 30 MBytes/s, pbs & cbs = 2048
+ Yelow color is dropted.

+ Expected bandwidth: 20 MBytes/s (or 160 Mbits/s)
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- --id=@newqos create qos type=egress-
policer-rfc2698 other-config:cir=20971520 other-config:cbs=2048 other-config:pir=31457280 other-
config:pbs=2048  other-config:ydropt=true`

# Check the result
~/ws/ovs_log$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2
QoS:  ovs-br0-vport2 egress-policer-rfc2698
cir: 20971520
pbs: 2048
rdropt: true
cbs: 2048
ydropt: true
pir: 31457280
```

Test result

```
# Start iperf server
$ iperf -s -u -p 8080

# Start iperf client (160 Mbits/s)
$ iperf -c 192.168.2.12 -u -p 8080 -b 160m

Server Report (Tested with 3 times):
1. Transfer: 163    MBytes & Bandwidth: 137 Mbits/s
2. Transfer: 160    MBytes & Bandwidth: 134 Mbits/s
3. Transfer: 168    MBytes & Bandwidth: 138 Mbits/s

=> it looks like limitation of environment
```

1.4.7.3. Test Case 3

Description

```
+ generate speed: 10 MBytes/s (80 Mbits/s)
```

```
+ Configuration: cir = 10 MBytes/s, pir = 15 MBytes/s, pbs & cbs = 2048
+ Yelow color is dropted.

+ Expected bandwidth: 10 MBytes/s (or 80 Mbits/s)
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- --id=@newqos create qos type=egress-
policer-rfc2698 other-config:cir=10485760 other-config:cbs=2048 other-config:pir=15728640 other-
config:pbs=2048  other-config:ydropt=true`

# Check the result
~/ws/ovs_log$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2
QoS: ovs-br0-vport2 egress-policer-rfc2698
cir: 10485760
pbs: 2048
rdropt: true
cbs: 2048
ydropt: true
pir: 15728640
```

Test result

```
# Start iperf server
$ iperf -s -u -p 8080

# Start iperf client (80 Mbits/s)
$ iperf -c 192.168.2.12 -u -p 8080 -b 80m

Server Report (Tested with 3 times):
1. Transfer: 94.9 MBytes & Bandwidth: 79.6 Mbits/s
2. Transfer: 94.8 MBytes & Bandwidth: 79.5 Mbits/s
3. Transfer: 95   MBytes & Bandwidth: 79.7 Mbits/s
```

1.4.7.4. Test Case 4

Description

```
+ generate speed: 15 MBytes/s (120 Mbits/s)
+ Configuration: cir = 10 MBytes/s, pir = 15 MBytes/s, pbs & cbs = 2048
+ Yelow color is dropted.

+ Expected bandwidth: 10 MBytes/s (or 80 Mbits/s)
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- --id=@newqos create qos type=egress-
policer-rfc2698 other-config:cir=10485760 other-config:cbs=2048 other-config:pir=15728640 other-
config:pbs=2048  other-config:ydropt=true`

# Check the result
~/ws/ovs_log$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2
QoS: ovs-br0-vport2 egress-policer-rfc2698
cir: 10485760
pbs: 2048
rdropt: true
cbs: 2048
ydropt: true
pir: 15728640
```

Test result

```
# Start iperf server
$ iperf -s -u -p 8080

# Start iperf client (120 Mbits/s)
$ iperf -c 192.168.2.12 -u -p 8080 -b 120m

Server Report (Tested with 3 times):
1. Transfer: 93.8 MBytes & Bandwidth: 78.7 Mbits/s
2. Transfer: 94.8 MBytes & Bandwidth: 77.6 Mbits/s
3. Transfer: 94.2 MBytes & Bandwidth: 79.1 Mbits/s
```

1.4.7.5. Test Case 5

Description

```
+ generate speed: 2 MBytes/s (16 Mbits/s)
+ Configuration: cir = 4 MBytes/s, pir = 6 MBytes/s, pbs & cbs = 2048
+ Yelow color is not dropted.

+ Expected bandwidth: 2 MBytes/s (or 16 Mbits/s)
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- --id=@newqos create qos type=egress-
policer-rfc2698 other-config:cir=4194304 other-config:cbs=2048 other-config:pir=6291456 other-
config:pbs=2048  other-config:ydropt=false`

# Check the result
~/ws/ovs_log$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2
QoS: ovs-br0-vport2 egress-policer-rfc2698
cir: 4194304
pbs: 2048
rdropt: true
cbs: 2048
ydropt: false
pir: 6291456
```

Test result

```
# Start iperf server
$ iperf -s -u -p 8080

# Start iperf client (16 Mbits/s)
$ iperf -c 192.168.2.12 -u -p 8080 -b 16m

Server Report (Tested with 3 times):
1. Transfer: 19     MBytes & Bandwidth: 15.9    Mbits/s
2. Transfer: 19     MBytes & Bandwidth: 15.9    Mbits/s
3. Transfer: 18.9   MBytes & Bandwidth: 15.9    Mbits/s
```

1.4.7.6. Test Case 6

Description

```
+ generate speed: 4 MBytes/s (32 Mbits/s)
+ Configuration: cir = 4 MBytes/s, pir = 6 MBytes/s, pbs & cbs = 2048
+ Yelow color is not dropted.

+ Expected bandwidth: 4 MBytes/s (or 32 Mbits/s)
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- --id=@newqos create qos type=egress-policer-rfc2698 other-config:cir=4194304 other-config:cbs=2048 other-config:pir=6291456 other-config:pbs=2048  other-config:ydropt=false`

# Check the result
~/ws/ovs_log$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2
QoS:  ovs-br0-vport2 egress-policer-rfc2698
cir: 4194304
pbs: 2048
rdropt: true
cbs: 2048
ydropt: false
pir: 6291456
```

Test result

```
# Start iperf server
$ iperf -s -u -p 8080

# Start iperf client (32 Mbits/s)
$ iperf -c 192.168.2.12 -u -p 8080 -b 32m

Server Report (Tested with 3 times):
1. Transfer: 37.9   MBytes & Bandwidth: 31.8    Mbits/s
2. Transfer: 37.9   MBytes & Bandwidth: 31.8    Mbits/s
3. Transfer: 38     MBytes & Bandwidth: 31.9    Mbits/s
```

1.4.7.7. Test Case 7

Description

```
+ generate speed: 6 MBytes/s (48 Mbits/s)
+ Configuration: cir = 4 MBytes/s, pir = 6 MBytes/s, pbs & cbs = 2048
+ Yelow color is not dropted.

+ Expected bandwidth: 6 MBytes/s (or 48 Mbits/s)
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- --id=@newqos create qos type=egress-policer-rfc2698 other-config:cir=4194304 other-config:cbs=2048 other-config:pir=6291456 other-config:pbs=2048  other-config:ydropt=false`

# Check the result
~/ws/ovs_log$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2
QoS:  ovs-br0-vport2 egress-policer-rfc2698
cir: 4194304
pbs: 2048
rdropt: true
cbs: 2048
ydropt: false
pir: 6291456
```

Test result

```
# Start iperf server
$ iperf -s -u -p 8080

# Start iperf client (48 Mbits/s)
```

```
$ iperf -c 192.168.2.12 -u -p 8080 -b 48m

Server Report (Tested with 3 times):
1. Transfer: 55.8   MBytes & Bandwidth: 46.8    Mbits/s
2. Transfer: 56.7   MBytes & Bandwidth: 47.5    Mbits/s
3. Transfer: 56.8   MBytes & Bandwidth: 47.7    Mbits/s
```

1.4.7.8. Test Case 8

Description

```
+ generate speed: 8 MBytes/s (64 Mbits/s)
+ Configuration: cir = 4 MBytes/s, pir = 6 MBytes/s, pbs & cbs = 2048
+ Yelow color is not dropted.

+ Expected bandwidth: 6 MBytes/s (or 48 Mbits/s)
```

Configure QoS (egress-policer-rfc2698)

```
# Create QoS in ovs-br0-vport2
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- --id=@newqos create qos type=egress-
policer-rfc2698 other-config:cir=4194304 other-config:cbs=2048 other-config:pir=6291456 other-
config:pbs=2048  other-config:ydropt=false`

# Check the result
~/ws/ovs_log$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2
QoS:  ovs-br0-vport2 egress-policer-rfc2698
cir: 4194304
pbs: 2048
rdropt: true
cbs: 2048
ydropt: false
pir: 6291456
```

Test result

```
# Start iperf server
$ iperf -s -u -p 8080

# Start iperf client (64 Mbits/s)
$ iperf -c 192.168.2.12 -u -p 8080 -b 64m

Server Report (Tested with 3 times):
1. Transfer: 50.4   MBytes & Bandwidth: 42.2    Mbits/s
2. Transfer: 50.8   MBytes & Bandwidth: 41.6    Mbits/s (log: received out-of-order)
3. Transfer: 50.5   MBytes & Bandwidth: 41.3    Mbits/s (log: received out-of-order)
```

1.5. OVS-DPDK QoS scheduling/shaper user interface command

1.5.1. Expectation of OVS-DPDK QoS scheduling

- Implement QoS scheduling per port/interface
- Implements QoS scheduling user commandline interface

1.5.2. Issue list

QoS scheduling ?

> Refer session **Queuing and Scheduling Concepts**

QoS scheduling per port/interface ? //TODO

QoS egress policing user commandline supported?

> Refer session **OVS Commands/QoS user command line**

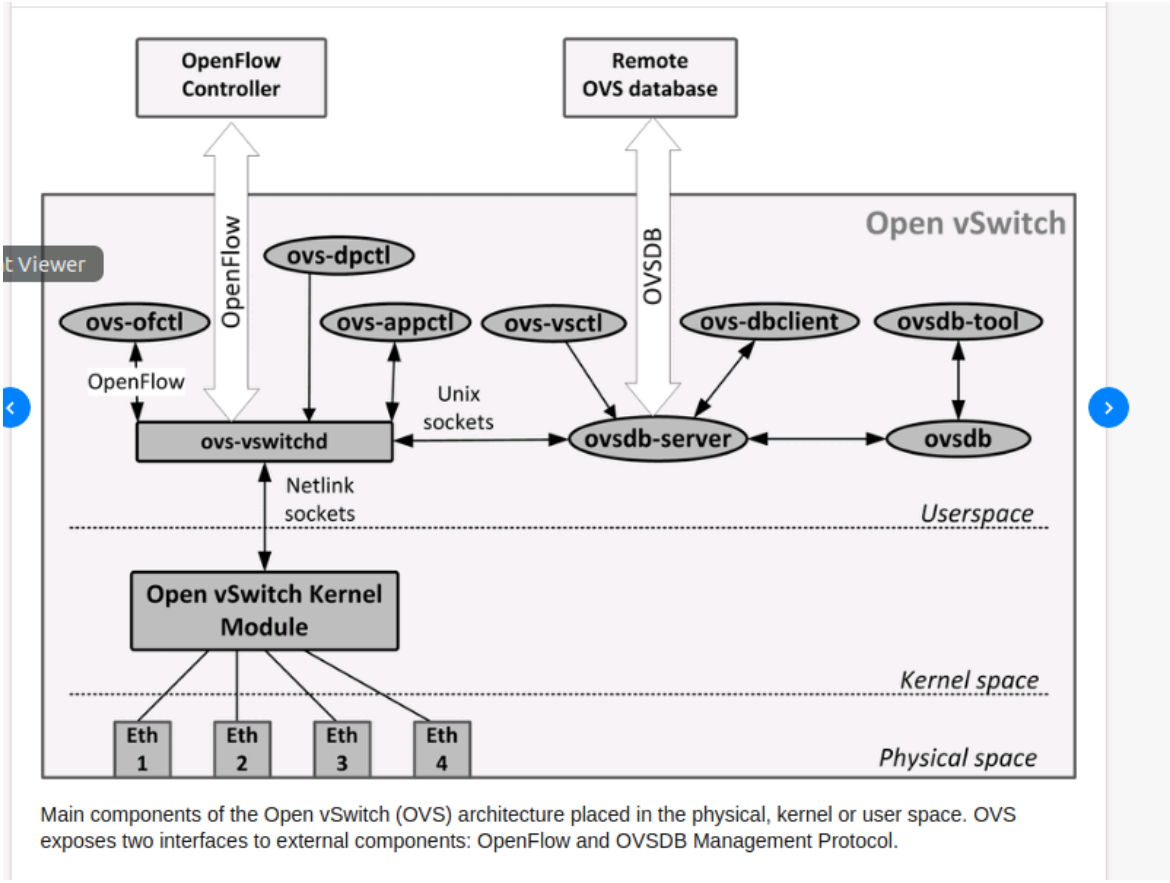


Figure 1-3 - OVS Diagram

How ovs-vsctl application connect with ovsdb-server ? //TODO

How ovs_appctl application connect with ovs-vswitchd ? //TODO

Do we have other user command line relate with QoS ? //TODO

1.5.3. Result / solution of this task

Suggestion: QoS egress scheduler command line refer to **${Task Note}/Egress policer: 3 Queue & 4 Policer trTCM rfc2698 (Sept 19, 2019)/Configuration Command**

Confirm effection of command in database by monitor command contents of COLUMNs in TABLE in DATABASE on SERVER

```
`$ sudo ovsdb-client monitor ALL`   //Monitor all table in DATABASE on SERVER

`$ sudo ovsdb-client monitor QoS`    //Monitor table QoS in DATABASE on SERVER

`$ sudo ovsdb-client monitor Queue` //Monitor table Queue in DATABASE on SERVER
```

1.5.4. Information & Note

- The two most important parameters associated with the queuing and scheduling mechanism are buffers and bandwidth

2. C/C++ programming language

2.1. pipe() System call

Prerequisite : I/O System calls

Conceptually, a pipe is a connection between two processes, such that the standard output from one process becomes the standard input of the other process. In UNIX Operating System, Pipes are useful for communication between related processes(inter-process communication).

- Pipe is one-way communication only i.e we can use a pipe such that One process write to the pipe, and the other process reads from the pipe. It opens a pipe, which is an area of main memory that is treated as a “virtual file”.
- The pipe can be used by the creating process, as well as all its child processes, for reading and writing. One process can write to this “virtual file” or pipe and another related process can read from it.
- If a process tries to read before something is written to the pipe, the process is suspended until something is written.
- The pipe system call finds the first two available positions in the process’s open file table and allocates them for the read and write ends of the pipe.

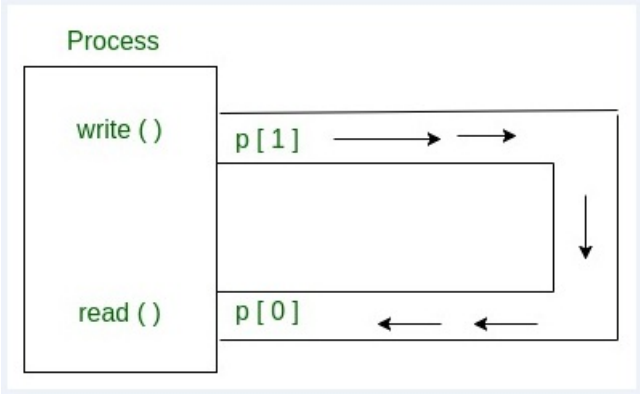


Figure 2-1 - Pipe Concept

2.2. Coding convention of Linux Kernel Source code

```
Refer to [Coding Style Linux Kernel](https://www.kernel.org/doc/html/v4.10/process/coding-style.html)
```

3. DPDK Note

3.1. DPDK Example

3.1.1. cmdline example

```
# start it with chelsio NIC (using pci 0000:05:00.4) in server 9
`$ sudo ./build/cmdline -l 2-3 --file-prefix cmdline -w 0000:05:00.3`
```

3.1.2. helloworld

```
# Start it with chelsio NIC (using pci 0000:05:00.4) in server 9
`$ sudo ./build/helloworld --file-prefix helloworld -l 0-4 -n 5 -w 0000:05:00.4`

Note: Update code for investigate
+ Communicate between master & slaver is PIPE (array pipe: pipe[0] for read & pip[1] for write)
+ Slaver can communicate between them ? - They can if known exactly the core_id of them.
```

3.1.3. QoS Sched (qos_sched)

```
# qos_sched running command (using pci 0000:05:00.4 - Chelsio)
`$ sudo ./build/qos_sched -l 1,5,7 -n 4 --file-prefix qos_sched -w 0000:05:00.4 -- --pfc "0,0,5,7"
--cfg ./profile.cfg`
```

3.2. DPDK Solved Issue

3.2.1. WARNING “No free hugepages reported in hugepages–1048576kB”

Root cause The application need huge for running

Solution using -m or –socketmem to explicit declare the huge page (-m 1024,1024 or –socketmem=1024,1024 ==> 2 NUMA node system)

3.3. CPU Core & Thread

- Master core (thread) communicate with slave core via 2 PIPE (**`pipe_master2slave & pipe_slave2master`**: [0] for read & [1] for write)
- Thread have lcore_id == lcore_id in lcore_config will execute input function via **`rte_eal_remote_launch()`** function
- Every configuration of core will be added in lcore_config[core_id]

4. Testing tools

4.1. Spirent TestCenter Application (STC)

```
This one is used to send & received package (testing with high bandwidth):

+ [Software](\\it-ref\All_Ref\ATVN_Ref_CD\Spirent SPT-N4U-220\Disk01\Spirent TestCenter Applications -
Windows)
+ Configuration information:
> * IP: 172.33.42.241
> * PORT: Slot2, port 5-8
> * STC port map: port 5 -> 0000:03:00.1, port 6 -> 0000:03:00.0, port 7 -> 0000:04:00.1, port 8 -
> 0000:04:00.0
```

4.2. intel parallel studio

This tool is used to analysis source code,... & it has already installed in SERVER 9 at **`/opt/intel`**. To start the tool, use below commands

```
$ source /opt/intel/parallel_studio_xe_2019.4.070/bin/psxevars.sh
$ amplxe-gui
```

How to configure & monitor a program? (TBU)

4.3. gprof2dot

This tool is create call graph of program

- Install gprof2dot, please refer **link**
- How to use it with result of **`intel parallel studio`**, please refer **link**

4.4. sFlowTrend

This tool is used to monitor VM traffic using sFlow.

- To install it, refer **link**
- Configure & use it, refer **link**

Example:

```
#export a few valua
export COLLECTOR_IP=172.33.41.8
export COLLECTOR_PORT=6343
export AGENT_IP=eno1
export HEADER_BYTES=128
export SAMPLING_N=64
export POLLING_SECS=10

#Create sFlow on OVS host
$ sudo ovs-vsctl -- --id=@sflow create sflow agent=${AGENT_IP}
target="\"${COLLECTOR_IP}:${COLLECTOR_PORT}\"" header=${HEADER_BYTES} sampling=${SAMPLING_N}
polling=${POLLING_SECS} -- set bridge ovs-atvn-br0 sflow=@sflow

# Start sFlowTren on Monitor host, one sFlowTren fully start up, it could be received data from OVS
host

Note:
+ To see all current sets of sFlow configuration: `$ sudo ovs-vsctl list sflow`
+ Troubleshooting: `sudo tcpdump -ni eno1 udp port 6343` (eno1: network interface)
```

4.5. iperf

This tool is used to perform network throughput tests.

- Install iperf in ubuntu by command **apt-get install iperf**
- For more information, use command **iperf --help**

Example using iperf for testing

```
#At Server side, deploy an iPerf server in UDP mode on port 8080
$ iperf -s -u -p 8080

#At client side, deploy an iPerf client in UDP mode on port 8080 with a transmission bandwidth of
100Mbps
$ iperf -c 192.168.2.11 -u -p 8080 -b 100m
```

4.6. trafgen

This tool is a fast, multithreaded network packet generator

```
+ Install trafgen (in netsniff-ng toolkit) by command `$sudo apt-get install netsniff-ng`
+ For more information about using it, please refer its manual by command `man trafgen`
```

Example:

```
# Generate & send out 1000 packages
$ sudo trafgen --dev enp7s0 --cpp --conf trafgen.cfg -n1000

Note:
--cpp: pass  the  packet configuration to the C preprocessor before reading it
into trafgen

trafgen.cfg example

/* Note: dynamic elements make trafgen slower! */
#include <stddef.h>

{
/* MAC Destination */
fill(0xff, ETH_ALEN),   /*Update destination mac address here, example: 0xA0,0x01,drnd(4),*/
/* MAC Source */
0x00, 0x02, 0xb3, drnd(3), /*Update source mac address here*/
/* IPv4 Protocol */
c16(ETH_P_IP),
/* IPv4 Version, IHL, TOS */
0b01000101, 0,
/* IPv4 Total Len */
c16(59),
/* IPv4 Ident */
drnd(2),
/* IPv4 Flags, Frag Off */
0b01000000, 0,
/* IPv4 TTL */
64,
/* Proto TCP */
0x06,
/* IPv4 Checksum (IP header from, to) */
csumip(14, 33),
/* Source IP */
drnd(4),        /*Update source IP address here: 192,168,2,10,*/
/* Dest IP */
drnd(4),        /*Update destination IP address here: 192,168,2,11,*/
/* TCP Source Port */
drnd(2),
/* TCP Dest Port */
```

```
c16(80),
/* TCP Sequence Number */
drnd(4),
/* TCP Ackn. Number */
c32(0),
/* TCP Header length + TCP SYN/ECN Flag */
c16((8 << 12) | TCP_FLAG_SYN | TCP_FLAG_ECE)
/* Window Size */
c16(16),
/* TCP Checksum (offset IP, offset TCP) */
csumtcp(14, 34),
/* TCP Options */
0x00, 0x00, 0x01, 0x01, 0x08, 0x0a, 0x06,
0x91, 0x68, 0x7d, 0x06, 0x91, 0x68, 0x6f,
/* Data blob */
"gotcha!",  /*Update data here*/
}
```

4.7. tshark

This tool is dump and analyze network traffic.

```
+ Install `wireshark network analyzer` (TBU)
+ More information, pleaser refer manual by command `man tshark`

Note: for simple using, just use command `$ sudo tshark`
```

5. Unix Commands

5.1. Huge page & numa node

Check huge page for every node

```
#using numactl
`$ sudo apt install numactl`    #install numactl
`$ numastat -m      #show statictis of numa node

# Check in system
`$ cat /proc/meminfo`   #show hugepage information of system
`$ cat /sys/devices/system/node/node*/meminfo` #show information of node*
```

Setup huge page by commandline

```
# For persistent allocation of huge pages, write to hugepages.conf file in /etc/sysctl.d
`$ echo 'vm.nr_hugepages=2048' > /etc/sysctl.d/hugepages.conf`

# For run-time allocation of huge pages, use the sysctl utility
`$ sysctl -w vm.nr_hugepages=N`

# Mount the hugepages, if not already mounted by default
`$ mount -t hugetlbfs none /dev/hugepages`
```

5.2. intel vtune profiler

```
#export all environment variable
`$ source /opt/intel/parallel_studio_xe_2019.4.070/bin/psxevars.sh`

#Start vtune profiler
`$ amplxe-gui   //should use sudo permission.
```

5.3. QEMU

```
#Convert image from virtualbox format (lubuntuuu.vdi) to qemu format (lubuntuuu.qcow2)
`$ qemu-img convert -f vdi -O qcow2 lubuntuuu.vdi lubuntuuu.qcow2`

# QEMU with dpdkvhostuser port (dpdkvhostuser ports are considered deprecated;  please migrate to
dpdkvhostuserclient ports)
`$ sudo qemu-system-x86_64 -m 1024 -smp 1 -cpu host -hda ${HOME}/qemu/lubuntu01.qcow2 -boot c -
enable-kvm -no-reboot -net none -chardev socket,id=char1,path=${HOME}/ovs_prj/ovs-
software/var/run/openvswitch/ovs-br0-vport1 -netdev type=vhost-user,id=ovs-br0-
vport1,chardev=char1,vhostforce -device virtio-net-pci,mac=A0:01:00:00:00:00,netdev=ovs-br0-vport1 -
object memory-backend-file,id=mem,size=1G,mem-path=/mnt/huge,share=on -numa node,memdev=mem -mem-
prealloc -virtfs local,path=${HOME}/iperf_debs,mount_tag=host0,security_model=none,id=vm1_dev`

`$ sudo qemu-system-x86_64 -m 1024 -smp 1 -cpu host -hda ${HOME}/qemu/lubuntu02.qcow2 -boot c -
enable-kvm -no-reboot -net none -chardev socket,id=char2,path=${HOME}/ovs_prj/ovs-
software/var/run/openvswitch/ovs-br0-vport2 -netdev type=vhost-user,id=ovs-br0-
vport2,chardev=char2,vhostforce -device virtio-net-pci,mac=A0:02:00:00:00:00,netdev=ovs-br0-vport2 -
object memory-backend-file,id=mem,size=1G,mem-path=/mnt/huge,share=on -numa node,memdev=mem -mem-
prealloc -virtfs local,path=${HOME}/iperf_debs,mount_tag=host0,security_model=none,id=vm2_dev`

# QEMU with dpdkvhostuserclient port
    `$ sudo qemu-system-x86_64 -m 1024 -smp 1 -cpu host -hda ${HOME}/qemu/lubuntu01.qcow2 -boot c -
enable-kvm -no-reboot \
-net none -chardev socket,id=char1,path=${HOME}/ovs_prj/ovs-
software/var/run/openvswitch/dpdkvhostuserclient/ovs-vport1-socket,server  \
-netdev type=vhost-user,id=ovs-br0-vport1,chardev=char1,vhostforce     \
-device virtio-net-pci,mac=A0:01:00:00:00:00,netdev=ovs-br0-vport1     \
-object memory-backend-file,id=mem,size=1G,mem-path=/mnt/huge,share=on -numa node,memdev=mem -mem-
prealloc     \
```

```
-virtfs local,path=${HOME}/iperf_debs,mount_tag=host0,security_model=none,id=vm1_dev`

    `$ sudo qemu-system-x86_64 -m 1024 -smp 1 -cpu host -hda ${HOME}/qemu/lubuntu02.qcow2 -boot c -
enable-kvm -no-reboot     \
-net none -chardev socket,id=char2,path=${HOME}/ovs_prj/ovs-
software/var/run/openvswitch/dpdkvhostuserclient/ovs-vport2-socket,server    \
-netdev type=vhost-user,id=ovs-br0-vport2,chardev=char2,vhostforce     \
-device virtio-net-pci,mac=A0:02:00:00:00:00,netdev=ovs-br0-vport2     \
-object memory-backend-file,id=mem,size=1G,mem-path=/mnt/huge,share=on -numa node,memdev=mem -mem-
prealloc     \
-virtfs local,path=${HOME}/iperf_debs,mount_tag=host0,security_model=none,id=vm2_dev`

    Note:
    # Add & set port
    `$ sudo ovs-vsctl add-port ovs-dpdk-br0 ovs-br0-vport2 -- set Interface ovs-br0-vport2 \
    type=dpdkvhostuserclient options:vhost-server-path=${HOME}/ovs_prj/ovs-
software/var/run/openvswitch/dpdkvhostuserclient/ovs-vport2-socket`
    # OVS side: Set socket path that will be created by QEMU
    `$ sudo ovs-vsctl set Interface ovs-br0-vport1 options:vhost-server-path=${HOME}/ovs_prj/ovs-
software/var/run/openvswitch/dpdkvhostuserclient/ovs-vport1-socket`


# QEMU with tuntap port
`$ sudo qemu-system-x86_64 -m 1024 -smp 1 -cpu host -hda ${HOME}/qemu/lubuntu01.qcow2 -boot c -
enable-kvm -no-reboot \
-object memory-backend-file,id=mem,size=1G,mem-path=/mnt/huge,share=on -numa node,memdev=mem -mem-
prealloc \
-virtfs local,path=${HOME}/iperf_debs,mount_tag=host0,security_model=none,id=vm1_dev \
-netdev tap,id=mynet0,ifname=ovs-br0-vport1,script=no,downscript=no -device
e1000,netdev=mynet0,mac=A0:01:00:00:00:00`

`$ sudo qemu-system-x86_64 -m 1024 -smp 1 -cpu host -hda ${HOME}/qemu/lubuntu02.qcow2 -boot c -
enable-kvm -no-reboot \
-object memory-backend-file,id=mem,size=1G,mem-path=/mnt/huge,share=on -numa node,memdev=mem -mem-
prealloc \
-virtfs local,path=${HOME}/iperf_debs,mount_tag=host0,security_model=none,id=vm2_dev \
-netdev tap,id=vport2,ifname=ovs-br0-vport2,script=no,downscript=no -device
e1000,netdev=vport2,mac=A0:02:00:00:00:00`

`$ sudo qemu-system-x86_64 -m 1024 -smp 1 -cpu host -hda ${HOME}/qemu/lubuntu03.qcow2 -boot c -
enable-kvm -no-reboot \
-object memory-backend-file,id=mem,size=1G,mem-path=/mnt/huge,share=on -numa node,memdev=mem -mem-
prealloc \
-virtfs local,path=${HOME}/iperf_debs,mount_tag=host0,security_model=none,id=vm2_dev \
-netdev tap,id=vport3,ifname=ovs-br0-vport3,script=no,downscript=no -device
e1000,netdev=vport3,mac=A0:03:00:00:00:00`
```

5.4. Samba note

```
#Start samba sever
`$ sudo /etc/init.d/smbd start`

Note: Configure file of samba server: /etc/samba/smb.conf
```

5.5. VNC Command

```
#Start vnc server in Server 5 (172.33.47.5)
`$ vncserver -geometry 1920x1080 :8`
#Start vnc server in Server 9 (172.33.47.9)
`$ vncserver -geometry 1920x1080 -localhost no :8`

#Configure in ~/.vnc/xstartup
`$ cat ~/.vnc/xstartup`
#!/bin/bash

xrdb $HOME/.Xresources
```

```
startxfce4 &
vncconfig -iconic &

# Kill a session of VNC server
`$ sudo vncserver -kill :8`
```

5.6. Mount (SMB Share folder, hugepages)

```
# Mount SharedOne on Server 5
`$ sudo mount -t cifs -o user=hungtm,file_mode=0777,dir_mode=0777 //172.33.47.5/scratch/SharedOne
/home/hungtm/SharedOne`
# Mount SharedOne on Windows PC
`$ sudo mount -t cifs -o user=hungtm,file_mode=0777,dir_mode=0777  //172.33.41.8/SharedOne
~/SharedOne`
# Mount the hugepages, if not already mounted by default
`$ mount -t hugetlbfs none /dev/hugepages`
```

5.7. Monitor file real time

```
# Using tail tool, below is example to monitor log file of ovs-vswitchd
`$ tail -f tail -f ~/ovs_prj/ovs-software/var/log/openvswitch/ovs-vswitchd.log -n 100`
```

5.8. Linux QoS htb configure (tc tools)

```
# delete QoS htb Linux
`$ sudo tc qdisc del dev ovs-br0-vport2 root`
# Add QoS htb Linux (configure)
`$ sudo tc qdisc add dev ovs-br0-vport2 root handle 1: htb default 10`
`$ sudo tc class add dev ovs-br0-vport2 parent 1: classid 1: htb rate 50mbit`
`$ sudo tc class add dev ovs-br0-vport2 parent 1: classid 1:10 htb rate 10mbit`
`$ sudo tc class add dev ovs-br0-vport2 parent 1: classid 1:11 htb rate 20mbit`
`$ sudo tc class add dev ovs-br0-vport2 parent 1:fffe classid 1:12 htb rate 30mbit`
# Add filter for TC (QoS Linux)
`$ sudo tc filter add dev ovs-br0-vport2 parent 1: protocol ip prio 1 u32 match ip dst 192.168.2.12
flowid 1:12`
`$ sudo tc filter add dev ovs-br0-vport2 parent 1:fffe protocol ip prio 1 u32 match ip dst
192.168.2.13 flowid 1:eb`
# Show TC configuration
`$ sudo tc -g class show dev ovs-br0-vport2`
`$ sudo tc -g filter show dev ovs-br0-vport2`
`$ sudo tc -g filter show dev ovs-br0-vport2 parent 1:fffe`
```

6. OVS Commands

6.1. OVS Status & Statistic Commands

6.1.1. PMD Thread Statistics

```
Ex: show statistics of PMD thread & Main thread
`$ sudo ovs-appctl dpif-netdev/pmd-stats-show`

pmd thread numa_id 1 core_id 40:
packets received: 0
packet recirculations: 0
avg. datapath passes per packet: 0.00
emc hits: 0
smc hits: 0
megaflow hits: 0
avg. subtable lookups per megaflow hit: 0.00
miss with success upcall: 0
miss with failed upcall: 0
avg. packets per output batch: 0.00
idle cycles: 2550143615191 (100.00%)
processing cycles: 0 (0.00%)
pmd thread numa_id 1 core_id 41:
packets received: 0
packet recirculations: 0
avg. datapath passes per packet: 0.00
emc hits: 0
smc hits: 0
megaflow hits: 0
avg. subtable lookups per megaflow hit: 0.00
miss with success upcall: 0
miss with failed upcall: 0
avg. packets per output batch: 0.00
idle cycles: 2550103519751 (100.00%)
processing cycles: 0 (0.00%)
pmd thread numa_id 1 core_id 47:
packets received: 0
packet recirculations: 0
avg. datapath passes per packet: 0.00
emc hits: 0
smc hits: 0
megaflow hits: 0
avg. subtable lookups per megaflow hit: 0.00
miss with success upcall: 0
miss with failed upcall: 0
avg. packets per output batch: 0.00
idle cycles: 2550107953282 (100.00%)
processing cycles: 0 (0.00%)
main thread:
packets received: 5
packet recirculations: 0
avg. datapath passes per packet: 1.00
emc hits: 0
smc hits: 0
megaflow hits: 0
avg. subtable lookups per megaflow hit: 0.00
miss with success upcall: 4
miss with failed upcall: 1
avg. packets per output batch: 1.00

Note: can clear PMD thread statistics by command
`$ sudo ovs-appctl dpif-netdev/pmd-stats-clear`
```

6.1.2. Status & statistics information of PORT/INTERFACE

6.1.2.1. ovs-vsctl list interface

```
Ex: List interface ovs-br0-vport2
`$ sudo ovs-vsctl list interface ovs-br0-vport2`

_uuid               : 11413d80-ec41-41b9-b0a9-541b84493c0b
admin_state         : up
bfd                 : {}
bfd_status          : {}
cfm_fault           : []
cfm_fault_status    : []
cfm_flap_count      : []
cfm_health          : []
cfm_mpid            : []
cfm_remote_mpids    : []
cfm_remote_opstate  : []
duplex              : []
error               : []
external_ids        : {}
ifindex             : 12327531
ingress_policing_burst: 0
ingress_policing_rate: 0
lacp_current        : []
link_resets         : 0
link_speed          : []
link_state          : up
lldp                : {}
mac                 : []
mac_in_use          : "00:00:00:00:00:00"
mtu                 : 1500
mtu_request         : []
name                : "ovs-br0-vport2"
ofport              : 2
ofport_request      : []
options             : {}
other_config        : {}
statistics          : {"rx_1024_to_1522_packets"=0, "rx_128_to_255_packets"=15,
"rx_1523_to_max_packets"=0, "rx_1_to_64_packets"=7, "rx_256_to_511_packets"=5,
"rx_512_to_1023_packets"=0, "rx_65_to_127_packets"=8, rx_bytes=5469, rx_dropped=0, rx_errors=0,
rx_packets=35, tx_bytes=2327, tx_dropped=21, tx_packets=22}
status              : {features="0x000000017020e782", mode=server, num_of_vrings="2", numa="1",
socket="/home/hungtm/ovs_prj/ovs-software/var/run/openvswitch/ovs-br0-vport2", status=connected,
"vring_0_size"="256", "vring_1_size"="256"}
type                : dpdkvhostuser
```

6.1.2.2. ovs-ofctl dump-ports [BRIDGE NAME][PORT NAME] (The result base on OpenFlow Version)

```
Ex: Dump port ovs-br0-vport2 of bridge ovs-dpdk-br0
`$ sudo ovs-ofctl dump-ports ovs-dpdk-br0 ovs-br0-vport2`

OFPST_PORT reply (xid=0x4): 1 ports
    port  "ovs-br0-vport2": rx pkts=38, bytes=7065, drop=0, errs=0, frame=?, over=?, crc=?
            tx pkts=85061, bytes=128576913, drop=21, errs=?, coll=?

`$ sudo ovs-ofctl --protocols=OpenFlow14 dump-ports ovs-dpdk-br0 ovs-br0-vport2`    //From OpenFlow14
-> : support show more information.
OFPST_PORT reply (OF1.4) (xid=0x4): 1 ports
    port  "ovs-br0-vport2": rx pkts=0, bytes=0, drop=0, errs=0, frame=?, over=?, crc=?
            tx pkts=0, bytes=0, drop=43, errs=?, coll=?
            duration=2316.874s
            rx rfc2819 1_to_64_packets=0, 65_to_127_packets=0, 128_to_255_packets=0,
256_to_511_packets=0, 512_to_1023_packets=0, 1024_to_1522_packets=0, 1523_to_max_packets=0,
            tx rfc2819 1_to_64_packets=0, 65_to_127_packets=0, 128_to_255_packets=0,
256_to_511_packets=0, 512_to_1023_packets=0, 1024_to_1522_packets=0, 1523_to_max_packets=0,
```

6.1.2.3. ovs-ofctl dump-ports-desc

```
Ex: Dump port description of bridge ovs-dpdk-br0
`$ sudo ovs-ofctl dump-ports-desc ovs-dpdk-br0`

OFPST_PORT_DESC reply (xid=0x2):
1(ovs-br0-vport2): addr:00:00:00:00:00:00
    config:     0
    state:      LINK_DOWN
    speed: 0 Mbps now, 0 Mbps max
2(ovs-br0-vport4): addr:00:00:00:00:00:00
    config:     0
    state:      LINK_DOWN
    speed: 0 Mbps now, 0 Mbps max
3(ovs-br0-vport3): addr:00:00:00:00:00:00
    config:     0
    state:      LINK_DOWN
    speed: 0 Mbps now, 0 Mbps max
4(ovs-br0-vport1): addr:00:00:00:00:00:00
    config:     0
    state:      LINK_DOWN
    speed: 0 Mbps now, 0 Mbps max
LOCAL(ovs-dpdk-br0): addr:c6:62:10:a6:48:4c
    config:     0
    state:      0
    current:    10MB-FD COPPER
    speed: 10 Mbps now, 0 Mbps max
```

6.1.3. Tracing Packages in OVS

```
Ex: Dump package trace (actions,...) without any special open flow configuration.
`$ sudo ovs-appctl ofproto/trace ovs-dpdk-br0 in_port=ovs-br0-vport1`

Flow: in_port=4,vlan_tci=0x0000,dl_src=00:00:00:00:00:00,dl_dst=00:00:00:00:00:00,dl_type=0x0000

bridge("ovs-dpdk-br0")
----------------------
0. priority 0
    NORMAL
    -> no learned MAC for destination, flooding

Final flow: unchanged
Megaflow:
recirc_id=0,eth,in_port=4,vlan_tci=0x0000/0x1fff,dl_src=00:00:00:00:00:00,dl_dst=00:00:00:00:00:00,dl_type=

Datapath actions: 4,2,1,3

Note: Base on Datapath actions list, we will know which actions will be applied.
```

6.1.4. Show information of Open Flow

```
Ex: dump all open flow in bridge (ovs-dpdk-br0)
`$ sudo ovs-ofctl dump-flows ovs-dpdk-br0`

cookie=0x0, duration=628.952s, table=0, n_packets=2, n_bytes=180, priority=0 actions=NORMAL
```

6.2. HQoS Configuration (Defined by ATVN)

6.2.1. Configuration command is defined base on qos type=linux-htb

```
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@qospl1 -- \
--id=@qospl1 create qos type=egress-hqos other-config:cir=46000 other-config:cbs=1024 qos=@qossch1 --
\
--id=@qossch1 create qos type=egress-rrobin-scheduler other-config:delta=1
qos=@qospl21,@qospl22,@qospl23 -- \
```

```
--id=@qospl21 create qos type=queue-egress-policer-rfc2698 other-config:cir=46000 other-
config:cbs=1024 other-config:pir=46000 other-config:pbs=1024 queues:21=@queue21 -- \
--id=@queue21 create queue other-config:type=tc1 other-config:priority=10 -- \
--id=@qospl22 create qos type=queue-egress-policer-rfc2698 other-config:cir=46000 other-
config:cbs=1024 other-config:pir=46000 other-config:pbs=1024 qos=@qospl221 queues:22=@queue22 -- \
--id=@queue22 create queue other-config:type=tc2 other-config:priority=30 -- \
--id=@qospl221 create qos type=queue-egress-policer-rfc2698 other-config:cir=46000 other-
config:cbs=1024 other-config:pir=46000 other-config:pbs=1024 queues:234=@queue221 -- \
--id=@queue221 create queue other-config:type=tc2 other-config:priority=20 -- \
--id=@qospl23 create qos type=queue-egress-policer-rfc2698 other-config:cir=46000 other-
config:cbs=1024 other-config:pir=46000 other-config:pbs=1024 queues:23=@queue23 qos=@qospl231 -- \
--id=@queue23 create queue other-config:type=tc2 other-config:priority=30 --  \
--id=@qospl231 create qos type=queue-egress-policer-rfc2698 other-config:cir=46000 other-
config:cbs=1024 other-config:pir=46000 other-config:pbs=1024 queues:231=@queue231 -- \
--id=@queue231 create queue other-config:type=tc2 other-config:priority=30`

Note: The above command is defined with below assume:
+ QoS name/QoS type: draft one
+ QoS node include: policer, shaper & scheduler (identifies by QoS type)
+ Policer Node: Configuration information based on trTCM rfc2698, GREEN will be sent, YELLOW: get
from user
+ QoS node contains QoS node (original one doesn't have this one so we have to add one more QoS
column in QoS table)
+ QoS contain Queue if need
+ In this define: We don't limit number of QoS in QoS (so sub-QoS & Queue is not limitted too)
```

6.2.2. Example for “HQoS 2 level policer - type = ”egress-hqos2l-policer"

Port policer is unlimited

```
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- \
--id=@newqos create qos type=egress-hqos2l-policer other-config:cir=0 other-config:cbs=2048 other-
config:pir=31457280 other-config:pbs=2048  other-config:ydropt=true \
qos=@qospl11,@qospl12,@qospl13,@qospl14 -- \
--id=@qospl11 create qos type=egress-policer-rfc2698 other-config:cir=1024 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:1=@queue1 -- \
--id=@qospl12 create qos type=egress-policer-rfc2698 other-config:cir=2048 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:2=@queue2 -- \
--id=@qospl13 create qos type=egress-policer-rfc2698 other-config:cir=4096 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:4=@queue4 -- \
--id=@qospl14 create qos type=egress-policer-rfc2698 other-config:cir=8192 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:8=@queue8 -- \
--id=@queue1 create queue other-config:desc=rfu -- \
--id=@queue2 create queue other-config:desc=rfu -- \
--id=@queue4 create queue other-config:desc=rfu -- \
--id=@queue8 create queue other-config:desc=rfu`
```

Port Policer is trTCM rfc2698

```
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- \
--id=@newqos create qos type=egress-hqos2l-policer other-config:cir=4096 other-config:cbs=2048 other-
config:pir=31457280 other-config:pbs=2048  other-config:ydropt=true \
qos=@qospl11,@qospl12,@qospl13,@qospl14 -- \
--id=@qospl11 create qos type=egress-policer-rfc2698 other-config:cir=1024 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:1=@queue1 -- \
--id=@qospl12 create qos type=egress-policer-rfc2698 other-config:cir=2048 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:2=@queue2 -- \
--id=@qospl13 create qos type=egress-policer-rfc2698 other-config:cir=4096 other-config:cbs=2048
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:4=@queue4 -- \
--id=@qospl14 create qos type=egress-policer-rfc2698 other-config:cir=8192 other-config:cbs=2048
```

```
other-config:pir=10240 other-config:pbs=2048  other-config:ydropt=true \
queues:8=@queue8 -- \
--id=@queue1 create queue other-config:desc=rfu -- \
--id=@queue2 create queue other-config:desc=rfu -- \
--id=@queue4 create queue other-config:desc=rfu -- \
--id=@queue8 create queue other-config:desc=rfu`


Note: This command is using now! based on below assume:
+ with QoS type == egress-hqos2l-policer (port level): cir == 0 is unlimited.
+ with other QoS node (Not port level): cir == 0, configuration returns error code base one
standard "trTCM rfc2698"
+ base on standard "trTCM rfc2698": cir have to less than pir otherwise configuration will return
error code.
+ Queue's other-config: desc=rfu (rfu is resever for future use)
+ Number of Queue is not limitted.
```

6.3. Vlog user command line

```
# list out all log module information.
`$ sudo ovs-appctl coverage/show`   //show coverage counters
`$ sudo ovs-appctl vlog/reopen`  // reopen log file
`$ sudo ovs-appctl vlog/list`    // list out all information of vlog
#set a few module to DBG level for all: console, syslog, file
`$ sudo ovs-appctl vlog/set netdev_dpdk:ANY:DBG` //set netdev_dpdk module to DBG level for all:
console, syslog, file
`$ sudo ovs-appctl vlog/set dpif_netdev:ANY:DBG`
`$ sudo ovs-appctl vlog/set bridge:ANY:DBG`
`$ sudo ovs-appctl vlog/set hqos:ANY:DBG`   //hqos is new module (defined by atvn)
# Using tail tool, below is example to monitor log file of ovs-vswitchd
`$ tail -f tail -f ~/ovs_prj/ovs-software/var/log/openvswitch/ovs-vswitchd.log -n 100`
```

6.4. MAC Address Table

```
`$ sudo ovs-appctl fdb/show ovs-br0`    //show mac address table of bridge that's name is ovs-br0
```

6.5. QoS user command line

```
# list out supported QoS type of port (already in use)
`$ sudo ovs-appctl -t ovs-vswitchd qos/show-types ovs-br0-vport2`

`$ sudo ovs-appctl coverage/show`   //show coverage counters

# Configure QoS egress policing (already in use for egress policer srTCM)
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- --id=@newqos create qos type=egress-policer
other-config:cir=46000000 other-config:cbs=2048`

# Examine the QoS configuration of port (already in use)
`$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2`

# clear or destroy QoS configuration of port (already in use)
`$ sudo ovs-vsctl destroy QoS ovs-br0-vport2 -- clear Port ovs-br0-vport2 qos`

# OVS database command
`$ sudo ovsdb-client list-tables` //list all table in database
`$ sudo ovsdb-client dump | grep table`   //List all table in database
`$ sudo ovsdb-client list-column`   //or
`$ sudo ovsdb-client list-columns Port` //list all columns

`$ sudo ovsdb-client monitor ALL` //Monitor all table in DATABASE on SERVER
`$ sudo ovsdb-client monitor QoS` //Monitor table QoS in DATABASE on SERVER
`$ sudo ovsdb-client monitor Queue` //Monitor table Queue in DATABASE on SERVER

`$ sudo ovs-vsctl list qos` //List all qos table information
```

```
`$ sudo ovs-vsctl list queue` //List all queue table information

`$ sudo ovs-vsctl  --  destroy qos _uuid`   //delete qos table row
`$ sudo ovs-vsctl  --  destroy queue _uuid`
`$ sudo ovs-vsctl --all destroy qos`    //delete all qos table rows
`$ sudo ovs-vsctl --all destroy queue`  //delete all queue table rows

# delete all queue ( QoS linux-htb user command)
`$ ovs-vsctl -- --all destroy QoS -- --all destroy Queue`

# Examine the ingress policing of port.
`$ sudo ovs-vsctl list interface ovs-br0-vport1`

# Set ingress policing of port
`$ sudo ovs-vsctl set interface ovs-br0-vport1 ingress_policing_rate=1024 ingress_policing_burst=128`

# clear ingress policing of port
`$ sudo ovs-vsctl set interface ovs-br0-vport1 ingress_policing_rate=0`

# QoS linux-htb (rate: bits/second - bps)
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- \
--id=@newqos create qos type=linux-htb other-config:max-rate=1000000000 queues:123=@vif10queue
queues:234=@vif20queue -- \
--id=@vif10queue create queue other-config:max-rate=10000000 -- \
--id=@vif20queue create queue other-config:max-rate=20000000

$ sudo ovs-ofctl add-flow ovs-br0 cookie=0x11,in_port="ovs-br0-vport1",actions=set_queue:123,normal
$ sudo ovs-ofctl add-flow ovs-br0 cookie=0x12,in_port="ovs-br0-vport3",actions=set_queue:234,normal
```

7. Queuing and Scheduling Concepts

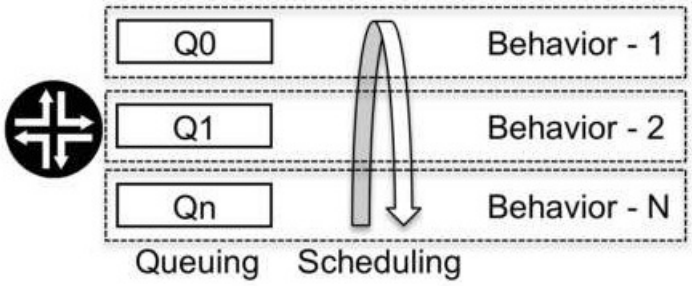


Figure 7.1 Queuing and scheduling applying different behaviors

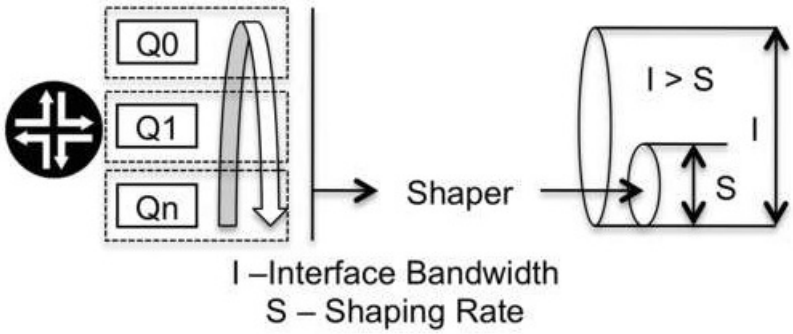


Figure 7.2 Bandwidth parameter in queuing and scheduling

Figure 7-1 - Queuing and Scheduling

- The two most important parameters associated with the queuing and scheduling mechanism are buffers and bandwidth
- Buffering is the length of the queue, that is, how much memory is available to store packets
- Scheduling determines how much is allocated to each queue. The total amount of bandwidth can be either the interface speed or the shaping rate if a shaper is applied after the scheduler

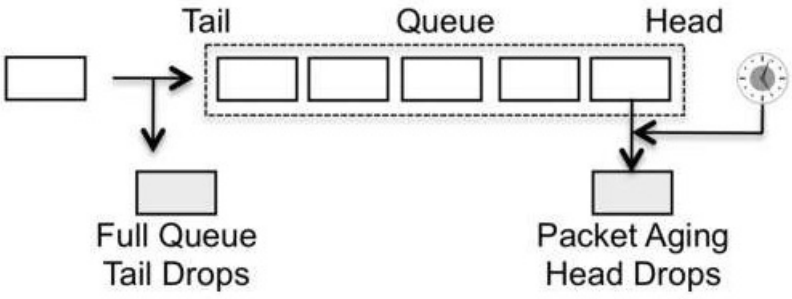


Figure 7.3 Tail and head aging drops in a queue

Figure 7-2 - Dropped Package In Queue

- package can be dropped at head becaused of aged out and tail because of full queue

Fore more information, refer **link**

8. *.ovsschema file (Json format)

Target

```
+ Can clear & add more columns into *.ovsschema file
```

view online Json Viewer

8.1. How to del or add columns in ovsdb (update vswitch.ovsschema)

To update (del/add) columns in ovsdb, we have to update vswitch.ovsschema. Below is step by step to do it (Example: we will update qos column in QoS)

Step 1 Update vswitch.ovsschema (**${OVS source code}/vswitchd/vswitch.ovsschema**)

```
#Add below code into the vswitch.ovsschema (in QoS table)
'''
"qos": {
     "type": {"key": {"type": "uuid",
                      "refTable": "QoS"},
              "min": 0, "max": 1}},
'''
The result look like below:
'''
"QoS": {
 "columns": {
   "type": {
     "type": "string"},
   "qos": {
     "type": {"key": {"type": "uuid",
                      "refTable": "QoS"},
              "min": 0, "max": 1}},
   "queues": {
       ...
'''
```

Step 2 Update documents for new columns in vswitch.xml (**${OVS source code}/vswitchd/vswitch.xml**)

```
#Add documents of new column in QoS table (column "qos")
'''
<column name="qos">
  Try defining QoS contain QoS (just for testing first).
</column>
'''
The result:
```

```
<table name="QoS" title="Quality of Service configuration">
  <p>Quality of Service (QoS) configuration for each Port that
  references it.</p>

  <column name="type">
  <column name="qos">
    Try defining QoS contain QoS (just for testing first).
  </column>
  <column name="queues">
```

Figure 8-1 - Update document of new column

Step 3 update cksum in vswitch.ovsschema (**${OVS source code}/vswitchd/vswitch.ovsschema**)

```
#Check or Caculate cksum by command
`$ ./build-aux/cksum-schema-check vswitchd/vswitch.ovsschema vswitchd/vswitch.ovsschema.stamp`
```

```
vswitchd/vswitch.ovsschema:3: The checksum "378462745 24004" was calculated from the schema file and
does not match cksum field in the schema file - you should probably update the version number and
the checksum in the schema file with the value listed here.

Note: update the new checksum __378462745 24004__ into the *.ovsschema file
```

Step 4 rebuild, install & configure again for new vswitch.ovsschema

```
We have to delete & create conf.db file again (TBC)
```

9. HQoS

Hierarchical Quality of Service (HQoS) organaizes a scheduler policy into a hierarchical tree that consists of a root node, branches node, and leaf node, where:

- The root node is the convergence point for all traffics and corresponds to a scheduler followed by a traffics shaper. The root node schedules and shapes is the aggregated egress traffic of a physical port.
- Branches node is located in the middle of the hierarchy and corresponds to a scheduler followed by a traffic shaper.
- A leaf node corresponds to a scheduling queue.

9.1. Question???

normalization changed ofp_match issue ?

Q: I ran “ovs-ofctl add-flow br0 nw_dst=192.168.0.1,actions=drop” but I got a funny message like this:

```
ofp_util|INFO|normalization changed ofp_match, details:
ofp_util|INFO| pre: nw_dst=192.168.0.1
ofp_util|INFO|post:
```

and when I ran “ovs-ofctl dump-flows br0” I saw that my nw_dst match had disappeared, so that the flow ends up matching every packet.

A: The term “normalization” in the log message means that a flow cannot match on an L3 field without saying what L3 protocol is in use. The “ovs-ofctl” command above didn’t specify an L3 protocol, so the L3 field match was dropped.

In this case, the L3 protocol could be IP or ARP. A correct command for each possibility is, respectively:

```
ovs-ofctl add-flow br0 ip,nw_dst=192.168.0.1,actions=drop
```

and

```
ovs-ofctl add-flow br0 arp,nw_dst=192.168.0.1,actions=drop
```

Similarly, a flow cannot match on an L4 field without saying what L4 protocol is in use. For example, the flow match “tp_src=1234” is, by itself, meaningless and will be ignored. Instead, to match TCP source port 1234, write “tcp,tp_src=1234”, or to match UDP source port 1234, write “udp,tp_src=1234”.

Is it the place where the upcall handling really happening?

Yes, for the kernel datapath. For userspace datapath, you’ll be more interested in upcall_cb(). The two meet at upcall_receive().

When I try to debug the handler with gdb, “recv_upcalls” is not called for the new packet in/flow insert (Please note the number of handler and revalidator threads are set to 1) This is a statement. **How can I adjust the timeout for revalidator and handler to debug the threads with gdb. Sometimes the threads are crashing because of it.**

Which timeout do you mean? If threads are crashing, then gdb should help you to find out why.

What is VLAN???

What is CoS (class of service)???

10. OvS deep dive note

10.1. OvS without DPDK datapath (TBU)

10.2. OvS with DPDK datapath (TBU)

10.3. QoS feature of OvS without DPDK (TBU)

Open vSwitch does not implement QoS itself. Instead, it can configure some, but not all, of the QoS features built into the **Linux kernel**.

For more information about configure & use it, please refer **link**

10.4. QoS feature of OvS with DPDK

At writing time, OvS+DPDK just support **`egress policing`** and **`rate limiting (ingress policing)`** only.

We will focus on setting & use it in session **Model Testing/OvS with DPDK QoS Testing** of this note.

For more information, please refer **link**.

11. OVS Basic Note

11.1. Overview Of Open vSwitch with DPDK

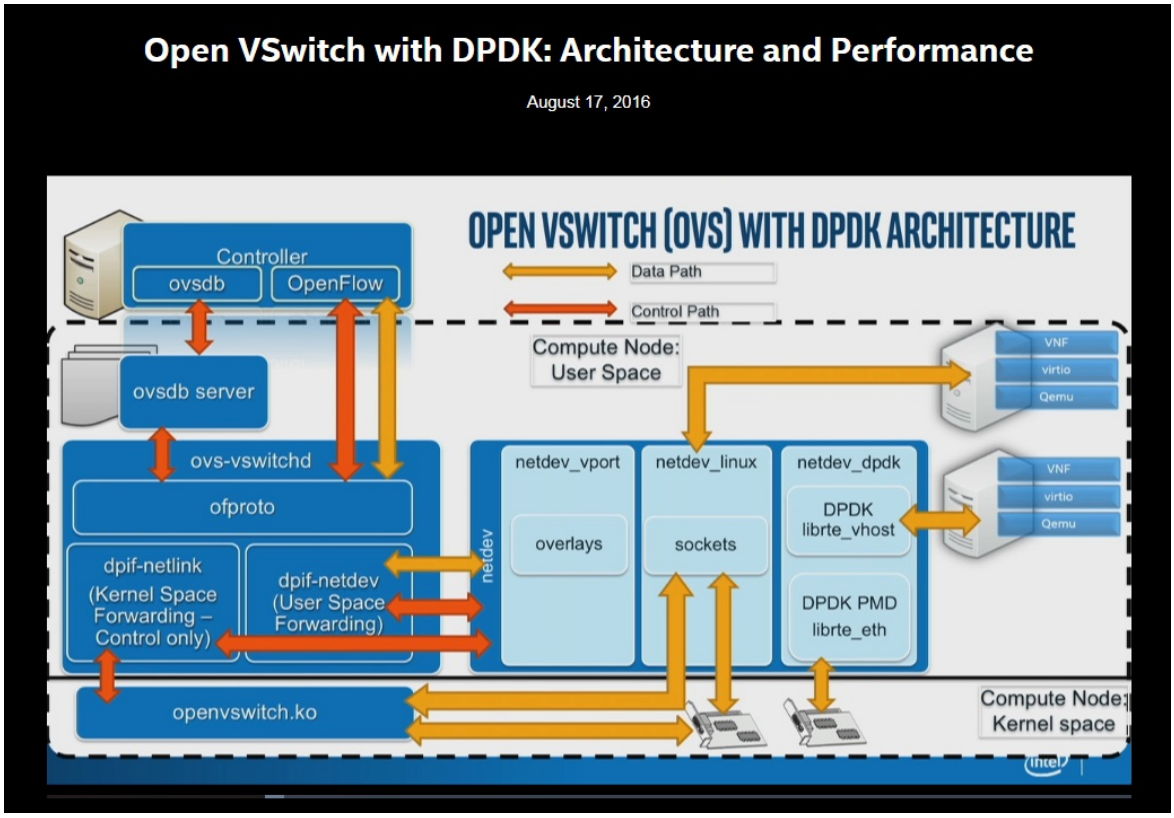


Figure 11-1 - OVS + DPDK Architecture

11.2. Build & Install Open vSwitch with DPDK in UBUNTU (18.02 LTS X86_64)

In this note, we built openvswitch–2.11.1 and dpdk-stable–18.11.2 in ubuntu 18.02 LTS x86_64, gcc 7.4.0

11.2.1. Build & Install DPDK

Goto DPDK source directory

```
$ make config T=x86_64-native-linuxapp-gcc #configure Target to build

$ make DESTDIR=[path of DPDK SDK folder] prefix=[prefix] #default value of prefix=/usr/local

$ make install DESTDIR=[path/of/DPDK-SDK/folder]
```

Note:

```
For more options of make file, please use `make help` command, The DPDK SDK will be allocated at
[path of DPDK SDK folder]/[prefix]
```

11.2.2. Build & Install OVS with DPDK

Goto OVS source directory

```
$ ./boot.sh
$ ./configure --prefix=[path/to/install/lib,binary,../of/ovs] --with-dpdk=
[path/of/DPDK-SDK]/share/dpdk/x86_64-native-linuxapp-gcc
$ make
```

```
$ make install
```

Note:

```
+ The ovs software will be stored in [path/to/install/lib,binary,../of/ovs]
+ For more information about configuration of OVS, please use `configure --help`
```

11.2.3. Build & Install igb_uio driver (provided by DPDK) (TBU)

11.3. Setup NIC, Using igb_uio driver (compatible with dpdk)

11.3.1. Setup NIC with with dpdk-devbind.py script.

dpdk-devbind.py is provided by DPDK SDK at **dpdk-stable-18.11.2/usertools/** folder.

```
To display current device status:
    `dpdk-devbind.py --status`

To display current network device status:
    `dpdk-devbind.py --status-dev net`

To bind eth1 from the current driver and move to use igb_uio
    `dpdk-devbind.py --bind=igb_uio eth1`

To unbind 0000:01:00.0 from using any driver
    `dpdk-devbind.py -u 0000:01:00.0`

To bind 0000:02:00.0 and 0000:02:00.1 to the ixgbe kernel driver
    `dpdk-devbind.py -b ixgbe 02:00.0 02:00.1`
```

For more information of this script, we can show help message by **dpdk-devbind.py --help**. Below is example to setup NIC in Server 5: 82574L Gigabit Network (pci address: 0000:02:00.0)

using **dpdk-devbind.py --status** to list out status of NIC in server 5

```
$ dpdk-devbind.py --status

Network devices using kernel driver
===================================
0000:00:19.0 '82579LM Gigabit Network Connection (Lewisville) 1502' if=eno1 drv=e1000e
unused=igb_uio,uio_pci_generic *Active*

Other Network devices
=====================
0000:02:00.0 '82574L Gigabit Network Connection 10d3' unused=e1000e,igb_uio,uio_pci_generic

No 'Crypto' devices detected
============================
```

In the result, don't have any NIC compatible with DPDK, use the script to bind **82574L Gigabit Network** NIC

```
$ sudo dpdk-devbind.py --bind=igb_uio 0000:02:00.0
$ dpdk-devbind.py --status

Network devices using DPDK-compatible driver
============================================
0000:02:00.0 '82574L Gigabit Network Connection 10d3' drv=igb_uio unused=e1000e,uio_pci_generic

Network devices using kernel driver
===================================
0000:00:19.0 '82579LM Gigabit Network Connection (Lewisville) 1502' if=eno1 drv=e1000e
```

```
unused=igb_uio,uio_pci_generic *Active*
```

Note:

```
igb_uio also provided by DPDK SDK, please check it in
{dpdk_source_code_path}/x86_64-native-linuxapp-gcc/kmod/igb_uio.ko
```

11.3.2. Setup NIC with dpdk-setup.sh script (TBU)

11.4. Setup hugepage in UBUNTU (18.02 LTS X86_64)

11.4.1. Setup manually (TBU)

11.4.2. Setup using dpdk-setup.sh script (TBU)

11.5. Configure & launch OVS+DPDK

11.5.1. Configure & launch OVS+DPDK manually

In this session, we tested OVS+DPDK on Server 9 (2 NUMA Node: 0 & 1)

- with huge_page, HugePages_Free: 7737 (using **`$ cat /proc/meminfo`** to check)
- pid file, log file will be used default.

Below is step by step to launch OVS+DPDK

Step 1: check & create database **`conf.db`** if it is not exist. The command to create the database

```
$ sudo ovsdb-tool create /etc/openvswitch/conf.db /share/openvswitch/vswitch.ovsschema
```

Step 2: start ovsdb-server

```
$ sudo ovsdb-server --remote=punix:/var/run/openvswitch/db.sock  -
-remote=db:Open_vSwitch,Open_vSwitch,manager_options --pidfile --detach --log-file
```

Step 3: Configure DPDK related parameter

```
#Initialize & support DPDK port in OVS (default is false)
$ sudo ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-init=true

#Specifies the CPU cores on which dpdk lcore threads should be spawned
$ sudo ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-lcore-mask="0x03"

#if we have more than one numa node, should initialize both
$ sudo ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-socket-mem="2048,2048"

#multiple PMD threads can be created and pinned to CPU cores by explicitly specifying pmd-cpu-masks
$ sudo ovs-vsctl --no-wait set Open_vSwitch . other_config:pmd-cpu-mask=0x3
```

Step 4: start ovs_vswitchd

```
$ sudo ovs-vswitchd --pidfile --detach --log-file
```

11.5.2. Configure & launch OVS+DPDK automatically by ovs_mini_tools.py script (TBU)

11.6. Initialize OvS+DPDK Bridge

11.6.1. Initialize OvS+DPDK Bridge manually (TBU)

In this session, we focus on how to add OvS+DPDK bridge (**ovsdb-server** & **ovs_vswitchd** have to started)

- NIC is Chelsio with PCI address 0000:03:00.4 (already bind with igb_uio drvier - compatible with DPDK)
- bridge with **datapath_type** is **netdev** (support DPDK ports)
- physical port with **type** is **dpdk** (connect to NIC)
- vhost user port with **type** is **dpdkvhostuser** (connect to virtual machine)

To check current bridge in system, we use command **$ sudo ovs-vsctl show**

```
$ sudo ovs-vsctl show
ec3086f5-67ac-4c14-9b47-ea7520235faf
```

For the first time, database is empty so we don't have any bridge as above. Below is step by step to add a OvS+DPDK bridge

Step 1: Add bridge (example: bridge name is **ovs-dpdk-br0**)

```
$ sudo ovs-vsctl add-br ovs-dpdk-br0 -- set bridge ovs-dpdk-br0 datapath_type=netdev

Note: if we already had bridge `ovs-dpdk-br0` with another datapath_type (check by command `$ sudo
ovs-vsctl get bridge ovs-dpdk-br0 datapath_type`), we can set datapath_type by command
$ sudo ovs-vsctl set bridge ovs-dpdk-br0 datapath_type=netdev
```

Step 2: Add physical port (physical port name **ovs-br0-pport1**)

```
$ sudo ovs-vsctl add-port ovs-dpdk-br0 ovs-br0-pport1 --set Interface ovs-br0-pport1 type=dpdk
options:dpdk-devargs=0000:03:00.4

Note: loop for others physical ports
```

Step 3: Add vhost user port (port name is **ovs-br0-vport1**)

```
#Add interface into bridge `ovs-dpdk-br0`
$sudo ovs-vsctl add-port ovs-dpdk-br0 ovs-br0-vport1 -- set Interface ovs-br0-vport1
type=dpdkvhostuser

Note: loop for others ports
```

After configure sucessfully, we can check by command **ovs-vsctl show**

```
$ sudo ovs-vsctl show
ec3086f5-67ac-4c14-9b47-ea7520235faf
Bridge "ovs-dpdk-br0"
    Port "ovs-br0-vport3"
        Interface "ovs-br0-vport3"
            type: dpdkvhostuser
    Port "ovs-br0-vport2"
        Interface "ovs-br0-vport2"
            type: dpdkvhostuser
    Port "ovs-br0-pport1"
        Interface "ovs-br0-pport1"
            type: dpdk
            options: {dpdk-devargs="0000:03:00.4"}
    Port "ovs-br0-vport1"
        Interface "ovs-br0-vport1"
            type: dpdkvhostuser
    Port "ovs-dpdk-br0"
```

```
        Interface "ovs-dpdk-br0"
            type: internal
    Port "ovs-br0-vport4"
        Interface "ovs-br0-vport4"
            type: dpdkvhostuser
```

Note:

- If we got a error : “could not open network device ovs-br0-vport* (Unknown error –1)” when show bridge information by **$sudo ovs-vsctl show** as below

```
Port "ovs-br0-vport3"
    Interface "ovs-br0-vport3"
        type: dpdkvhostuser
        error: "could not open network device ovs-br0-vport3 (Unknown error -1)"
```

  Try fixing it by remove “port file” in ${ovs-software}/var/run/openvswitch that will be created when we add dpdkvhostuser port (just workaround).

- In the result, we have a **internal port** that has **type: internal**, it is used for host. without it, the host will lose connection. Different with other ports, it is L3 port, we have to setup a IP address (example: 192.168.2.1) for it by command $sudo ifconfig ovs-dpdk-br0 192.168.2.1 up

For more detail about **internal port**, please refer **link**

11.6.2. Initialize OvS+DPDK bridge automatically by ovs_mini_tools.py script (TBU)

11.7. Add virtual machine into OvS+DPDK Bridge (Using QEMU)

11.7.1. Install and Configure KVM on Ubuntu 18.04 LTS

Verify Whether our system support hardware virtualization by **kvm-ok** command, if the system support hardware virtualization, the result should like below:

```
$sudo kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used
```

Run the below commands to install KVM and its dependencies

```
$sudo apt install qemu qemu-kvm libvirt-bin  bridge-utils  virt-manager
```

11.7.2. Start virtual machine (VM) & connect it with our bridge

Start VM that is connected with our bridge with below informations

- Ram 1024M (-m 1024), CPU core 1 (-smp 1), image file lubuntu01.qcow2, mac address **A0:01:00:00:00:00**,....
- Connected with our bridge by port **ovs-br0-vport1**
- For other information of command, please refer munual of **qemu-system-x86_64**.

The command to start virtual machine as below:

```
$ sudo qemu-system-x86_64 -m 1024 -smp 1 -cpu host -hda ${HOME}/qemu/lubuntu01.qcow2 -boot c
-enable-kvm -no-reboot -net none -chardev
socket,id=char1,path=${HOME}/ovs_prj/ovs-software/var/run/openvswitch/ovs-br0-vport1 -netdev
type=vhost-user,id=ovs-br0-vport1,chardev=char1,vhostforce -device
virtio-net-pci,mac=A0:01:00:00:00:00,netdev=ovs-br0-vport1 -object
memory-backend-file,id=mem,size=1G,mem-path=/mnt/huge,share=on -numa node,memdev=mem -mem-prealloc
-virtfs local,path=${HOME}/iperf_debs,mount_tag=host0,security_model=none,id=vm1_dev
```

```
Note:
+ After the VM fully start up, check its IP address. if it doesn't have IP address, please setup
it manually by `ifconfig` command. The subnet Ip has to same with `internal port`
+ We can add more VM in other ports, example below is add VM2 into our bridge via `ovs-br0-vport2`
port
$ sudo qemu-system-x86_64 -m 1024 -smp 1 -cpu host -hda ${HOME}/qemu/lubuntuuu.qcow2 -boot c -
enable-kvm -no-reboot -net none -chardev socket,id=char2,path=${HOME}/ovs_prj/ovs-
software/var/run/openvswitch/ovs-br0-vport2 -netdev type=vhost-user,id=ovs-br0-
vport2,chardev=char2,vhostforce -device virtio-net-pci,mac=A0:02:00:00:00:00,netdev=ovs-br0-vport2 -
object memory-backend-file,id=mem,size=1G,mem-path=/mnt/huge,share=on -numa node,memdev=mem -mem-
prealloc -virtfs local,path=${HOME}/iperf_debs,mount_tag=host0,security_model=none,id=vm2_dev
```

For more information about **DPDK vhost user port** & how to add **DPDK vhost user port** to the guest (qemu), please refer **link**

12. OVS Model Testing

12.1. Model I - 2VM connect with OvS bridge (without DPDK)

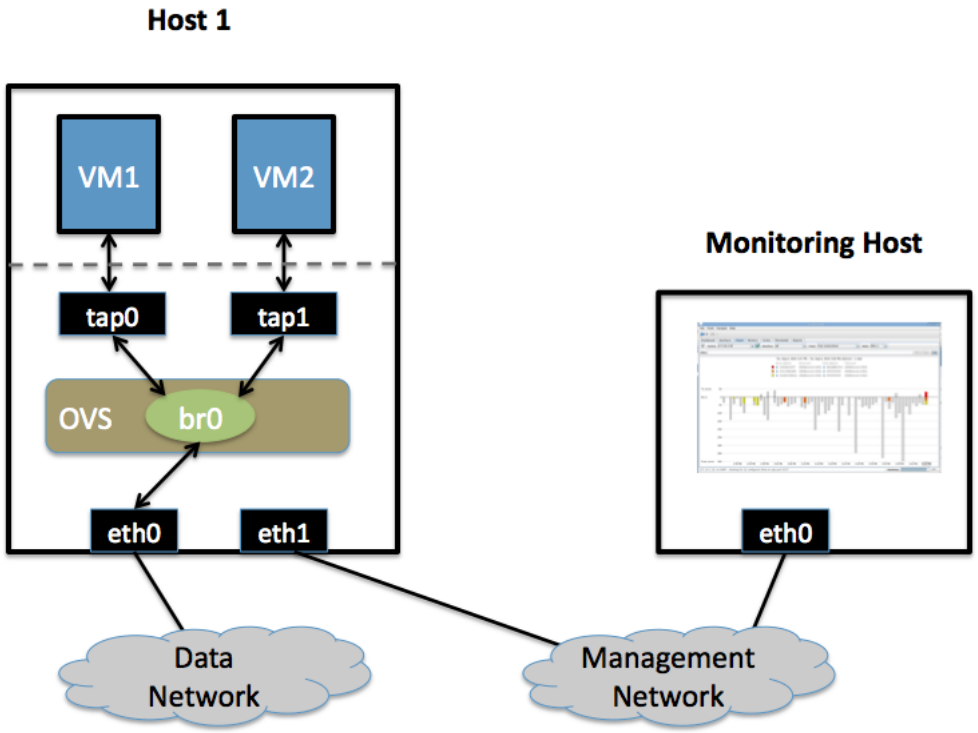


Figure 12-1 - Model I

12.1.1. Target

```
+ 2VM connect to bridge.
+ use trafgen to send out package.
+ use tshark to trace the received package
+ use sFlowTrend to monitor
+ Add more VM to bridge & checking
```

12.1.2. Setup (TBU)

12.1.3. Testing (TBU)

12.2. Model II - 2VM connect with OvS+DPDK bridge

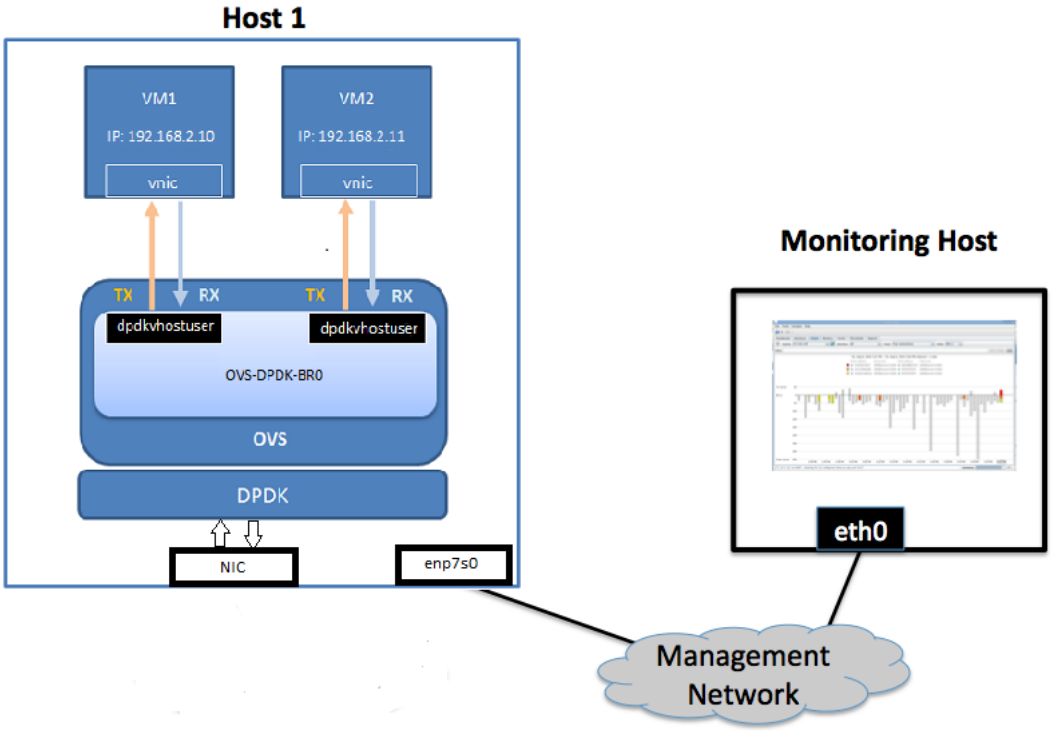


Figure 12-2 - Model II

12.2.1. Target:

```
+ 2VM connect to bridge.
+ use trafgen to send out package.
+ use tshark to trace the received package
+ use sFlowTrend to monitor
+ Add more VM to bridge & checking
```

12.2.2. Setup:

Step 1: Build & install OVS+DPDK (refer session **Build & Install Open vSwitch With DPDK in UBUNTU (18.02 LTS X86_64)**)

Step 2: Add OVS+DPDK bridge with 1 physical port `ovs-br0-pport1` and 2 vhost user ports `ovs-br0-vport1 & ovs-br0-vport2` (refer session **Initialize OvS+DPDK Bridge**)

Step 3: Start VM1 connect with bridge via `ovs-br0-vport1` & Start VM2 connect with bridge via `ovs-br0-vport2` (refer **Start virtual machine (VM) & connect it with our bridge**)

Step 4: Check connection between them by `ping` command.

Note:

- if every thing is correct, we will have a model with 2 VM connect with our bridge, if we want to have more than 2 VM, in the `Step 3`, add more VM then check connection between them by `ping` command.
- Other later steps are optional, dependent on your purpose.

Step 5: configure `trafgen` in our VMs (refer **trafgen**)

Step 6: Configure sFlowTrend to monitor (refer **sFlowTrend**)

12.2.3. Testing:

```
# Start tshark on VM1 & VM2 by command
```

```
$ sudo tshark

#On VM1, open terminal & start trafgen to send 10000 pakages to VM2
$ sudo trafgen --dev ens3 --cpp --conf trafgen.cfg -n10000

#On VM2, open terminal & start trafgen to send 10000 pakages to VM1
$ sudo trafgen --dev ens3 --cpp --conf trafgen.cfg -n10000

#Check the result on `tshark` terminal & sFlowTrend on Monitor host
```

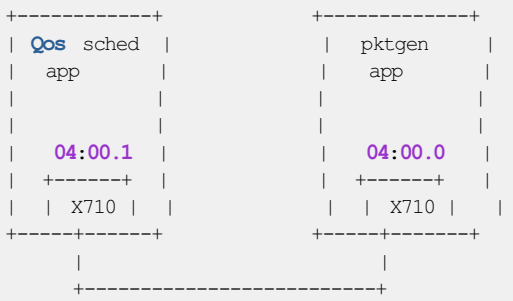
12.3. Model III: DPDK TM with pktgen and qos_shed sample application

12.3.1. Model

Environment +

- OS: Linux 18.04
- DPDK: v18.11
- pktgen: pktgen–3.6.6
- HW: Dual ports Intel Corporation Ethernet Controller X710 for 10GbE SFP+
  - 0000:04:00.0 ‘Ethernet Controller X710 for 10GbE SFP+ 1572’ drv=igb_uio unused=i40e
  - 0000:04:00.1 ‘Ethernet Controller X710 for 10GbE SFP+ 1572’ drv=igb_uio unused=i40e

Model:



12.3.2. Setup

12.3.2.1. DPDK

```
cd /path/to/dpdk/repo
export RTE_SDK=`pwd`
export RTE_TARGET=x86_64-native-linuxapp-gcc
cd $RTE_SDK
git checkout v18.11
make install T=x86_64-native-linuxapp-gcc DESTDIR=$RTE_SDK/sdk -j
```

12.3.2.2. QOS sched example

```
cd $RTE_SDK/examples/qos_sched
make
```

12.3.2.3. pktgen

```
cd ${WD}
git clone http://dpdk.org/git/apps/pktgen-dpdk
cd ${WD}/pktgen-dpdk
```

```
git checkout pktgen-3.6.6
make -j8
```

12.3.3. Run

12.3.3.1. Bind 2 port to DPDK driver

```
cd $RTE_SDK
sudo rmmod igb_uio
sudo modprobe uio
sudo insmod ${RTE_TARGET}/kmod/igb_uio.ko
sudo ./usertools/dpdk-devbind.py -b igb_uio 0000:04:00.0 0000:04:00.1
```

12.3.3.2. pktgen

```
cd ${WD}/pktgen-dpdk
sudo -E ./app/x86_64-native-linuxapp-gcc/pktgen -l 0-4 -n 4 --proc-type auto --log-level 7 -
-file-prefix pg -w 0000:04:00.1 -- -T -P -m [1:2].0
```

Use cmd:

- **start 0** to send packet
- **set 0 rate ${value}**: to set port rate with ${value} in percentage
- **set 0 size ${value}**: set packet size

12.3.3.3. DPDK QOS sched example

```
cd $RTE_SDK/examples/qos_sched
sudo ./build/qos_sched -l 1,5,7 -n 4 --file-prefix p1 -w 0000:04:00.0 -- --pfc "0,0,5,7" --mst 1 -
-cfg ./profile.cfg
```

Command explain:

- **--file-prefix**: for multi dpdk application
- **-w 0000:04:00.0**: run application with port **0000:04:00.0**
- **--pfc "0,0,5,7"**: this is application param
  - RX port: 0
  - Tx port: 0
  - Rx LCORE: 5
  - WT LCORE: 7
- **--mst 1**: set master lcore to 1

12.4. OvS with DPDK QoS testing

12.4.1. Target:

```
+ Use Model II - 2VM connect with OvS+DPDK bridge
+ Use iperf to check the QoS features of OvS+DPDK
+ Find the way to show data rate in OvS+DPDK bridge (perspective of the switch)
+ Add more VM & check again
```

12.4.2. Setup

Base on **Model II**, Setup 2 VM connect with our bridge. No need setup for trafgen & sFlowTrend (refer **Model II - 2VM connect with OvS+DPDK bridge/Setup**). Below is example configuration of testing

Internal port IP address: 192.168.2.1

VM1:

```
+ vhost user port: ovs-br0-vport1
+ Ip address: 192.168.2.10
```

VM2:

```
+ vhost user port: ovs-br0-vport2
+ Ip address: 192.168.2.11
```

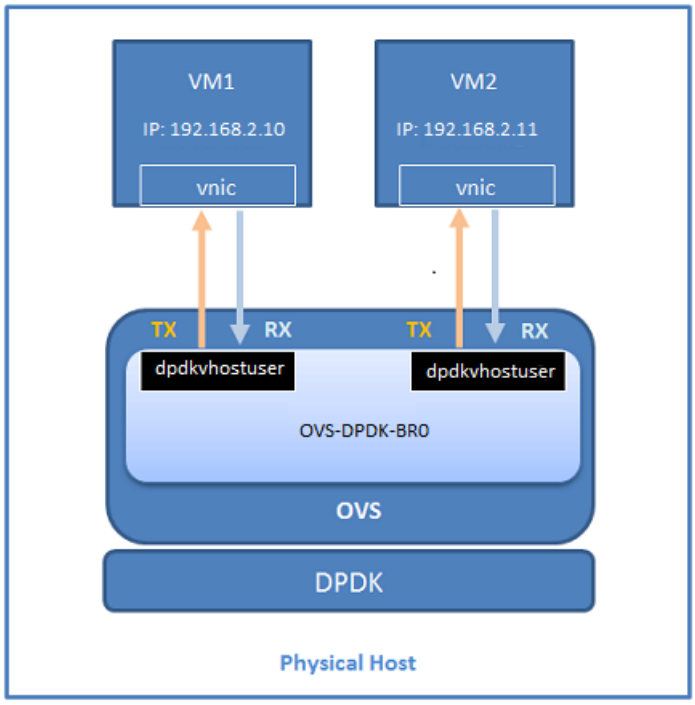


Figure 12-3 - Model II OVS+DPDK QoS Tesing

12.4.3. Testing

A list of supported QoS types for a given port (for example, ovs-br0-vport2) can be obtained with the following command

```
$ sudo ovs-appctl -t ovs-vswitchd qos/show-types ovs-br0-vport2
QoS type: egress-policer
QoS type: egress-policer-rfc4115(TODO)
QoS type: egress-policer-rfc2698

Note:

+ At writing time, the OVS+DPDK just support egress-policer (srTCM rfc2697).

+ The result as above, egress-policer-rfc2698 is added by ATVN. The OVS source code that support
this one can be checkout at [ovs feature/qos](http://gitlab.atvn.com.vn/ovs/ovs) branch
__feature/qos__
```

12.4.3.1. Testing without QoS

```
#Check & make sure the egress policing is not configured for `ovs-br0-vport2` port (we use VM2 as
```

```
__iperf server__ so we will test egress policing in `ovs-br0-vport2`)
$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2
QoS not configured on ovs-br0-vport2

#Note
+ if QoS already configured on ovs-br0-vport2, disable it by command `$ sudo ovs-vsctl -- destroy
QoS ovs-br0-vport2 -- clear Port ovs-br0-vport2 qos`

#Check & make sure the __Rate limiting (QoS ingress policing)__ is not configured for `ovs-br0-
vport1` (we use VM1 as __iperf client__ so we will test ingress policing in `ovs-br0-vport1`)
$ sudo ovs-vsctl list interface ovs-br0-vport1
_uuid               : c6a57b70-6df2-491b-830d-f15c7a0686a6
admin_state         : up
bfd                 : {}
bfd_status          : {}
cfm_fault           : []
cfm_fault_status    : []
cfm_flap_count      : []
cfm_health          : []
cfm_mpid            : []
cfm_remote_mpids    : []
cfm_remote_opstate  : []
duplex              : []
error               : []
external_ids        : {}
ifindex             : 3159192
ingress_policing_burst: 0
ingress_policing_rate: 0
lacp_current        : []
link_resets         : 0
link_speed          : []
link_state          : up
lldp                : {}
mac                 : []
mac_in_use          : "00:00:00:00:00:00"
mtu                 : 1500
mtu_request         : []
name                : "ovs-br0-vport1"
ofport              : 2
ofport_request      : []
options             : {}
other_config        : {}
statistics          : {"rx_1024_to_1522_packets"=0, "rx_128_to_255_packets"=46,
"rx_1523_to_max_packets"=0, "rx_1_to_64_packets"=19, "rx_256_to_511_packets"=27,
"rx_512_to_1023_packets"=0, "rx_65_to_127_packets"=33, rx_bytes=22060, rx_dropped=0, rx_errors=0,
rx_packets=125, tx_bytes=219289, tx_dropped=14326, tx_packets=1047}
status              : {features="0x000000017020e782", mode=server, num_of_vrings="2", numa="0",
socket="/home/hungtm/ovs_prj/ovs-software/var/run/openvswitch/ovs-br0-vport1", status=connected,
"vring_0_size"="256", "vring_1_size"="256"}
type                : dpdkvhostuser

#Note:
+ In the result, `ingress_policing_rate: 0` is mean `Rate limiting` is disable.
+ if ingress_policing_rate already configured, disable it by command `$ sudo ovs-vsctl set interface
ovs-br0-vport2  ingress_policing_rate=0`

#Start __iperf server__ on VM2
$ iperf -s -u -p 8080

#Start __iperf client__ on VM1
$ iperf -c 192.168.2.11 -u -p 8080 -b 100m
```

Waiting for **iperf** on VM1 finish, Bandwidth is 100Mbits/sec on both of side as below:

On VM1 (iperf client side):

```
hungtm@hungtm-VirtualBox:~$ iperf -c 192.168.2.11 -u -p 8080 -b 100m
------------------------------------------------------------
Client connecting to 192.168.2.11, UDP port 8080
Sending 1470 byte datagrams, IPG target: 117.60 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 192.168.2.10 port 37265 connected with 192.168.2.11 port 8080
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   119 MBytes   100 Mbits/sec
[  3] Sent 85035 datagrams
[  3] Server Report:
[  3]  0.0-10.0 sec   119 MBytes   100 Mbits/sec   0.000 ms    0/85035 (0%)
```

Figure 12-4 - Iperf Result On VM1

On VM2 (iperf server side):

```
hungtm@hungtm-VirtualBox:~$ iperf -s -u -p 8080
------------------------------------------------------------
Server listening on UDP port 8080
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 192.168.2.11 port 8080 connected with 192.168.2.10 port 37265
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[  3]  0.0-10.0 sec   119 MBytes   100 Mbits/sec   0.000 ms    0/85035 (0%)
```

Figure 12-5 - Iperf Result on VM2

12.4.3.2. Testing with egress policing (srTCM rfc2697)

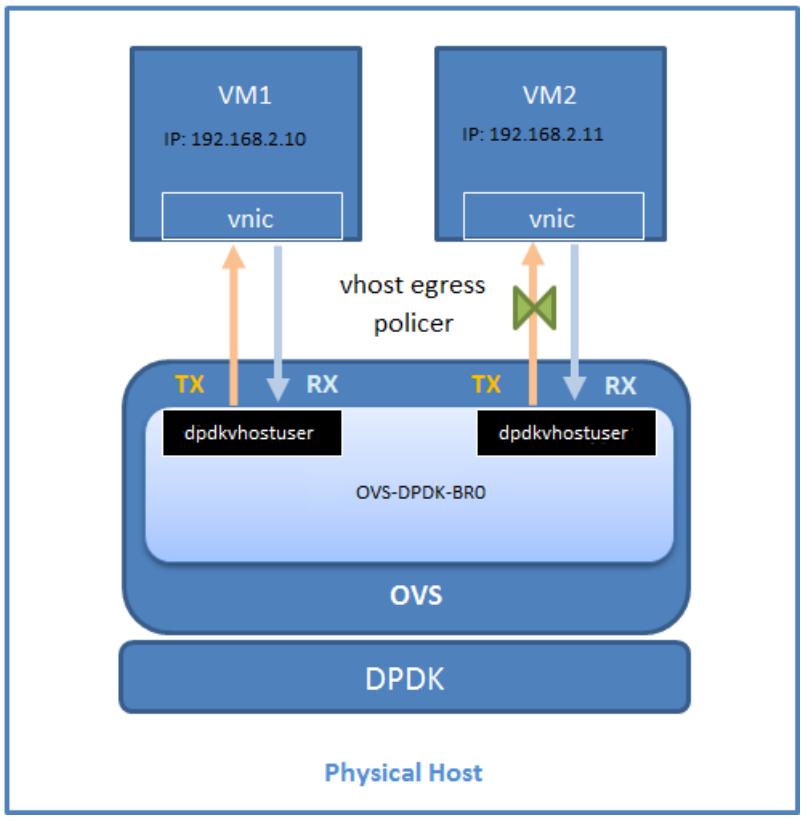


Figure 12-6 - Egress Policing Testing Molde

```
#Set limitation for `ovs-br0-vport2` port.
$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- --id=@newqos create qos type=egress-policer
other-config:cir=1250000 other-config:cbs=2048

#Check the qos configuration
$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2
```

```
QoS: ovs-br0-vport2 egress-policer
cir: 1250000
cbs: 2048
```

Start `iperf on VM1 & VM2 again to check the qos egress policing. the result in VM1 (client side) should like below:

```
hungtm@hungtm-VirtualBox:~$ iperf -c 192.168.2.11 -u -p 8080 -b 100m
------------------------------------------------------------
Client connecting to 192.168.2.11, UDP port 8080
Sending 1470 byte datagrams, IPG target: 117.60 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 192.168.2.10 port 58125 connected with 192.168.2.11 port 8080
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   119 MBytes   100 Mbits/sec
[  3] Sent 85035 datagrams
[  3] Server Report:
[  3]  0.0-10.3 sec  11.7 MBytes  9.57 Mbits/sec   0.000 ms 76690/85034 (0%)
```

Figure 12-7 - QoS Egress Policing Result

Fore more information about this one, please refer **link**

12.4.3.3. Testing with egress policing (trTCM rfc2698)

Almost is the same with srTCM rfc2697, just configure **ovs-br0-vport2** port with QoS type: egress-policer-rfc2698 as below:

```
# Configure QoS egress policing trTCM (rfc2698)
`$ sudo ovs-vsctl set port ovs-br0-vport2 qos=@newqos -- --id=@newqos create qos type=egress-policer-rfc2698 other-config:cir=46000000 other-config:cbs=2048 other-config:pir=46000000 other-config:pbs=2048  other-config:ydropt=false other-config:rdropt=true`

# Edit parameter of QoS
`$ sudo ovs-vsctl set qos __uuid__ other-config:...

Note:
+ PIR have to equal or greater CIR
+ Default actions of yellow & red color is dropt (ydropt=true & rdropt=true)

# Check the qos configuration
`$ sudo ovs-appctl -t ovs-vswitchd qos/show ovs-br0-vport2`
QoS: ovs-br0-vport2 egress-policer-rfc2698
cir: 46000000
pbs: 2048
cbs: 2048
pir: 46000000
```

Then start iperf on VM1 & VM2 to testing, change cir,cbs,pir & pbs for more result.

12.4.3.4. Testing with ingress policing (Rate limiting)

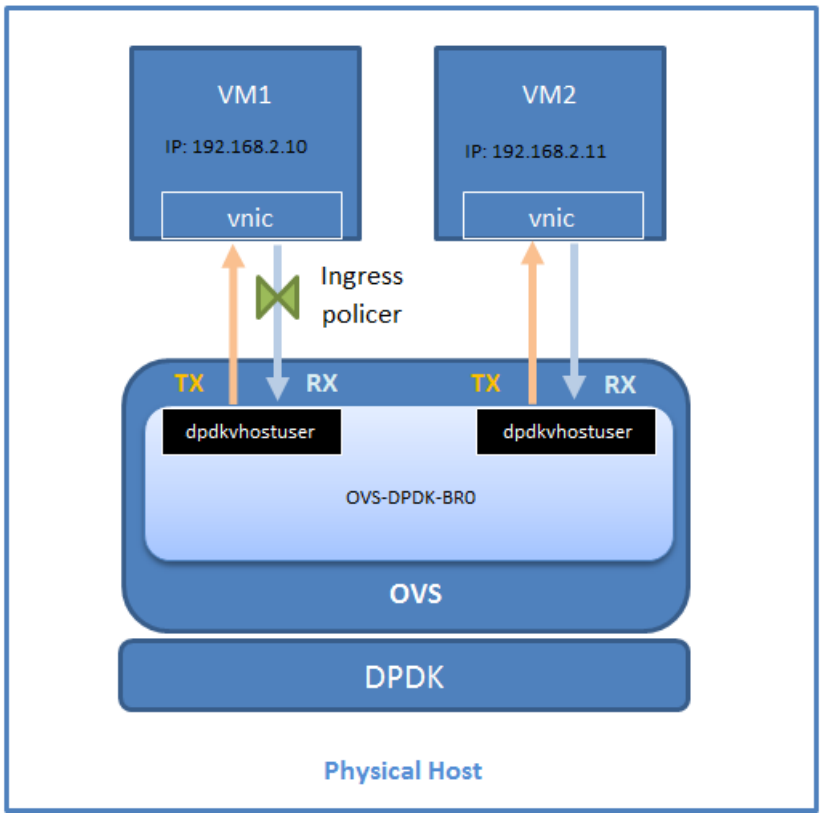


Figure 12-8 - Ingress Policing Testing Molde

```
#Set limitation for `ovs-br0-vport1` port.  (should remove `egress policing` to testing this one)
$ sudo ovs-vsctl set interface ovs-br0-vport1 ingress_policing_rate=1024 ingress_policing_burst=128

$ sudo ovs-vsctl list interface ovs-br0-vport1
_uuid               : c6a57b70-6df2-491b-830d-f15c7a0686a6
admin_state         : up
bfd                 : {}
bfd_status          : {}
cfm_fault           : []
cfm_fault_status    : []
cfm_flap_count      : []
cfm_health          : []
cfm_mpid            : []
cfm_remote_mpids    : []
cfm_remote_opstate  : []
duplex              : []
error               : []
external_ids        : {}
ifindex             : 3159192
ingress_policing_burst: 128
ingress_policing_rate: 1024
lacp_current        : []
link_resets         : 0
link_speed          : []
link_state          : up
lldp                : {}
mac                 : []
mac_in_use          : "00:00:00:00:00:00"
mtu                 : 1500
mtu_request         : []
name                : "ovs-br0-vport1"
ofport              : 2
ofport_request      : []
options             : {}
other_config        : {}
statistics          : {"rx_1024_to_1522_packets"=340146, "rx_128_to_255_packets"=46,
"rx_1523_to_max_packets"=0, "rx_1_to_64_packets"=27, "rx_256_to_511_packets"=27,
"rx_512_to_1023_packets"=0, "rx_65_to_127_packets"=33, rx_bytes=514323148, rx_dropped=0, rx_errors=0,
```

```
rx_packets=340279, tx_bytes=354648, tx_dropped=14326, tx_packets=1663}
status              : {features="0x000000017020e782", mode=server, num_of_vrings="2", numa="0",
socket="/home/hungtm/ovs_prj/ovs-software/var/run/openvswitch/ovs-br0-vport1", status=connected,
"vring_0_size"="256", "vring_1_size"="256"}
type                : dpdkvhostuser
```

Start `iperf on VM1 & VM2 again to check the qos ingress policing. the result in VM1 (client side) should like below:

```
hungtm@hungtm-VirtualBox:~$ iperf -c 192.168.2.11 -u -p 8080 -b 100m
------------------------------------------------------------
Client connecting to 192.168.2.11, UDP port 8080
Sending 1470 byte datagrams, IPG target: 117.60 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 192.168.2.10 port 49273 connected with 192.168.2.11 port 8080
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   119 MBytes   100 Mbits/sec
[  3] Sent 85035 datagrams
[  3] Server Report:
[  3]  0.0-10.3 sec  1.21 MBytes   992 Kbits/sec   0.000 ms 84169/85034 (0%)
```

Figure 12-9 - QoS igress Policing Result

For more information about this one, please refer **link** & fast refer configure command, refer **link**

12.4.4. Testing with more than 2VM

Base on **Model II**, add more VM then testing again on all VM. The step is the same with 2 VM.

12.4.5. Show data rate in OVS (TBU)