DPDK Tunneling Offload

RONY EFRAIM & YONGSEOK KOH

MELLANOX


DPDK
DATA PLANE DEVELOPMENT KIT
Rony Efraim

Introduction to DC w/ overlay network

- Modern data center (DC) use overly network like Virtual Extensible LAN (VXLAN) and GENEVE to create a virtual network on top of the physical network.

- The problem : all the HW acceleration used to accelerate TCP/IP stop working

The goal

- VMs on the DC eventually see a plain eth packet.
- Provide all necessary offloads for DPDK vSwitch to offload as much as it can to spare CPU and to scale
- Support all stateless HW acceleration for overly like:
  - RSS on inner 5 tuple
  - inner csum rx/tx (~15% gain)
  - Inner TSO (~30% gain in BW and 30% reduce in CPU cycles)
  - Switch rules offload

Yongseok Koh

Inner RSS

- To achieve uniform distribution for tunneled packets
  - Vary UDP source port if UDP is the transport (e.g., VxLAN)
  - Use inner header for RSS hash

- Inner RSS
  - Set level in struct rte_flow_action_rss
    - 0 for default behavior
    - 1 for outermost header
    - 2 and higher for inner header
  - testpmd> flow create 0 ingress
    pattern eth / ipv4 / udp / vxlan vni is 100 / eth / ipv4 / tcp / end
    actions **rss level 2** queues 0 1 2 3 end / end

Inner Checksum

- DEV_RX_OFFLOAD_XXX_CKSUM
  - XXX = [IPV4|UDP|TCP|SCTP|OUTER_IPV4|OUTER_UDP]
  - PMD marks mbuf->ol_flags with
    - PKT_RX_XXX_CKSUM_[UNKNOWN|BAD|GOOD|NONE]

- DEV_TX_OFFLOAD_XXX_CKSUM
  - XXX = [IPV4|UDP|TCP|OUTER_IPV4|OUTER_UDP]
  - App requests by marking mbuf->ol_flags with
    - PKT_TX_XXX_CKSUM
      - XXX = [IP|L4_NO|TCP|SCTP|UDP|OTUER_IP|OUTER_UDP]

Inner TSO

- App requests by
  - mbuf->ol_flags |= PKT_TX_TCP_SEG | PKT_TX_[IPV4|IPV6]
  - For IPv4 packet, set PKT_TX_IP_CKSUM

- DEV_TX_OFFLOAD_XXX_TNL_TSO
  - Some device supports only specific tunnels
    - XXX=[VXLAN|GRE|IPIP|GENEVE]

- DEV_TX_OFFLOAD_[IP|UDP]_TNL_TSO
  - Others (e.g. MLX5) also support generic tunnels (not listed above)
    - mbuf must have proper flags and offset values
      - PKT_TX_TUNNEL_[IP|UDP]
      - outer_l2_len, outer_l3_len, l2_len, l3_len and l4_len

Encap/Decap

- There were only two tunnel types defined until v18.08
  - RTE_FLOW_ACTION_TYPE_[VXLAN|NVGRE]_[ENCAP|DECAP]
- Take a list of rte_flow_items for encap header
- Needed to define more tunnel types
- “L3 encap/decap” is required, where there’s no inner L2.
  - VXLAN-GPE – [ETH/IP/UDP/VXLAN/IP/…]
  - MPLS over UDP – [ETH/IP/UDP/MPLS/IP/…]

Raw Encap/Decap

- Generic Encap/Decap action
  - RTE_FLOW_ACTION_TYPE_RAW_[ENCAP|DECAP]
  - Added in v18.11
- Encap header is specified with raw data
  - Data pointer in struct rte_flow_action_raw_encap
- Two actions need to be specified for ”L3 encap/decap”
  - [ETH / IPV6 / TCP] → Decap L2
    → [IPV6 / TCP] → Encap MPLSoUDP
    → [ETH / IPV4 / UDP / MPLS / IPV6 / TCP]

Modify Header

- RTE_FLOW_ACTION_TYPE_SET_XXX
  - XXX = [MAC_[SRC|DST]|IPV4_[SRC|DST]|IPV6_[SRC|DST]|TP_[SRC|DST]]
- RTE_FLOW_ACTION_[SET|DEC]_TTL
- Suitable for
  - NAT
  - Hairpin

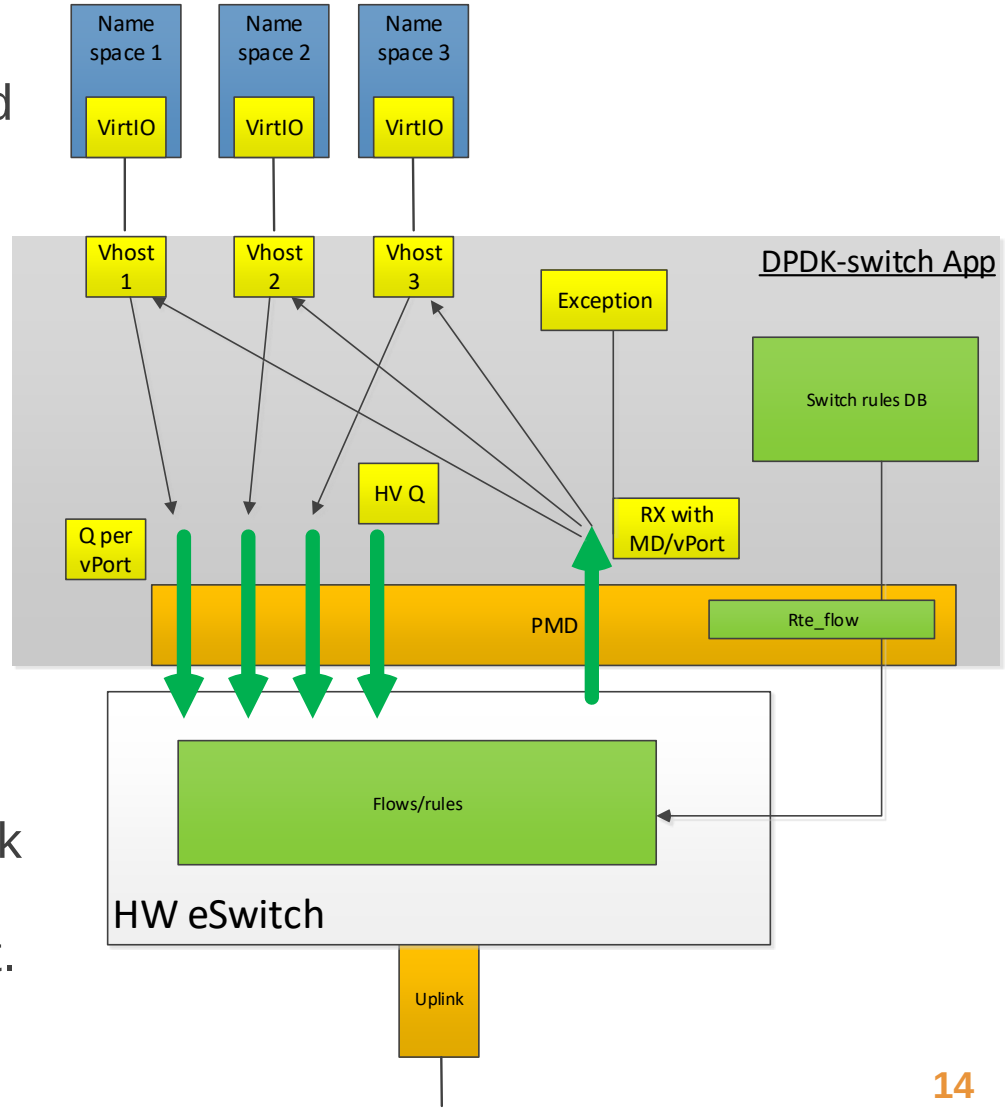
Metadata Match

- RTE_FLOW_ITEM_META
  - Added in 18.11
- Match on egress
  - mbuf->tx_metadata
  - mbuf->ol_flags |= PKT_TX_METADATA
- Useful to apply different encap/decap actions between VMs
  - VMs may have same network topology, i.e. same network headers
    - Need to distinguish different VMs
    - VM ID as metadata


DPDK
DATA PLANE DEVELOPMENT KIT
Rony Efraim

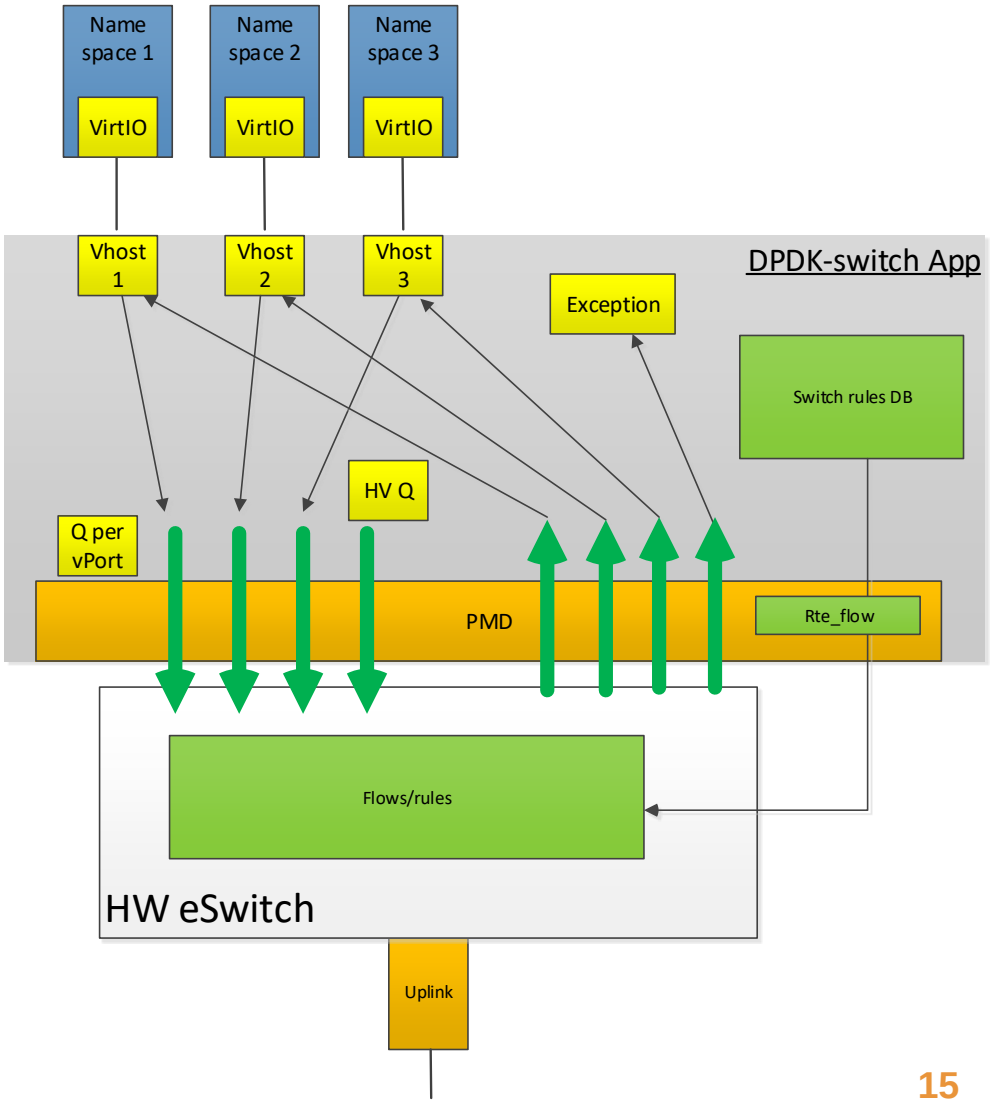
Example

- Flow rules
  - add rte_flow on Tx to match on vPort (send-Q/MD) and inner L4/L3/L2 with action to
    - Encap + send to wire
    - Loopback packet + set a vPort ID / Rx metadata
      - Exceptions: Rx metadata will be the vPort/VM id sent the packet
      - VM -> local VM Rx metadata will be the destination vPort/VM
  - add rte_flow on Rx to match on vxlan + inner L4/L3/L2 with action decap + set a vPort ID / Rx metadata.
- From VM (Tx)
  - pulls from vHost and sends the buffer on the vPort send-Q or on HV-Q with MD
  - No need to classify or fetch the data buffer.
- To VM (Rx)
  - Single receive-Q for packets from all sources (loopback + wire)
  - polls NIC (PMD) and get a metadata with every packet.
  - The metadata is used either to forward the packet to the VM or further process by the switch application.



Zero copy on RX

- The HW can steer the traffic according to the vPort ID/ Rx Metadata to receive-Q.
- By using a receive-Q per VM the software switch application can offload :
  - The need to read the Rx metadata and to pull a batch of packets and forward to a specific vHost.
  - Using the buffer form the vRing or buffers of the VM there is no need to copy the buffers.
- Mellanox NIC support up to millions of receive-Q.




DPDK
DATA PLANE DEVELOPMENT KIT
QnA


DPDK
DATA PLANE DEVELOPMENT KIT
Thank You!