Status, Open Issues and Extensions for switchdev SR-IOV mode

- Or Gerlitz, Mellanox
- Simon Horman, Netronome

- Netdev 2.2, Nov 2017, Seoul, Korea

BoF Outline:

- summary of the current upstream support
- SRIOV e-switch use cases with L2 FDB / L3 FIB offloads
- SRIOV e-switch Linux Bridge offload
- SRIOV e-switch Linux Router offload
- Representor netdev naming conventions
- port model for CPU and physical (uplink) ports
- Applicability for switching container networking

summary of the current upstream support

- 4.8 - basic support for the proposed architecture was introduced
  - SRIOV switchdev mode (devlink)
  - VF rep netdevs and TC/flower e-switch management (mlx5)
- 4.9..4.12 - VLAN, tunnel and header-rewrite actions were face-lifted/defined/offloaded (so far mlx5 only)
  - vlan action offload
  - tunnel key action definition and offload
  - pedit action face list and offload
- 4.13.. -- nfp driver support was added for most/all the above
- 4.14 -- bnxt driver support was added for some of the above
- 4.15.. -- more drivers coming up?! (liquidio, i40e)

SRIOV e-switch use cases with L2 FDB / L3 FIB offloads

- **If the control plane is flow based e.g (OVS, ODL, DVR)** we need the HW driver to support a simple set of flow matches and actions, to be programmed directly e.g through tc/flower:

  FDB: match on mac/vlans, push/pop vlan + fwd

  FIB: match on mac/vlans, apply header re-writes (MACs, TTL) + fwd

- For L3, we assumed here a simple use-case ☺

- For both cases, flow counters HW support is typically needed for packet/bytes statistics and last-used based aging

SRIOV use cases with L2 / L3 offloads – cont’

- Common NIC HWs don't have the same pipeline building blocks as switch ASIC

  Still, we aim to conduct offloading of L2 (FDB) and L3 (FIB) Linux data-paths

- Proposal: adjust the NIC HW driver to register and act on the kernel L2/L3 notifications as done by switch ASIC driver

- Adjust some aspects of the core networking to deal with these type of drivers

- When the HW API is flow based, some translation can be done in the HW driver from the kernel object (FIB/FDB) to flow

SRIOV e-switch Linux Bridge offload

- create linux bridge (e.g .1q), assign VF and uplink rep netdevices to the bridge
- support the switchdev FDB notifications in the HW driver

- learning:
  - respond to *SWITCHDEV_FDB_ADD_TO_DEVICE* events
  - if the HW API doesn't support FDBs, translate to flow: match on mac/vlan --> fwd to port

- aging:
  - respond *to SWITCHDEV_FDB_DEL_TO_DEVICE* events (del FDB from HW)
  - enhance the driver/bridge API to allows drivers provide last-use indications on FDB entries

- STP:
  - fwd - offload FDBs as explained above
  - learning - make sure HW flow miss (slow path) goes to CPU
  - discard - add drop HW rule
- flooding:
  - use SW based flooding

SRIOV e-switch Linux Router offload

- create linux bridge, assign VF reps netdevices to the bridge, assign IP address to the bridge
- assign IP address to the uplink rep, to be used as router port (the uplink rep is not part of the bridge)
- support FIB (*FIB_EVENT_ADD/DEL*) notifications in the HW driver

- under a simple use-case, router HW does two lookups prior to xmit:
  - LPM: match on dest IP, resolve the interface and next-hop
  - Neigh: based on the dest ip + interface, resolve the next-hop mac

- flow based NIC HW APIs can be used to emulate that
  - LPM: do decreasing match on prefixes of offloaded routes: /32../31.../0
  - Neigh: based on the LPM matching, resolve dest mac using offloaded table of kernel neighs

- doing this whole data-path in HW can use MD (Meta-Data) to remember previously matched elements along the pipe-line
- this doesn't cover ECMP, VRF, etc.

Representor netdev naming conventions

- Proposal
  - “[RFC] switchdev: clarify ndo_get_phys_port_name formats”
  - pA for physical ports
    - where A is port name or ID
  - pAsB for split physical ports
    - where B is sub-port name or ID
  - pfC for PF representors
    - where C is PCIe PF name or ID
  - pfCvfD for VF representors
    - where D is PCIe VF name or ID

Port Model for CPU and Physical (Uplink) Ports

- Model 1 (mlx5)
  - PF netdevs = represents phys ports
- Model 2 (nfp)
  - PF netdev = link between host and NIC
  - MAC representor netdevs = e-switch side of phys ports

Applicability for switching container networking

- easy option, reuse: assign containers with VFs netdevs mapped to their name-space, connect the VF representors netdevs to SW switching, etc
- scalability can be problematic, ##VFs support by NICs is limited, amount of resources opened by the driver/FW per VF might be way too much for simple container app

- consider VMDQ to be a set of NIC HW queues opened over the PF netdev assigned to a container, e.g based on the upstream HW accelerated MACVLAN (John F. and Co. 2012).
- idea, generalize the VF rep model for VMDQ reps such that slow path works just the same:
  - Xmit on the VMDQ rep --> Recv into VMDQ nic
  - Xmit on the VMDQ nic --> Recv into VMDQ rep
  - support tc/flower offloads on the VMDQ rep net-device for fast-path offload