OvS-DPDK Tunneling and Connection Tracing Hardware Offload via Rte_flow

XIAO WANG, ROSEN(WEIHUA) XU

Agenda

DPDK SUMMIT APAC•2021

- OvS-DPDK Partial Offload
  - Virtual switch partial offload status quo
  - Flow flattening cache introduction
  - HW acceleration based on FFC
- OvS-DPDK Full offload
  - HW platform
  - Tunnel full offload
  - Connect Track and NAT full offload

OvS-DPDK Partial Offload

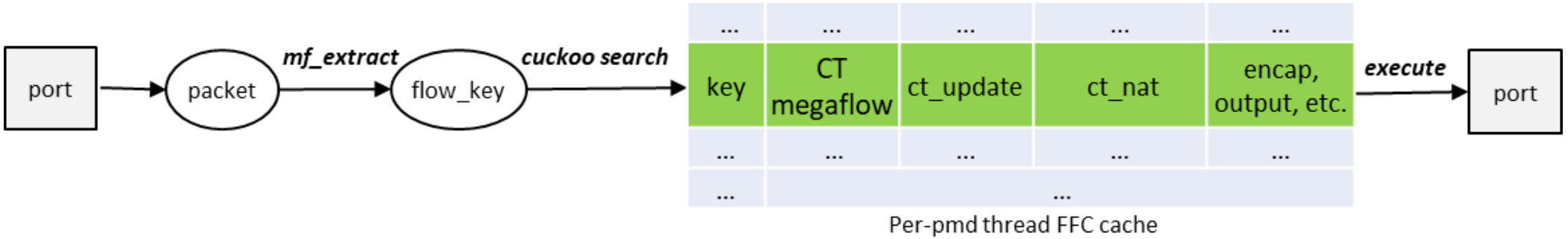
vSwitch Partial Offload Status Quo

DPDK SUMMIT APAC•2021

- SW-centric solution comes with operational flexibility
  - Live migration
  - Hot-upgrade
  - Little NIC dependency
- Partial offload allows flow lookup HW acceleration
  - Based on rte_flow API, in a best-effort way
  - Only basic feature w/o tunnel and CT support
  - Recirculation mechanism limits the effectiveness of HW offload

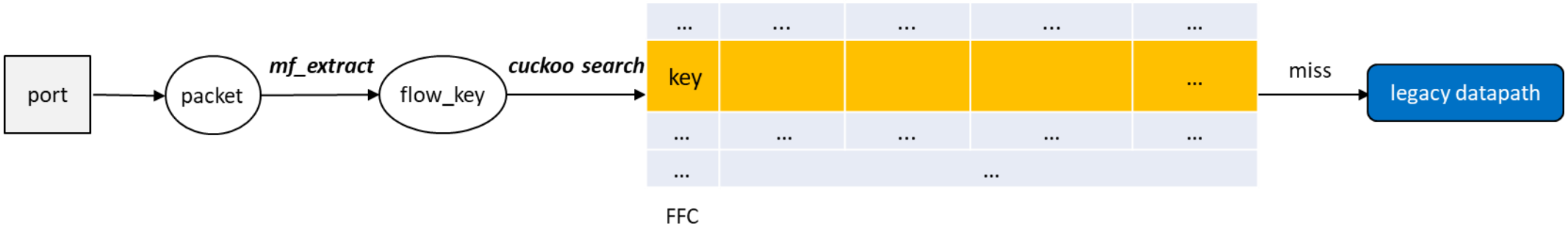
Introduce Flow Flattening Cache

- Purpose
  - Reducing the overhead of multiple packet header parsing and flow entry searching.
- Method
  - Exact match based FFC cache, caching the search results of previous packet with the same flow_key.
  - Each FFC entry includes the associated megaflow(s), CT connection.
  - Parse/Lookup once, execute on multi flows/conn.
  - Only established sessions can generate a FFC entry.



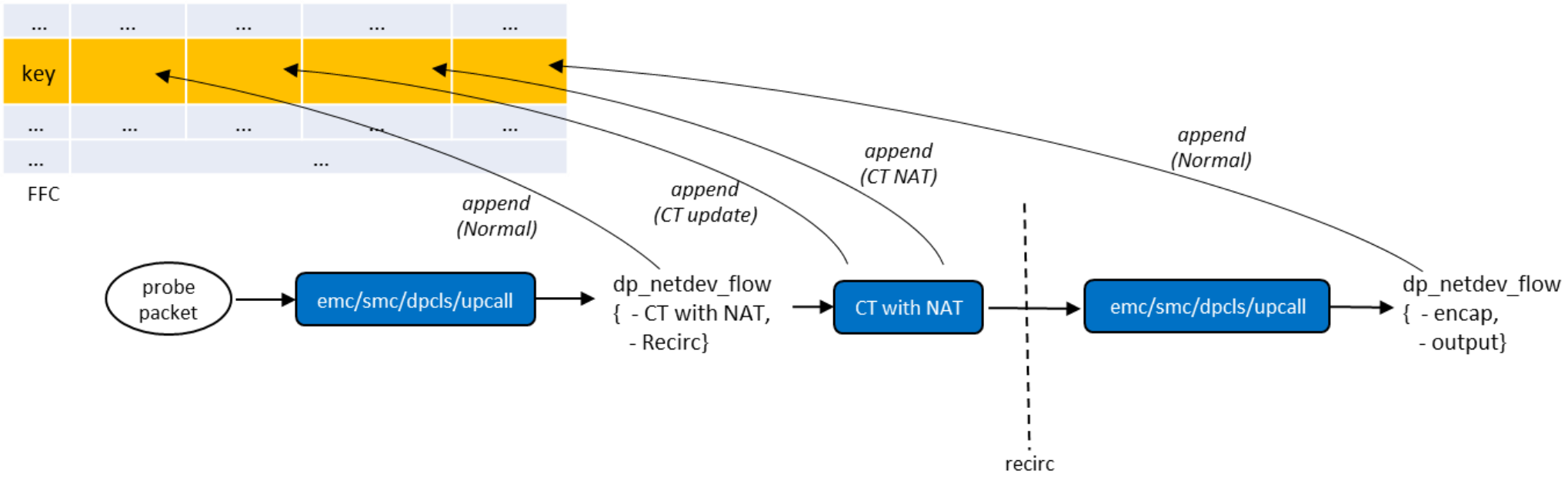
Flow Flattening Cache Generation (step 1)

- Reserve an FFC entry for cache generation.
- This packet takes a probe role and goes into the legacy OVS pipeline.



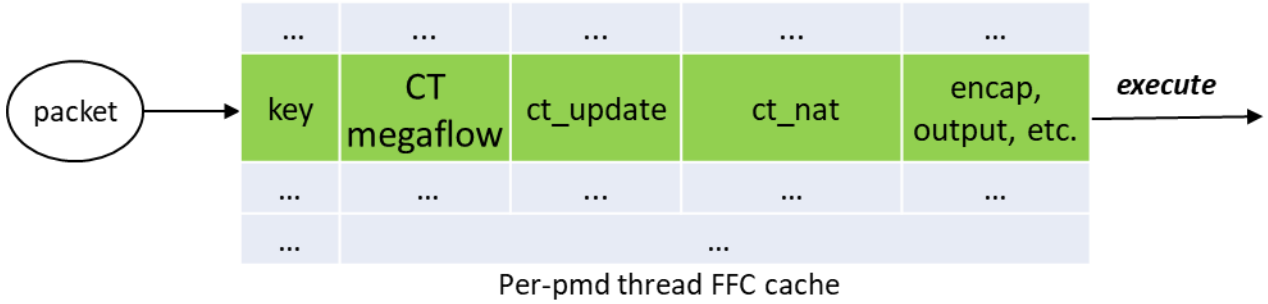
Flow Flattening Cache Generation (step 2)

- The probe packet travels through a multi-phased packet processing.
- FFC entry is generated along with the CT handling and recirculation.



Flow Flattening Cache Execution

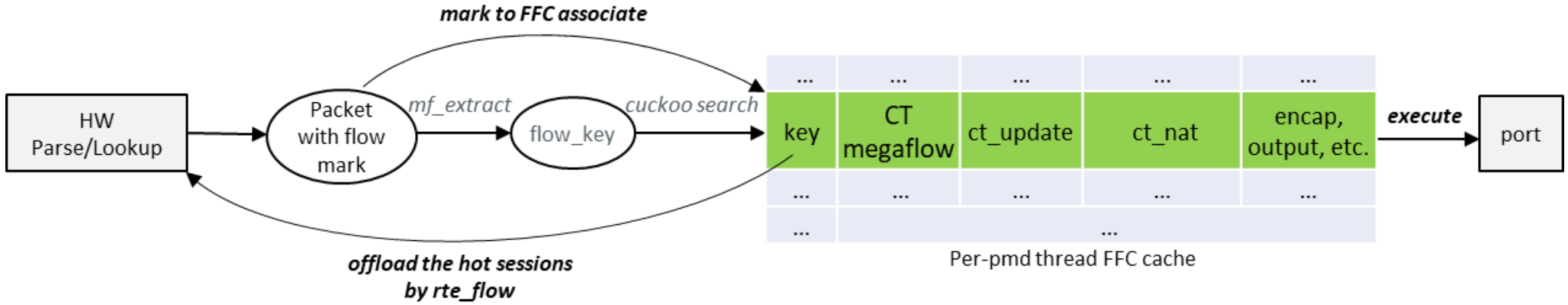
- A cache entry is active only if all the associated flows, conn are active.
- Cut down ~50% overhead for flow parsing/lookup.




```
For each ffc_flow in this ffc entry
{
        enter_ffc_mode(pmd);
        switch{ffc_flow->type} {
        case FFC_TYPE_NORMAL:
                /* leverage existing flow execution code routine,
                   no recirc when ffc_mode detected */
                break;
        case FFC_TYPE_CT:
                /* CT conn update, add new CT API */
                break;
        case FFC_TYPE_CT_NAT:
                /* CT nat packet, add new CT API */
                break;
        }
        exit_ffc_mode(pmd);
}
```

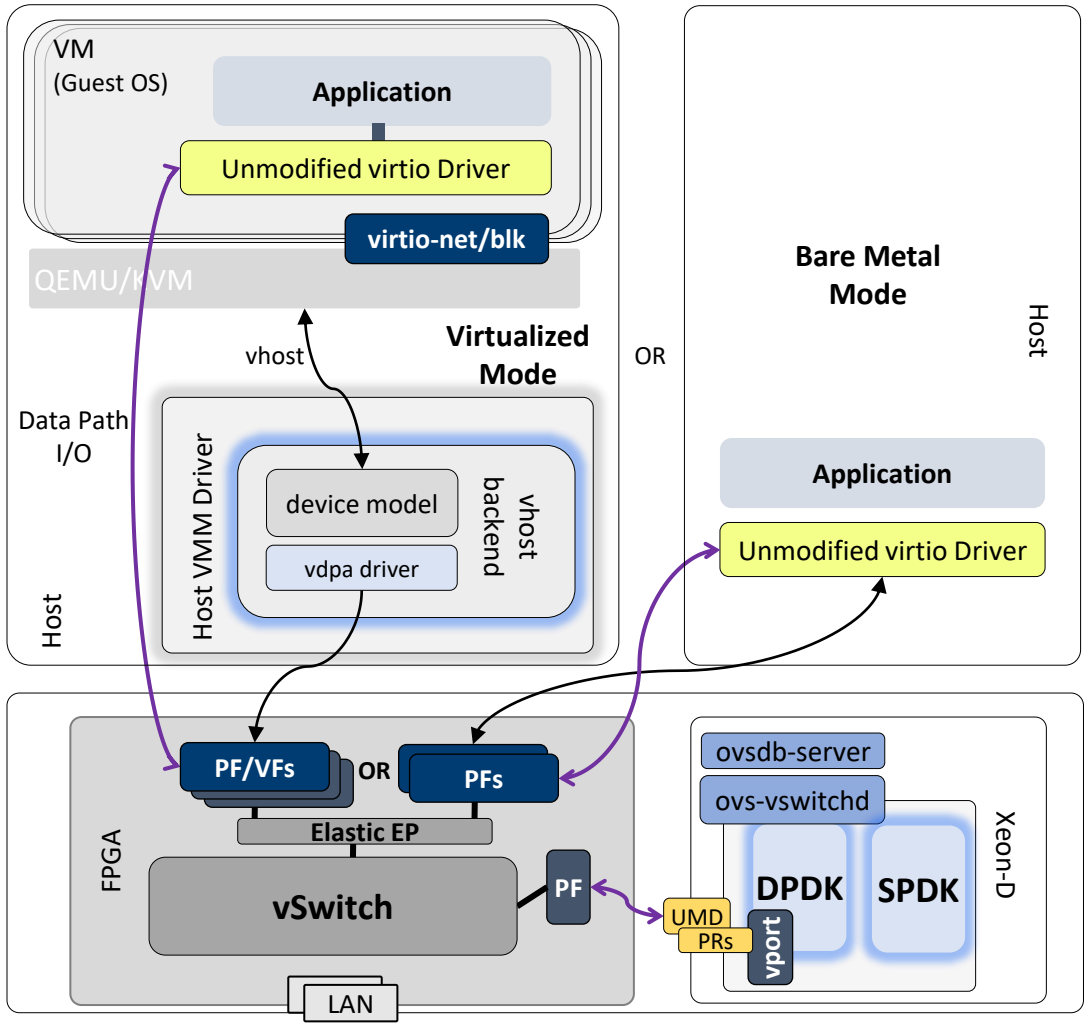
HW parser assists with FFC lookup

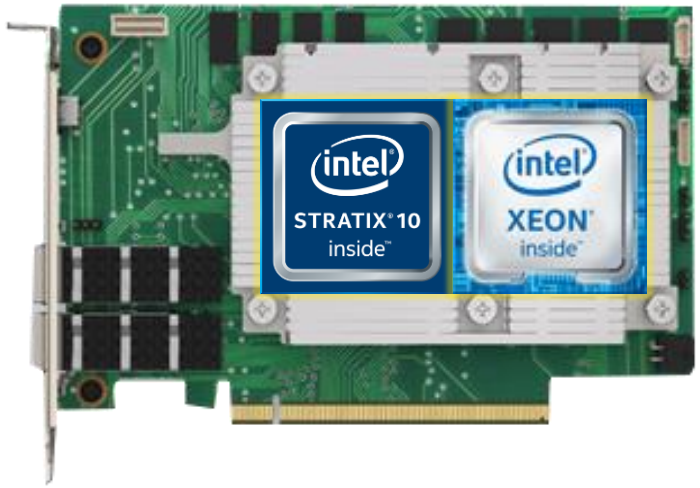
- HW do the parse/lookup, SW directly execute on vtep/ct/forward actions.
- Require flexible and performant HW parser and flow classifier.
- Make most use of HW capacity by offloading the hot sessions.
- Example with E810 VXLAN Acceleration: https://www.sdnlab.com/24697.html



OvS-DPDK Full Offload

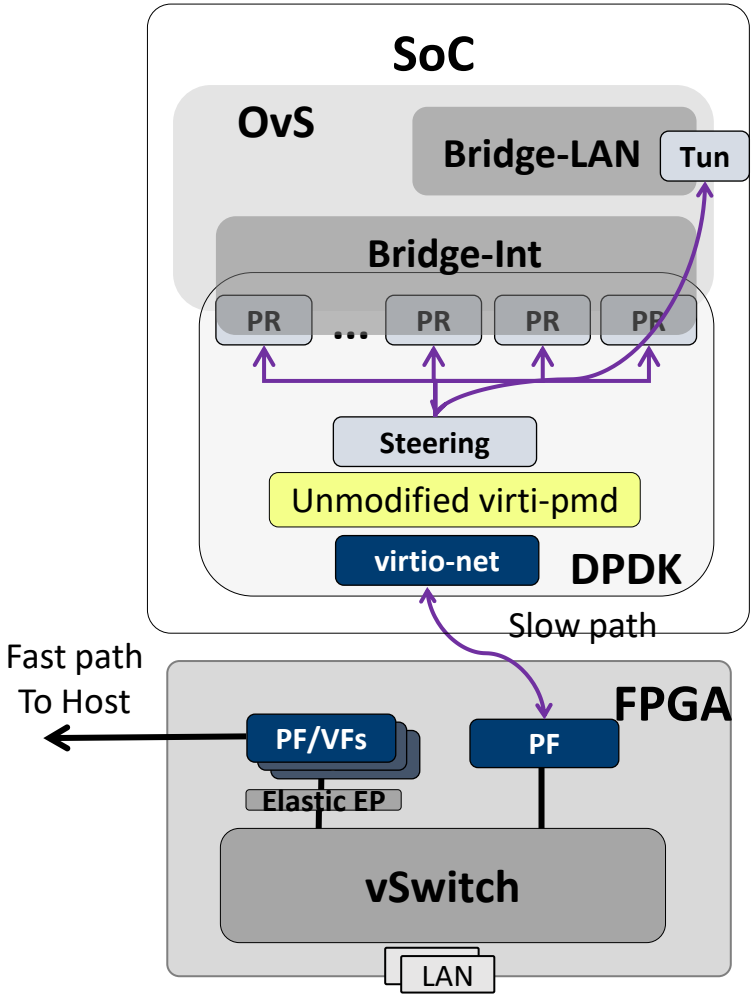
Full offload HW Platform - Intel Big Spring Canyon





- Virtualization and Bare Metal scenarios
- Virtualization Private Cloud(VPC) is powered by DPDK
- Elastic Block Storage(EBS) is powered by SPDK
- https://www.intel.com/content/www/us/en/products/network-io/smartnic.html

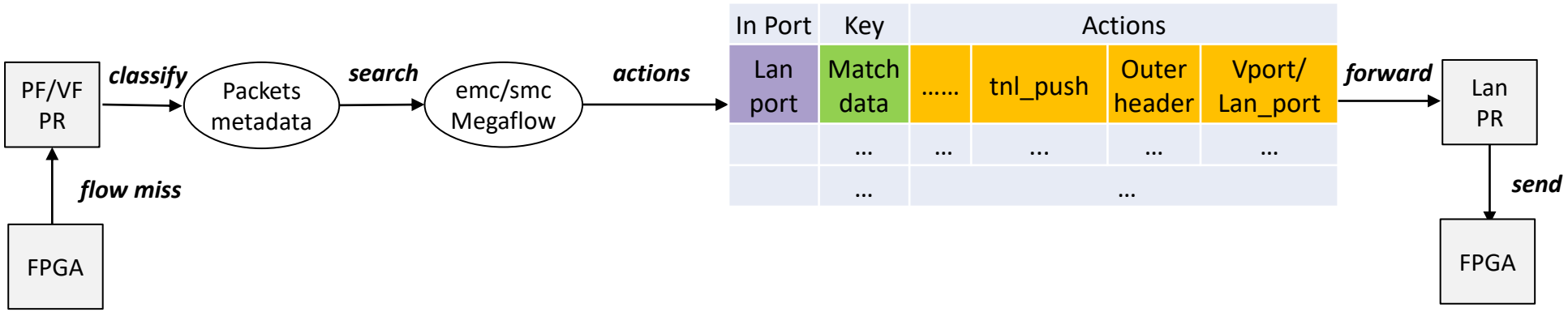
Tunnel – VxLAN and Geneve Introduction



- There are 2 bridges in OVS instance
  - Br-int: VMs connection and VxLAN/Geneve tunnel access
  - Br-LAN: VxLAN/Geneve peer connection
- Bridges are connected by vport
  - VxLAN/Geneve pop/push
  - Packet recircle in SW pipeline
- Difference between OvS SW pipeline and HW acceleration
  - Encap and Decap are basic operations supported by HW
  - It's not friendly for HW to aware of vport

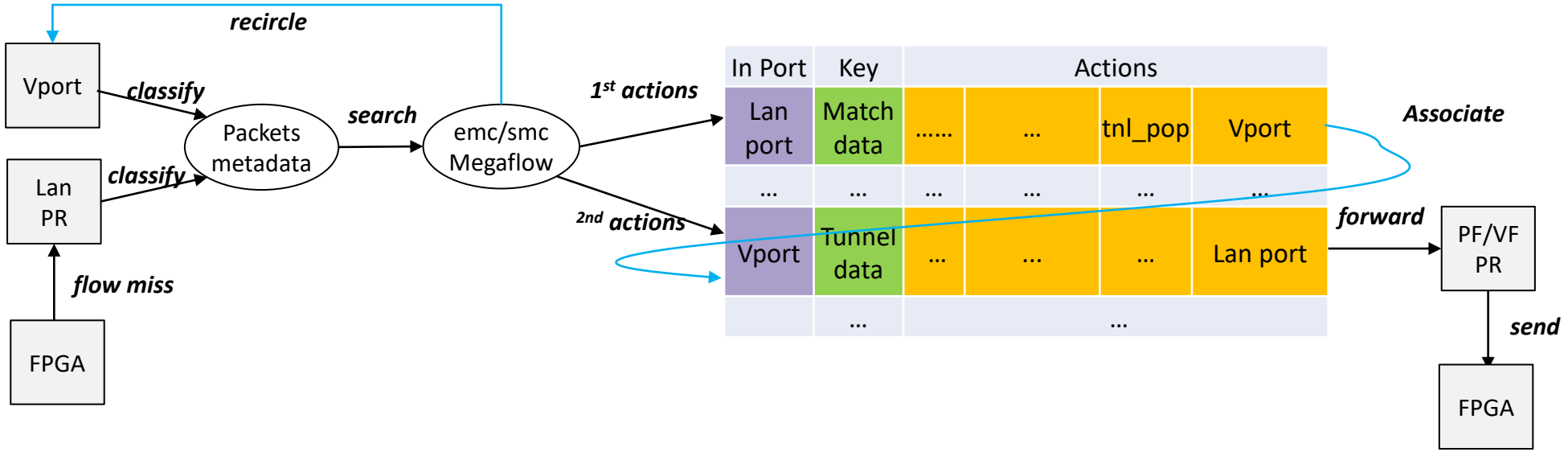
Tunnel Full Offload – Encap

- VxLAN/Geneve Encap – tnl_push
  - Executed by one netdev rte_flow
  - Entire packet header and Encap date in one flow
  - Easily to reuse existing DPDK rte_flow offload process

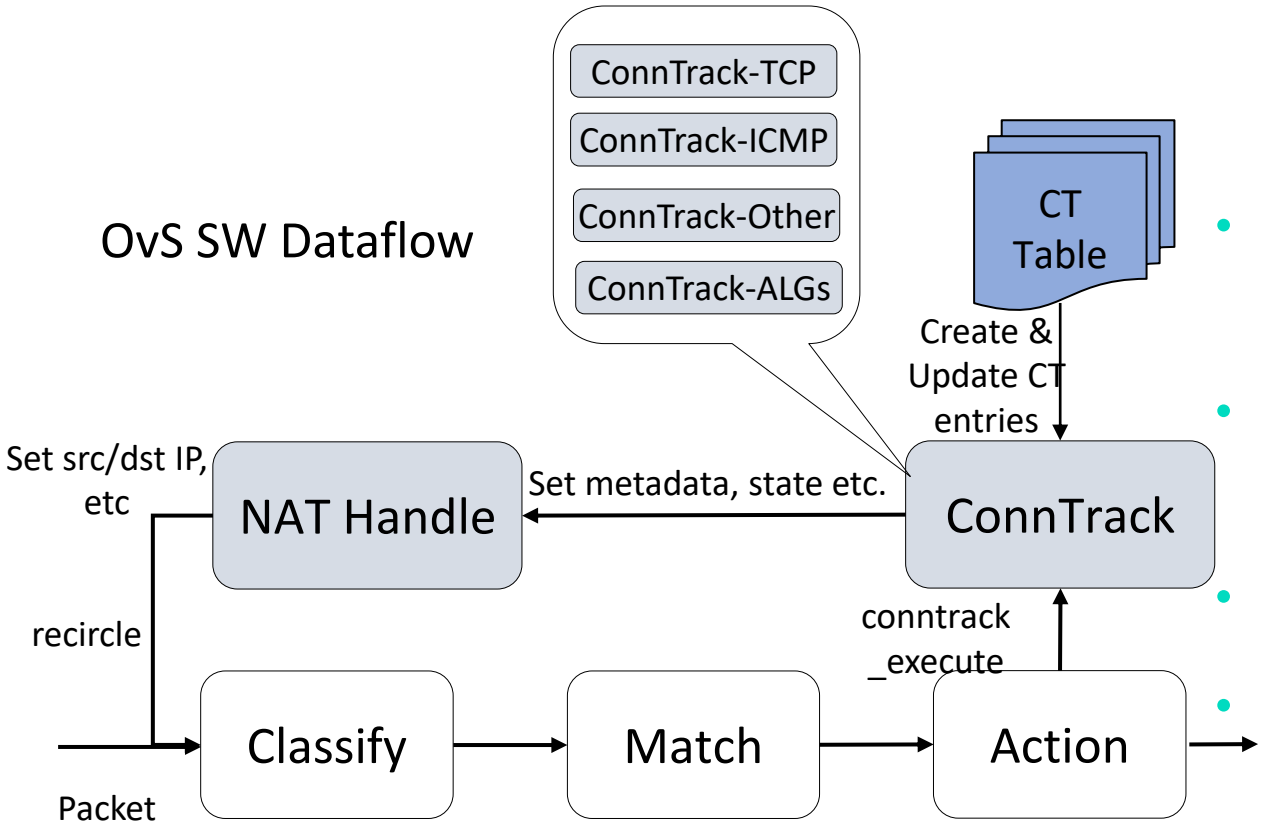


Tunnel Full Offload – Decap

- VxLAN/Geneve Decap – tnl_pop
  - Executed by two netdev rte_flow
  - Decap tunnel and forward to vport in 1st netdev rte_flow
  - Packets recircled in OvS SW pipeline and execute 2nd netdev rte_flow
  - Focusing on association for two netdev rte_flow



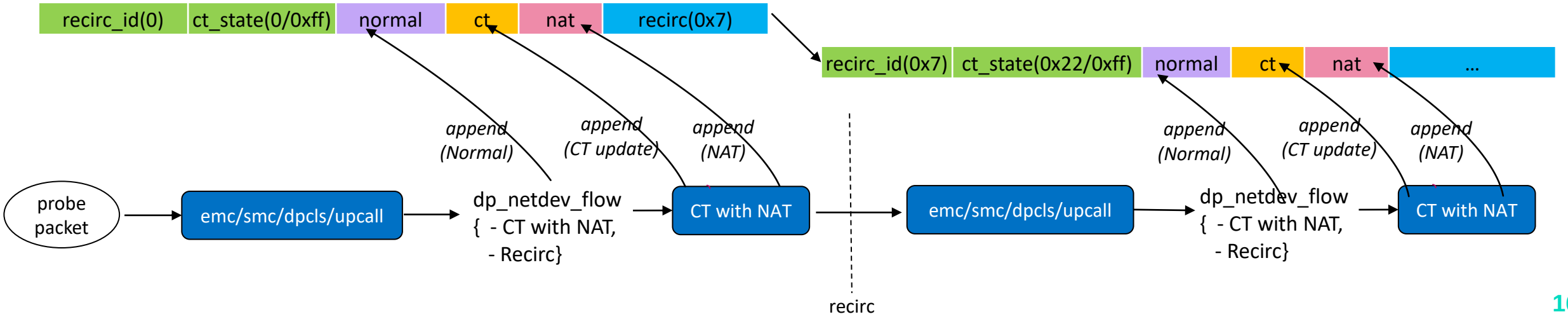
Connection Tracking Introduction



- Tracking both stateless and stateful protocols(TCP/ICMP/ALG etc)
- CT state for every packets(New/Established/Related/Invalid)
- Recircle with new CT state
- Supporting NAT features

Connection Tracking – DPDK Flow Chain

- DPDK has to be enhanced to support flow recirc
  - OvS recirc_id is represented by rte_flow_attr->group
  - Implement RTE_FLOW_ACTION_TYPE_JUMP action
  - For HW supports recircle, it's easy to mapping rte_flow chain to HW pipeline
  - For HW doesn't support recircle, rte_flow chain merging is necessary
- ConnTrack state offload
  - For HW implemented ConnTrack FSM – State of every packet remains consistent between SW and HW
  - For HW doesn't implemented ConnTrack FSM – Making the decision to offload in packets in 'est' state



DPDK

SUMMIT

APAC • 2021