NFV Use-case Enablement on DPDK and FD.IO

Cristian Dumitrescu, SW Architect, Intel

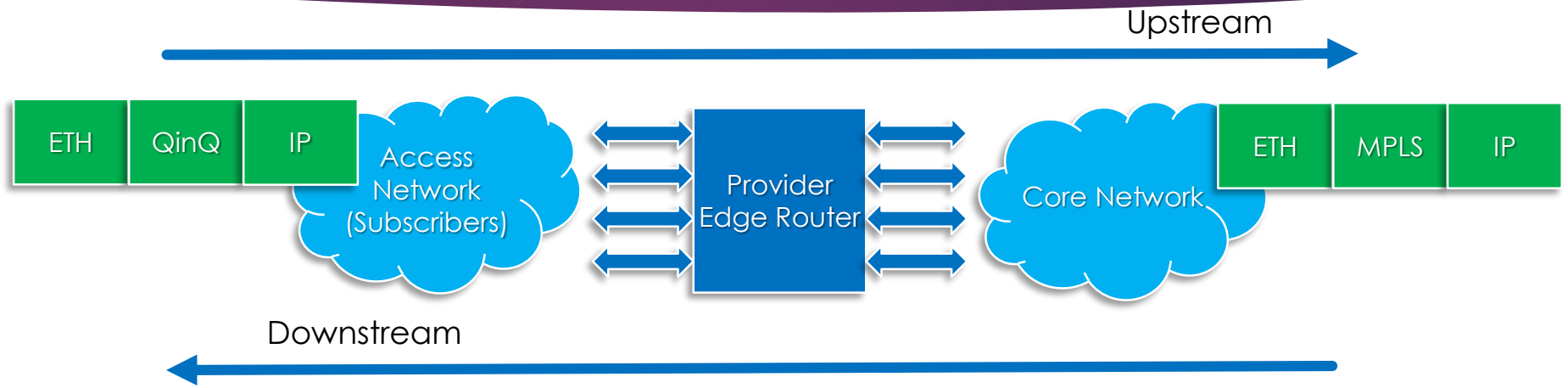
Jasvinder Singh, SW Engineer, Intel

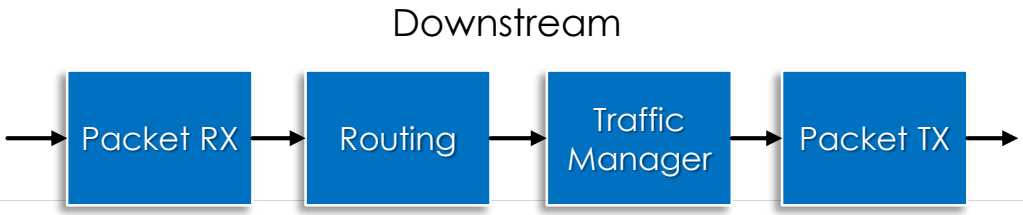
DPDK Summit Userspace - Dublin- 2016

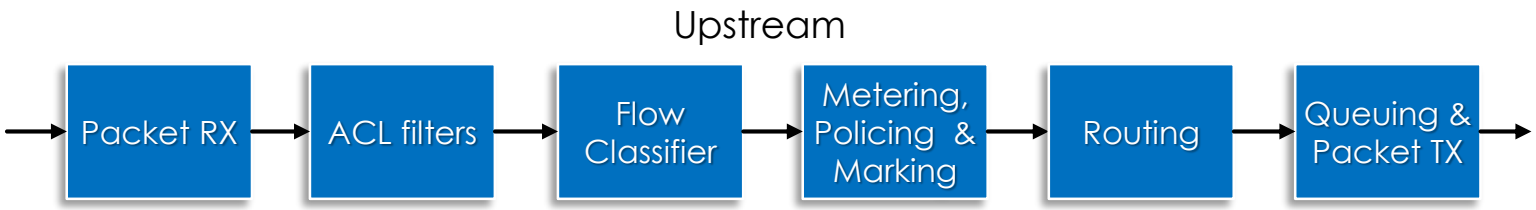
NFV Use-case Enablement on DPDK and FD.IO

- Edge Router Use-case Overview
- Implementation using DPDK
- Implementation using FD.IO VPP

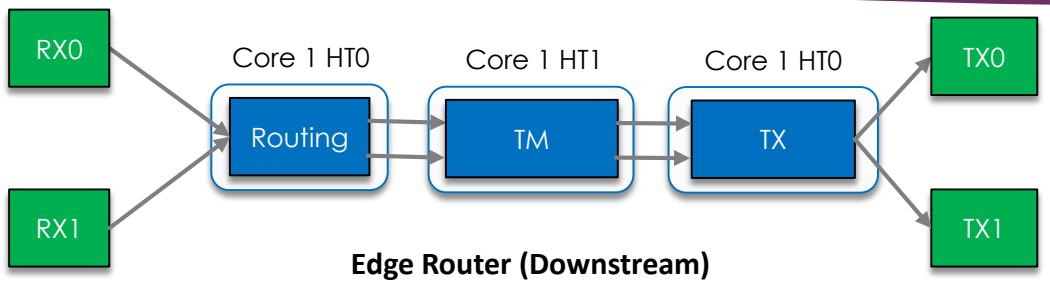
Edge Router Use-case Overview







Implementation using DPDK (1)



Edge Router (Downstream)

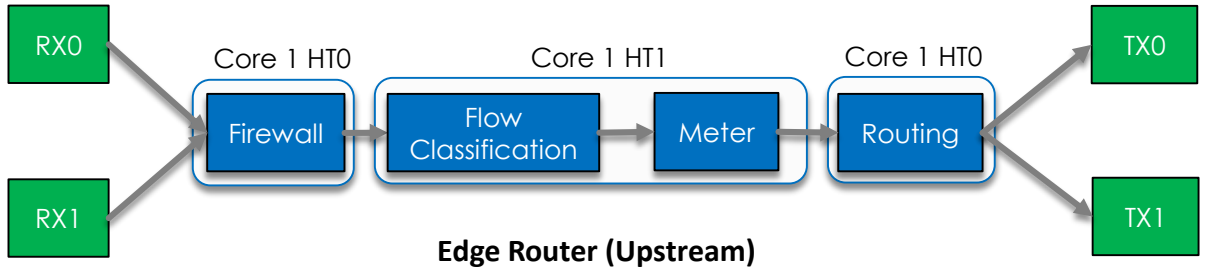
#File“<DPDK>/examples/ip_pipeline/edge_router_downstream.cfg” :

[PIPELINE1]
type = ROUTING
core = 1
pktq_in = RXQ0.0 RXQ1.0
pktq_out = SWQ0 SWQ1 SINK0
encap = ethernet_qinq
ip_hdr_offset = 270

[PIPELINE2]
type = PASS-THROUGH
core = 1h
pktq_in = SWQ0 SWQ1 TM0 TM1
pktq_out = TM0 TM1 SWQ2 SWQ3

[PIPELINE3]
type = PASS-THROUGH
core = 1
pktq_in = SWQ2 SWQ3
pktq_out = TXQ0.0 TXQ1.0

[MEMPOOL0]
pool_size = 2M



Edge Router (Upstream)

File “<DPDK>/examples/ip_pipeline/edge_router_upstream.cfg”:

[PIPELINE1]
type = FIREWALL
core = 1
pktq_in = RXQ0.0 RXQ1.0
pktq_out = SWQ0 SINK0
n_rules = 4096
pkt_type = qinq_ipv4

[PIPELINE3]
type = FLOW_CLASSIFICATION
core = 1h
pktq_in = SWQ0
pktq_out = SWQ1 SINK1
n_flows = 65536
key_size = 8
key_offset = 268
hash_offset = 128
Key mask = 00000FFF00000FFF

[PIPELINE4]
type = FLOW_ACTIONS
core = 1h
pktq_in = SWQ1
pktq_out = SWQ2
n_flows = 65536
n_meters_per_flow = 1
flow_id_offset = 132
ip_hdr_offset = 278
color_offset = 136

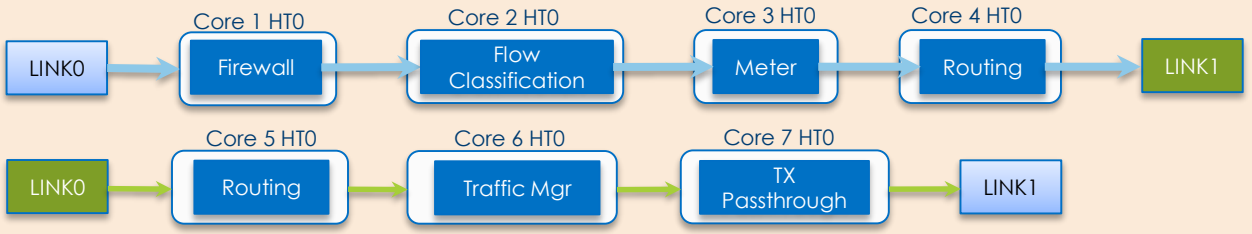
[PIPELINE5]
type = ROUTING
core = 1
pktq_in = SWQ2
pktq_out = TXQ0.0 TXQ1.0 SINK2
encap = ethernet_mpls
mpls_color_mark = yes
ip_hdr_offset = 278
color_offset = 136

Implementation using DPDK (2)

DPDK

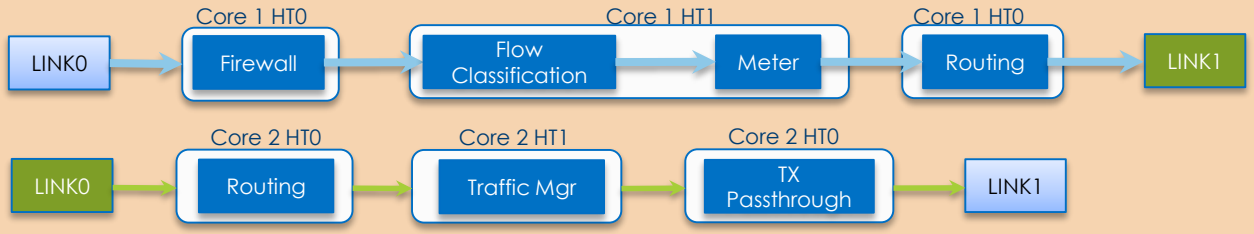
Step 1: Functional development

- Focus on functional correctness and block-level performance optimizations
- Each pipeline on separate CPU core



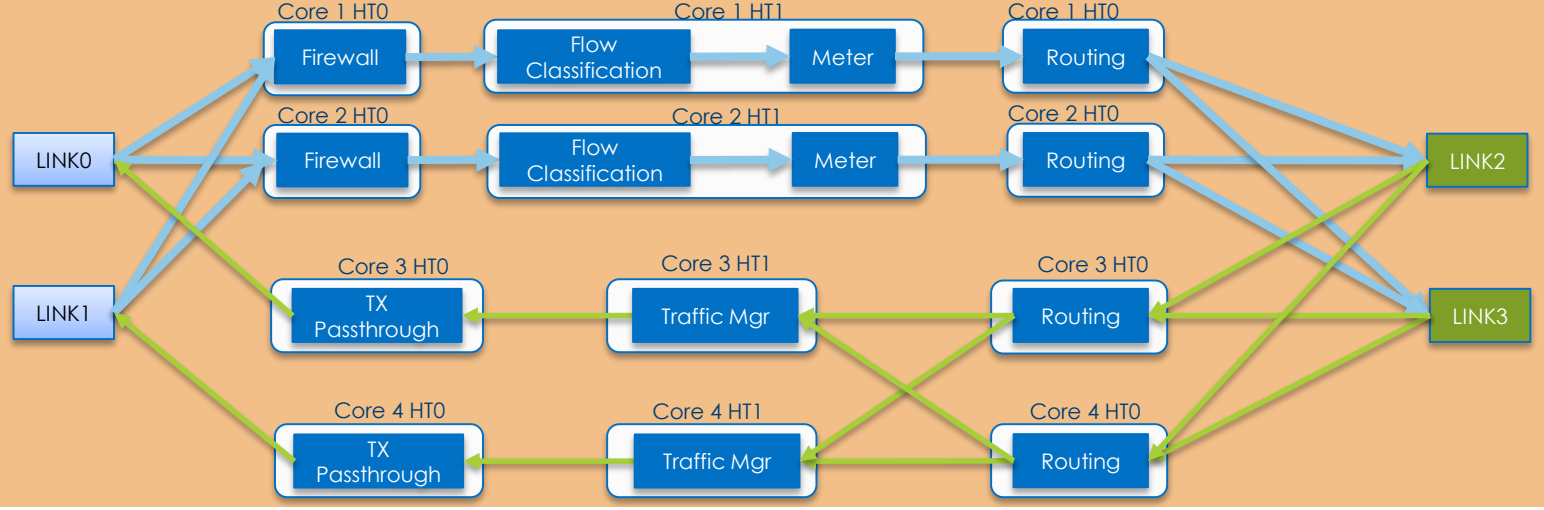
Step 2: Optimal mapping to CPU cores

- Identify optimal mapping of pipelines to CPU cores for single app instance (golden mapping)
- Maximize performance per physical CPU core: max(P/N), where: P=app throughput, N=#physical CPU cores



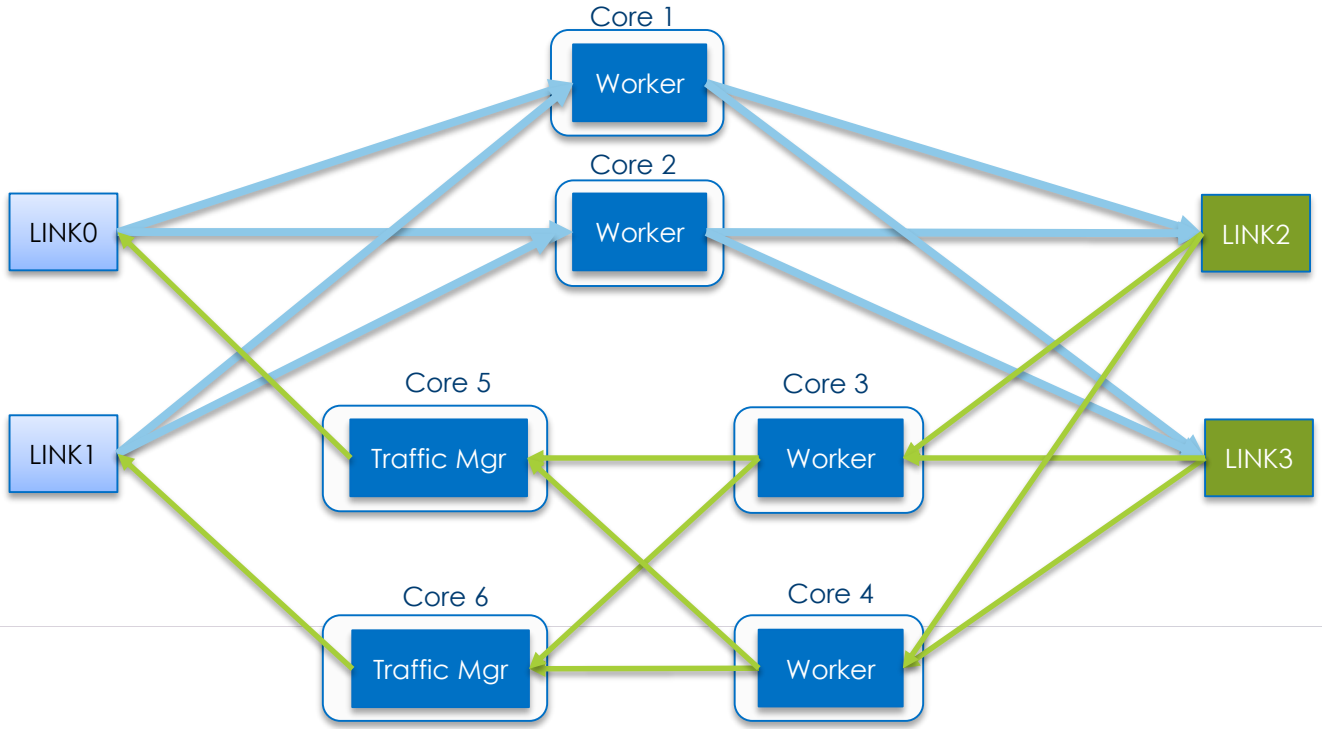
Step 3: Performance scale up

- Replicate the golden mapping several times
- Run multiple app instances on different CPU cores



Implementation using FD.IO VPP (1)

- VPP Framework
  - Process a vector of packets (frame) using directed graph of nodes
  - Large library of network functions
- Egress Traffic Management in VPP
  - Leverage DPDK QoS library (librte_sched)
  - 5-level hierarchical scheduler to make best use of available bandwidth



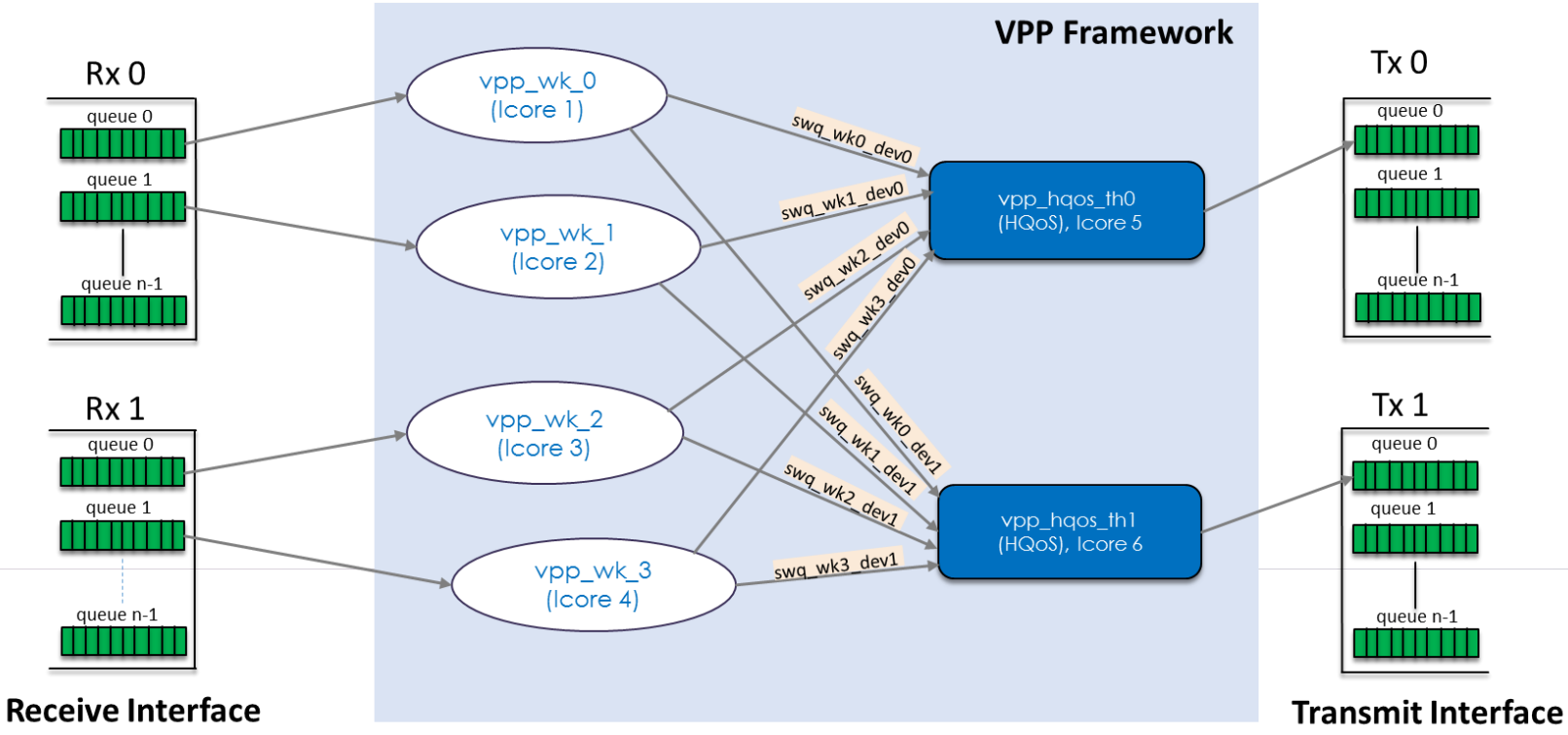
Implementation using FD.IO VPP(2)

```
#File "/etc/vpp/startup.conf":

unix {
interactive
log /tmp/vpp.log
cli-listen localhost:5002
full-coredump
}

dpdk {
socket-mem 16384,16384
dev 0000:02:00.0 {num-rx-queues 2 hqos}
dev 0000:04:00.0 {num-rx-queues 2 hqos}
num-mbufs 1000000
}

cpu {
main-core 0
corelist-workers  1,2,3,4
corelist-hqos-threads  5,6
}
```



Questions?

Cristian Dumitrescu

cristian.dumitrescu@intel.com

Jasvinder Singh

jasvinder.singh@intel.com

Legal Disclaimers

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. **No computer system can be absolutely secure**. Check with your system manufacturer or retailer or learn more at intel.com.