DPDK Summit

Vamsi Attunuru

DPDK on an Intelligent NIC

Vamsi Attunuru

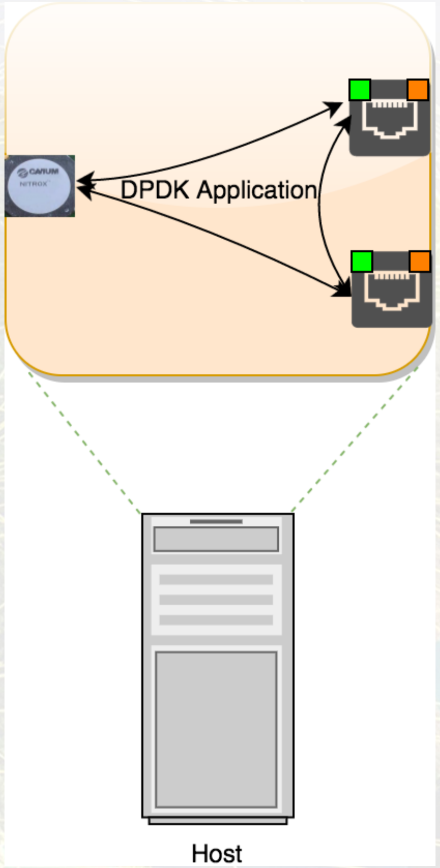
Background

- What is PCI-e EndPoint?
  - Target mode in PCI-express.
  - Common devices - NIC, Graphic Cards, Security Coprocessors.
  - The PCI-e channel can be used for control or data plane.

- What is DPDK?
  - A set of libraries and drivers for fast packet processing.
  - Enables third-party fast path stacks in Linux userspace.

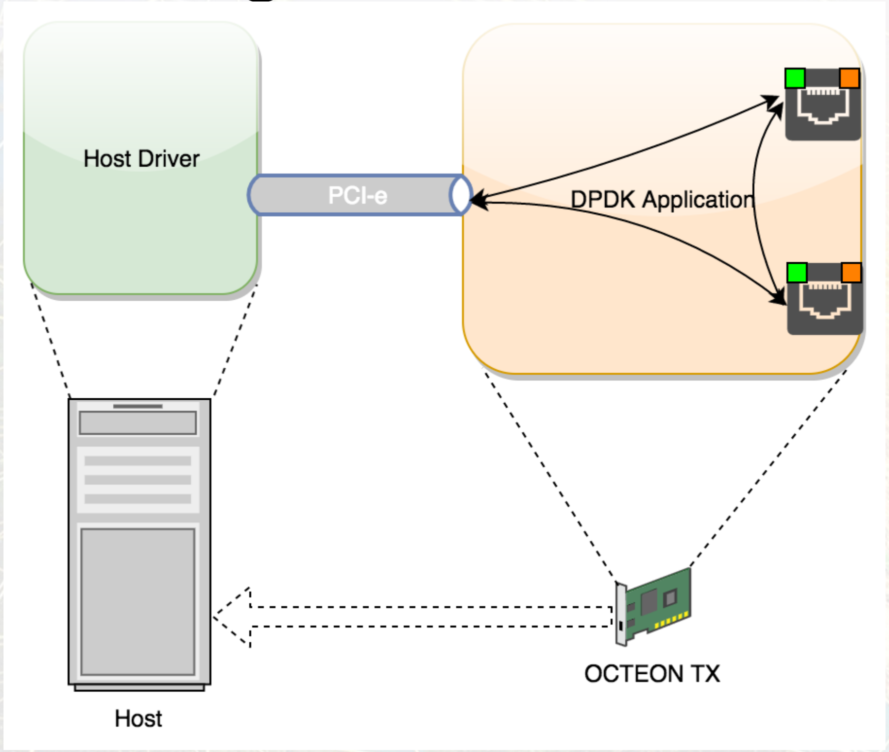
Conventional DPDK Usage

- Runs on standalone data plane processor
- DPDK application is Bus Master and owns the hardware



DPDK on an Intelligent NIC

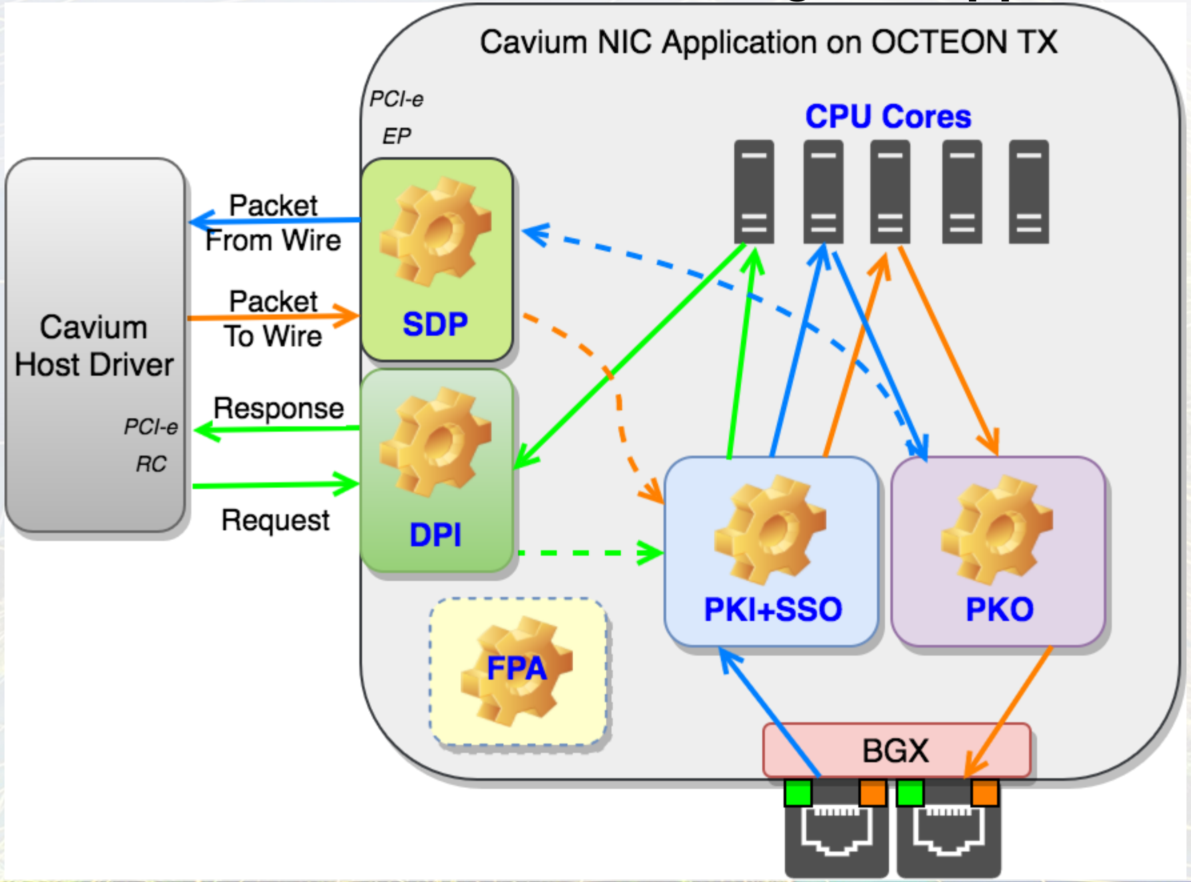
- DPDK runs on a co-processor
- DPDK application is a PCI-e slave.



Content on the PCI-e bus

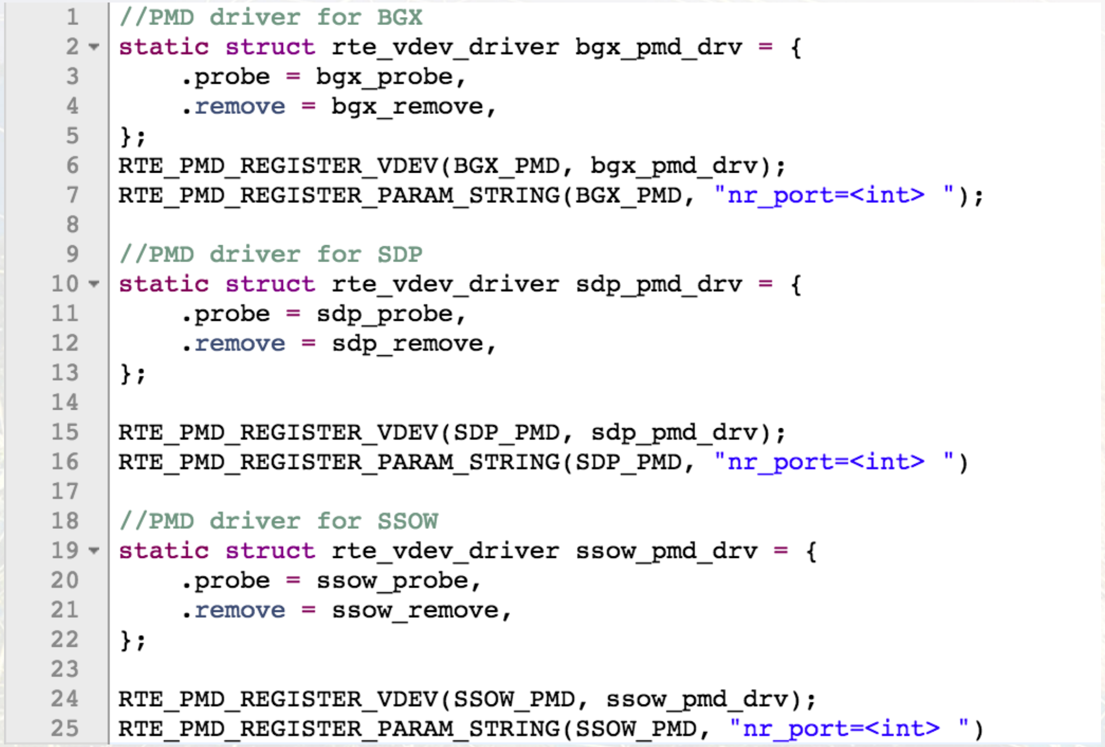
- Networks Packet with L2,L3,etc headers
- Custom message - typically as a request/response

OCTEON TX as a PCI-e EP running NIC application



PCI-e EP as NIC

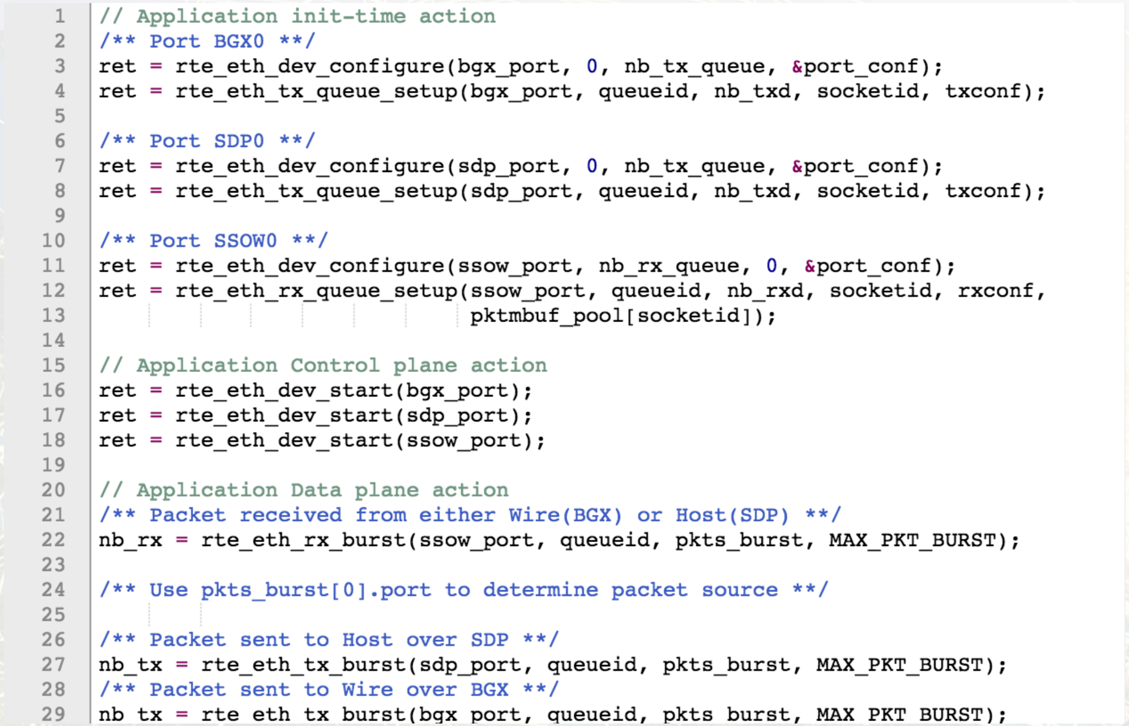
- Host projected as a NIC device
- Fits with the DPDK PMD model
- Register PCI-e EP as a RTE_ETH_DEV



```
//PMD driver for BGX
static struct rte_vdev_driver bgx_pmd_drv = {
        .probe = bgx_probe,
        .remove = bgx_remove,
};
RTE_PMD_REGISTER_VDEV(BGX_PMD, bgx_pmd_drv);
RTE_PMD_REGISTER_PARAM_STRING(BGX_PMD, "nr_port=<int> ");

//PMD driver for SDP
static struct rte_vdev_driver sdp_pmd_drv = {
        .probe = sdp_probe,
        .remove = sdp_remove,
};

RTE_PMD_REGISTER_VDEV(SDP_PMD, sdp_pmd_drv);
RTE_PMD_REGISTER_PARAM_STRING(SDP_PMD, "nr_port=<int> ")

//PMD driver for SSOW
static struct rte_vdev_driver ssow_pmd_drv = {
        .probe = ssow_probe,
        .remove = ssow_remove,
};

RTE_PMD_REGISTER_VDEV(SSOW_PMD, ssow_pmd_drv);
RTE_PMD_REGISTER_PARAM_STRING(SSOW_PMD, "nr_port=<int> ")
```

PCI-e EP as NIC

- Use established DPDK API for:
  - Convenient registration of hardware blocks - BGX, SDP, SSOW
  - Seamless send/recv of packets



```
// Application init-time action
/** Port BGX0 **/
ret = rte_eth_dev_configure(bgx_port, 0, nb_tx_queue, &port_conf);
ret = rte_eth_tx_queue_setup(bgx_port, queueid, nb_txd, socketid, txconf);

/** Port SDP0 **/
ret = rte_eth_dev_configure(sdp_port, 0, nb_tx_queue, &port_conf);
ret = rte_eth_tx_queue_setup(sdp_port, queueid, nb_txd, socketid, txconf);

/** Port SSOW0 **/
ret = rte_eth_dev_configure(ssow_port, nb_rx_queue, 0, &port_conf);
ret = rte_eth_rx_queue_setup(ssow_port, queueid, nb_rxd, socketid, rxconf,
                             pktmbuf_pool[socketid]);

// Application Control plane action
ret = rte_eth_dev_start(bgx_port);
ret = rte_eth_dev_start(sdp_port);
ret = rte_eth_dev_start(ssow_port);

// Application Data plane action
/** Packet received from either Wire(BGX) or Host(SDP) **/
nb_rx = rte_eth_rx_burst(ssow_port, queueid, pkts_burst, MAX_PKT_BURST);

/** Use pkts_burst[0].port to determine packet source **/

/** Packet sent to Host over SDP **/
nb_tx = rte_eth_tx_burst(sdp_port, queueid, pkts_burst, MAX_PKT_BURST);
/** Packet sent to Wire over BGX **/
nb_tx = rte_eth_tx_burst(bgx_port, queueid, pkts_burst, MAX_PKT_BURST);
```

PCI-e EP as a Coprocessor

- For Custom Messages, host is projected as a coprocessor to DPDK.
- Coprocessor can be registered as a ethdev PMD, but it is not intuitive and not a clean fit.
- Coprocessor can better leverage the eventdev PMD.
- DPDK eventdev PMD is a Work In Progress.

Status

- Packet and Message exchanges
  - Work in progress to adopt NIC firmware application to use DPDK’s PMD model to use PCI-e as a Network Packet Interface and as a Coprocessor.

- Roadmap
  - Run performance benchmarks with NIC firmware running over DPDK and compare against non-DPDK NIC firmware implementation.
  - Use eventdev PMD instead of ethdev for custom messages and even for Network packet exchange.

Q & A

Vamsi Attunuru
Vamsi.Attunuru@Cavium.com

THANK YOU