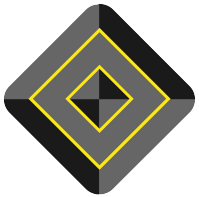
OpenDataPlane as the base technology for SDNs

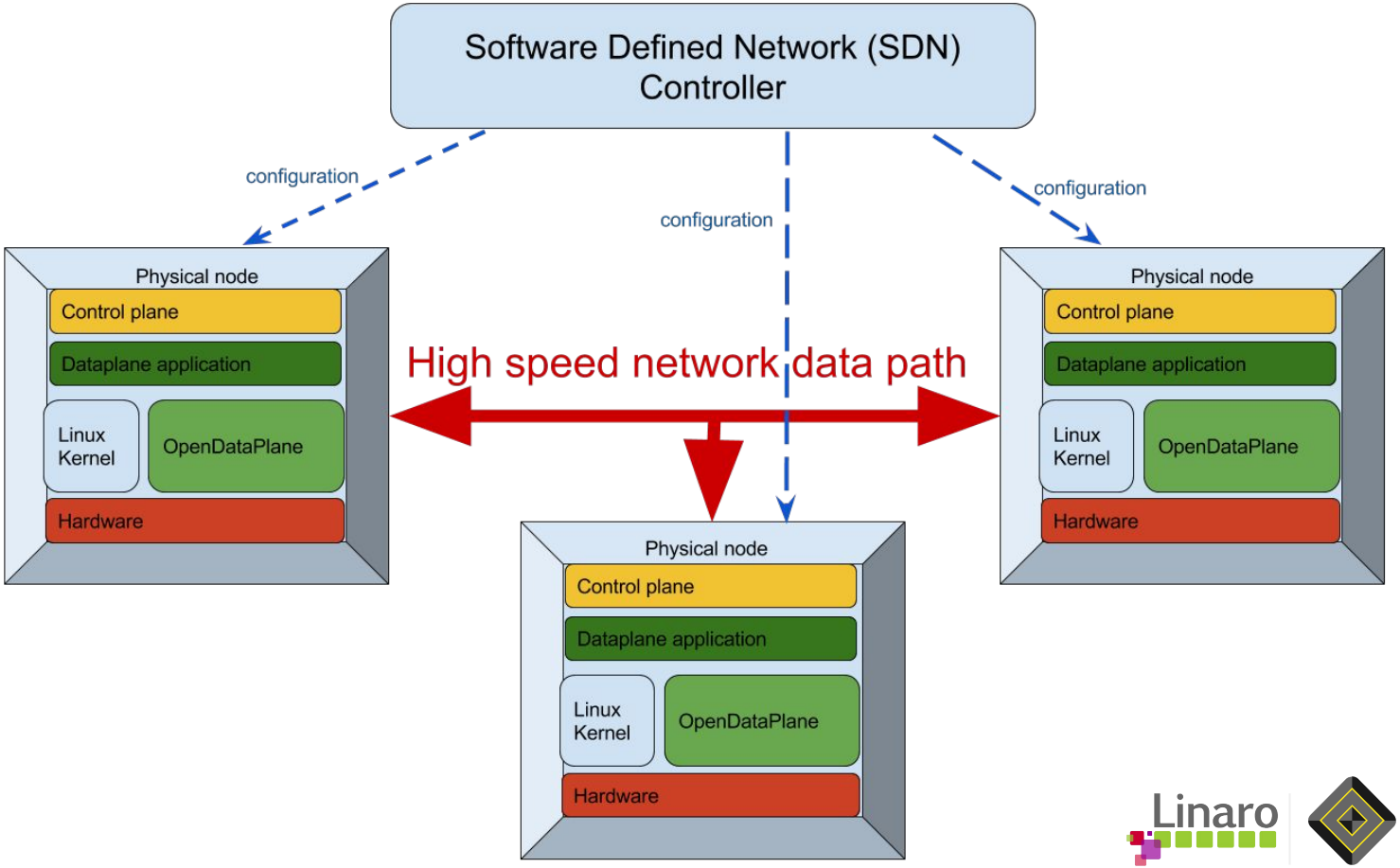
Linux Peter conference,
Nov3 and Nov4 2017, Saint Petersburg, Russia.

Maxim Uvarov
Senior Software Engineer,
Linaro Networking Group
maxim.uvarov@linaro.org

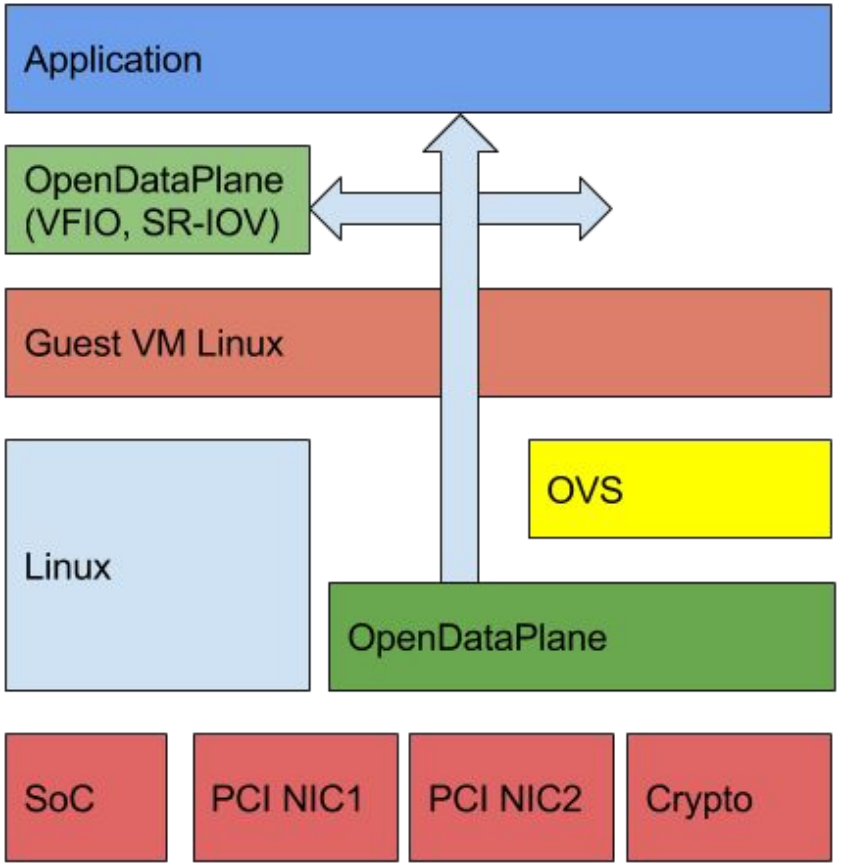


OpenDataPlane
.org

SDN picture and dataplane place



SDN - NFV node with ODP



Dataplane

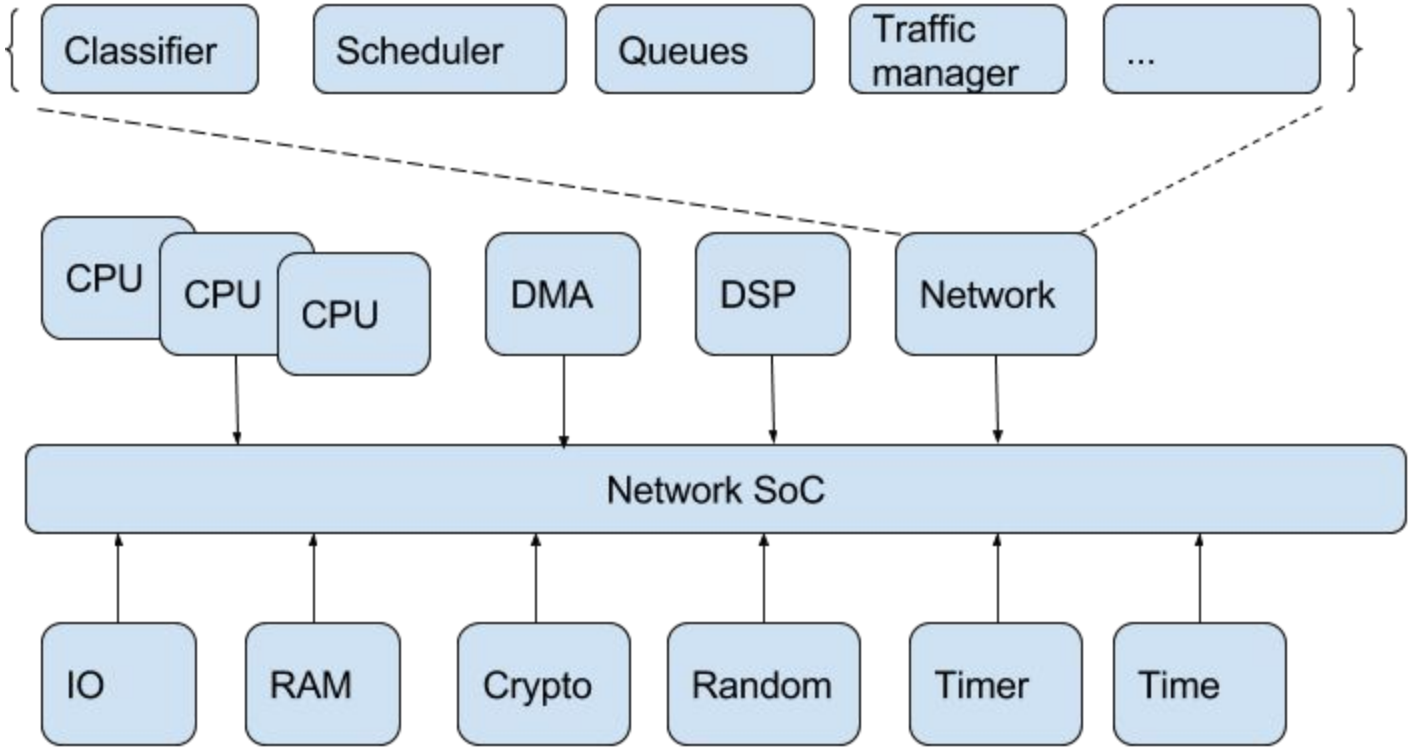
Data plane refers to all the functions and processes that forward packets/frames from one interface to another.

Control plane refers to all the functions and processes that determine which path to use. Routing protocols, spanning tree, ldp, etc are examples.

Management plane combines functions you use to control and monitor devices.

These are mostly logical concepts but things like SDN separate them into actual devices.

Abstract network hardware, SoC



OpenDataPlane intro

The ODP project has been established to produce an open-source, cross-platform set of application programming interfaces (APIs) for the networking data plane. Similar to OpenGL for graphics, but for networking stack.

OpenDataPlane
.org

- ODP is the set of APIs describing hardware in software.
- It is represented as library both static and dynamic.
- Supported ABI (binary) compatibility libodp version to use with different implementations.
- Non ABI compat version uses “native types” for more speed optimization.
- ODP defines API and does not put any restrictions for implementations.

- Known projects similar to ODP are: dpdk, netmap, pf_ring, libpcap.
- Known implementations are: **Linux generic**, **DPDK based**, Cavium Thunder-X, Freescale based on DPAA, TI Keystone II, Kalray MPAA.

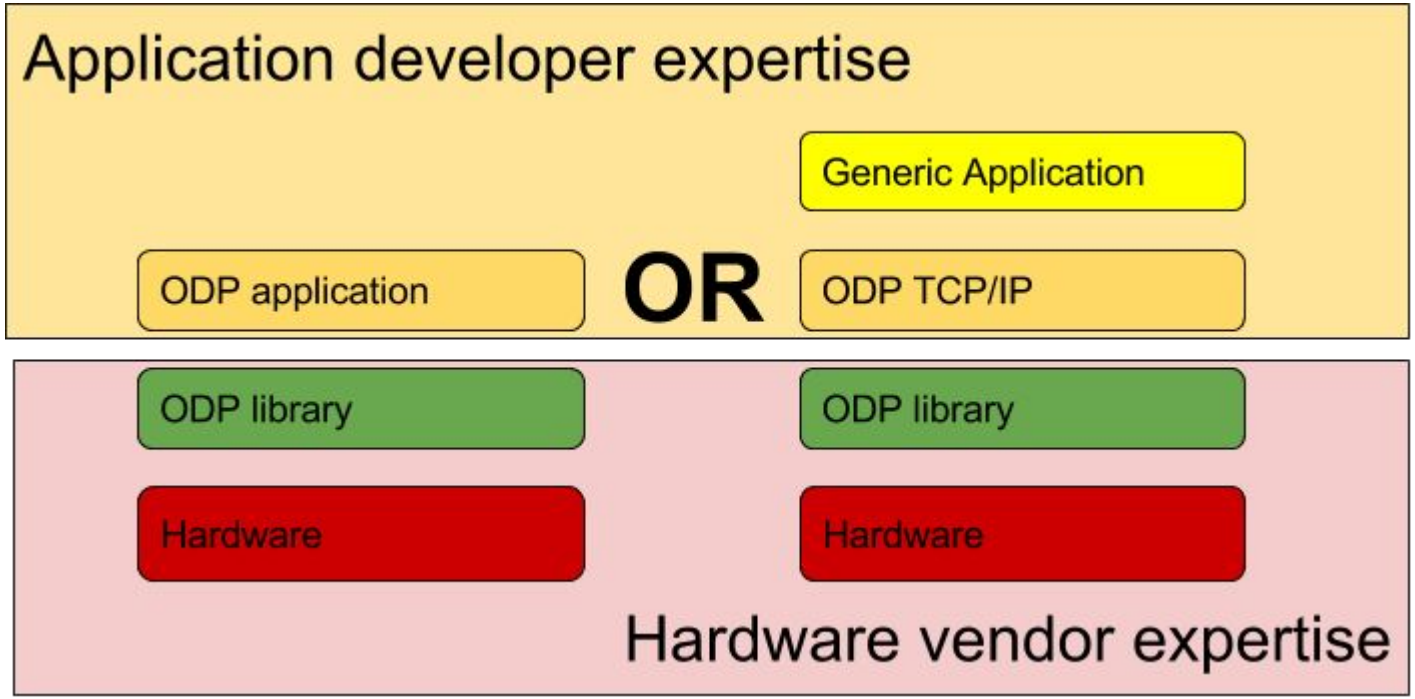
Current ODP implementations done in linux user space. Where hardware delivers packets directly to user space memory. If ODP API can not be accelerated with hardware, than software realisation is used.

Packets per nanoseconds and cpu instructions

Speed	bytes/second	Maximum PPS	CPU cycles @1Ghz 1 cpu cycle = 1 ns
100Mbps	12,500,000	148,810	6720
1Gbps	125,000,000	1,488,095	672
10Gbps	1,250,000,000	14,880,952	67,2
100Gbps	12,500,000,000	148,809,524	6,72 ns

Problems solved by ODP

- Applications are for hardware agnostic API. They are portable!



- Maximum performance guaranteed by hardware vendor doing ODP implementation for specific hardware.

Why not linux kernel stack?

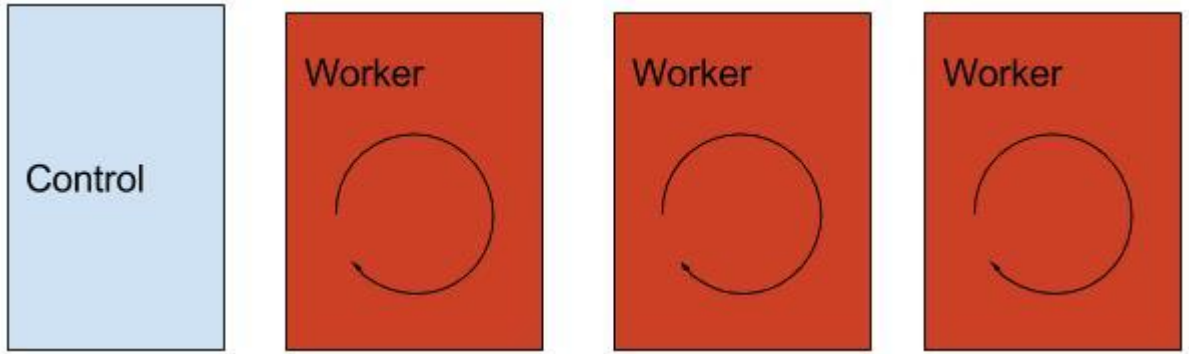
- Long and unpredictable latencies for network packets;
- Memory size for metadata for network packet (SKB metadata, TCP/IP and netfilter metadata) - few kilobytes for 64 byte packet;
- System fight for cpu time between kernel and userland, spending cpu cycles is not optimal;
- TLB entries for both userland and kernel, no Huge Pages support;
- Packet body copy, strip out VLAN tags and later reinsertion in userland for AF packet;
- Other interrupts involved which may delay execution;
- No hw accelerators support;

- Crash in kernel makes whole system not operable and you have to reboot. Reboot whole system vs one single application.

ODP run model

- Main control thread/process to allocate memory, initialize packet i/o and configure components;

- Number of worker threads/processes owns core to process packets.



ODP internals: packet and packet I/O

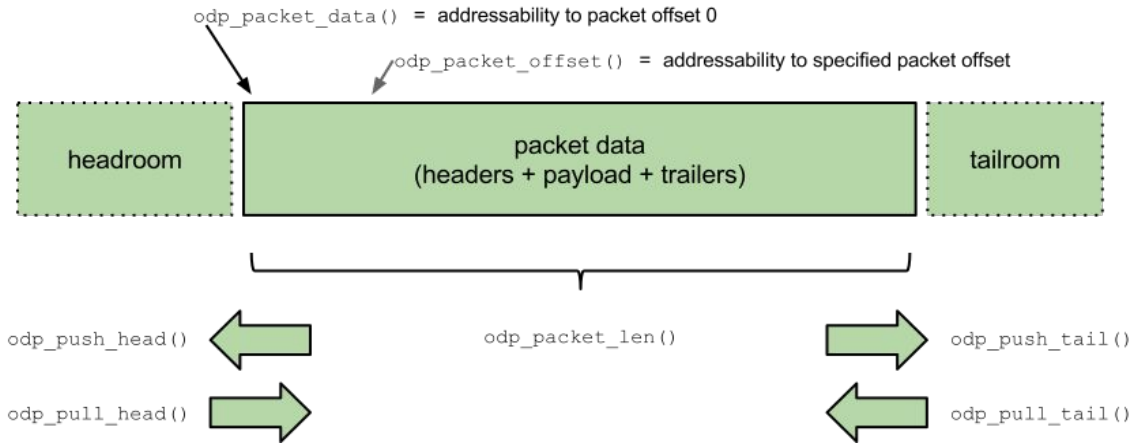
odp_pktio_t - is abstraction type of packet input/output

odp_pktio_start() , odp_pktio_stop() and etc

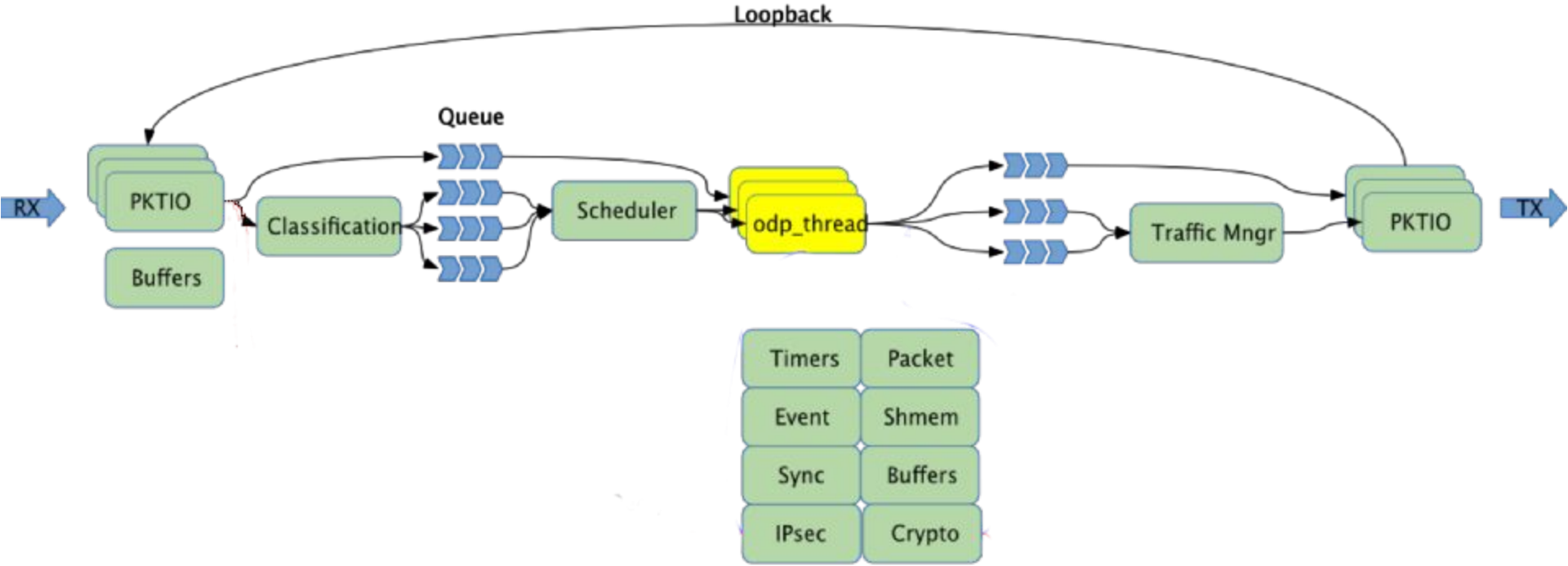
odp_packet_t - abstract type for network packet

int odp_pktin_recv (odp_pktin_queue_t queue, odp_packet_t packets[], int num)

Int odp_pktout_send (odp_pktout_queue_t queue, const odp_packet_t packets[], int num)



ODP components



ODP internals: ODP_PKTIN_MODE_DIRECT

```
odp_pktin_queue_t    queue_in,
odp_pktout_queue_t  queue_out

odp_pktin_queue(pktio, &queue_in, 1);
odp_pktout_queue(pktio, &queue_out, 1);

while (1)  {
        pkts = odp_pktin_recv_tmo(queue_in, pkt_tbl, MAX_PKT_BURST,
                                  ODP_PKTIN_NO_WAIT);
        sent = odp_pktout_send(queue_out, pkt_tbl, pkts);
}
```

ODP internals: ODP_PKTIN_MODE_SCHED

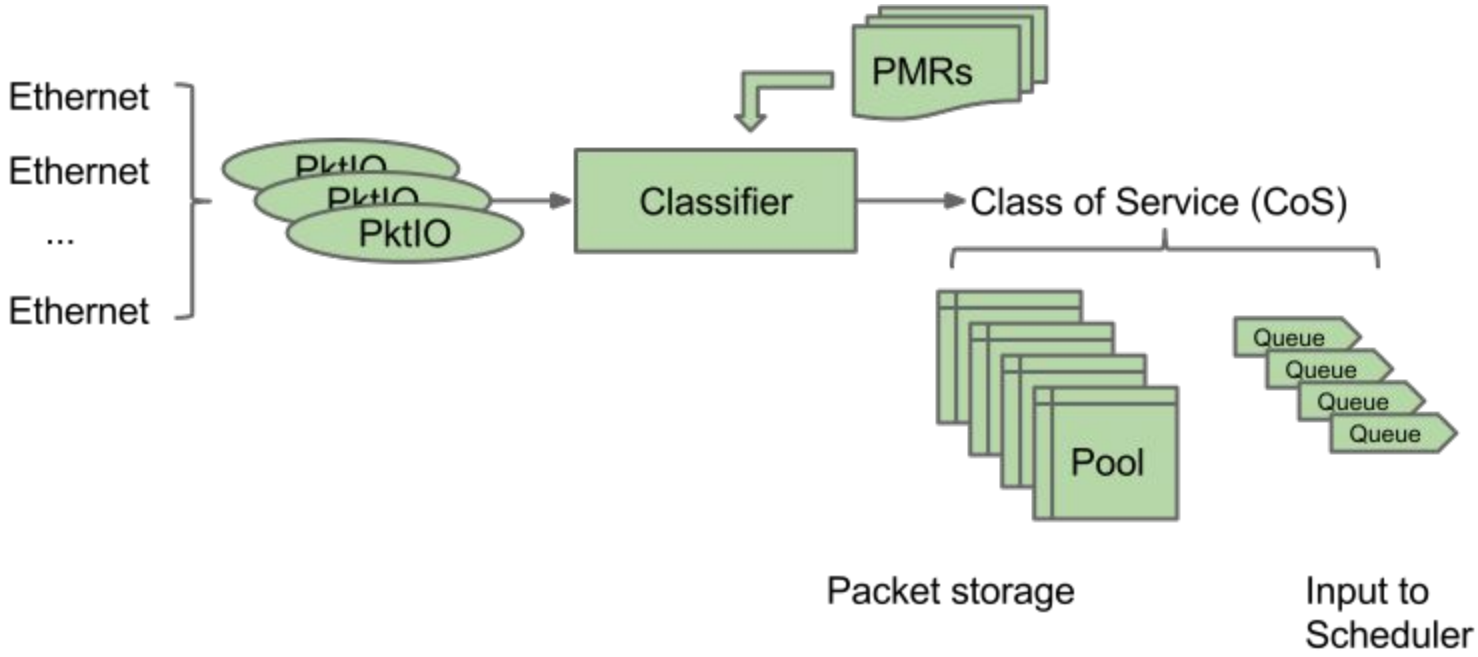
```
while (1)  {
        odp_event_t ev;
    ev = odp_schedule();

    switch (odp_event_type(ev)) {
    case ODP_EVENT_PACKET:
        pkt = odp_packet_from_event(ev);
        sent = odp_pktout_send(queue_out, pkt, 1);
    case  ODP_EVENT_TIMEOUT:
    case ODP_EVENT_CRYPTO_COMPL:
   }
}
```

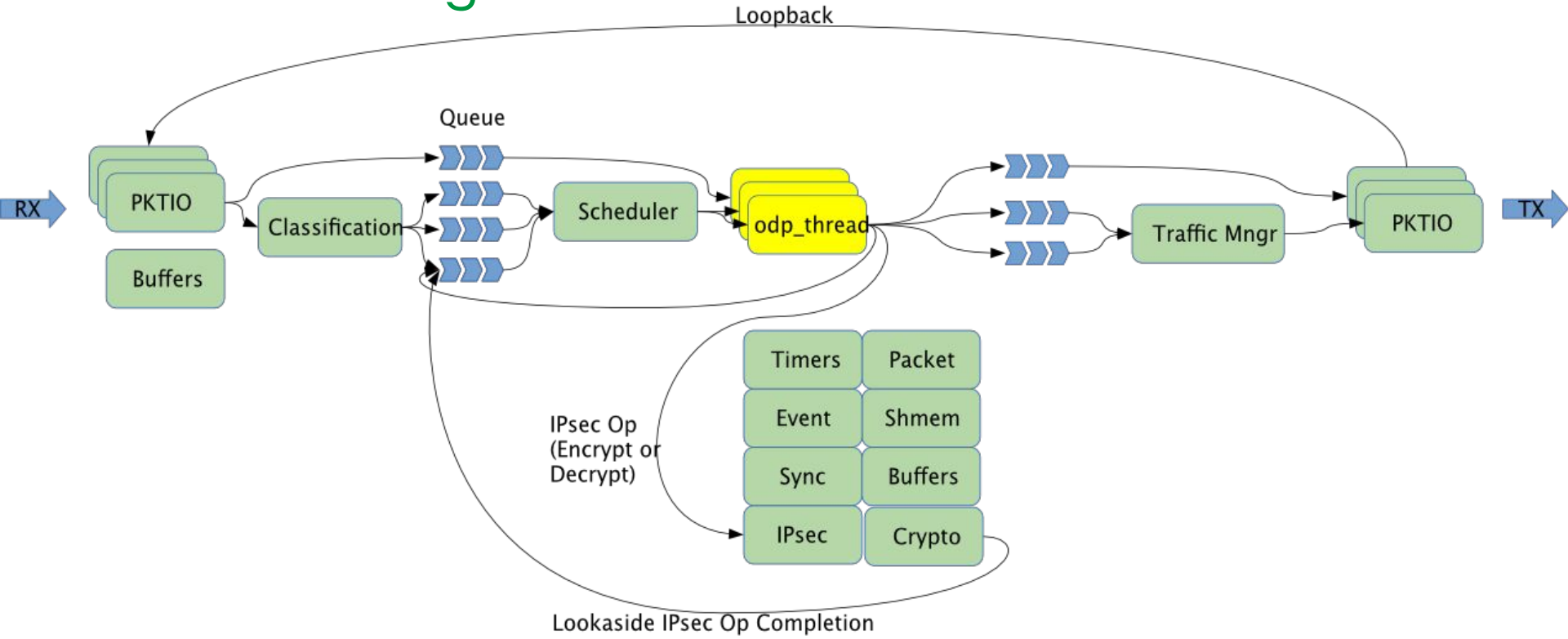
odp_event_t

odp_buffer_t odp_packet_t odp_timer_t

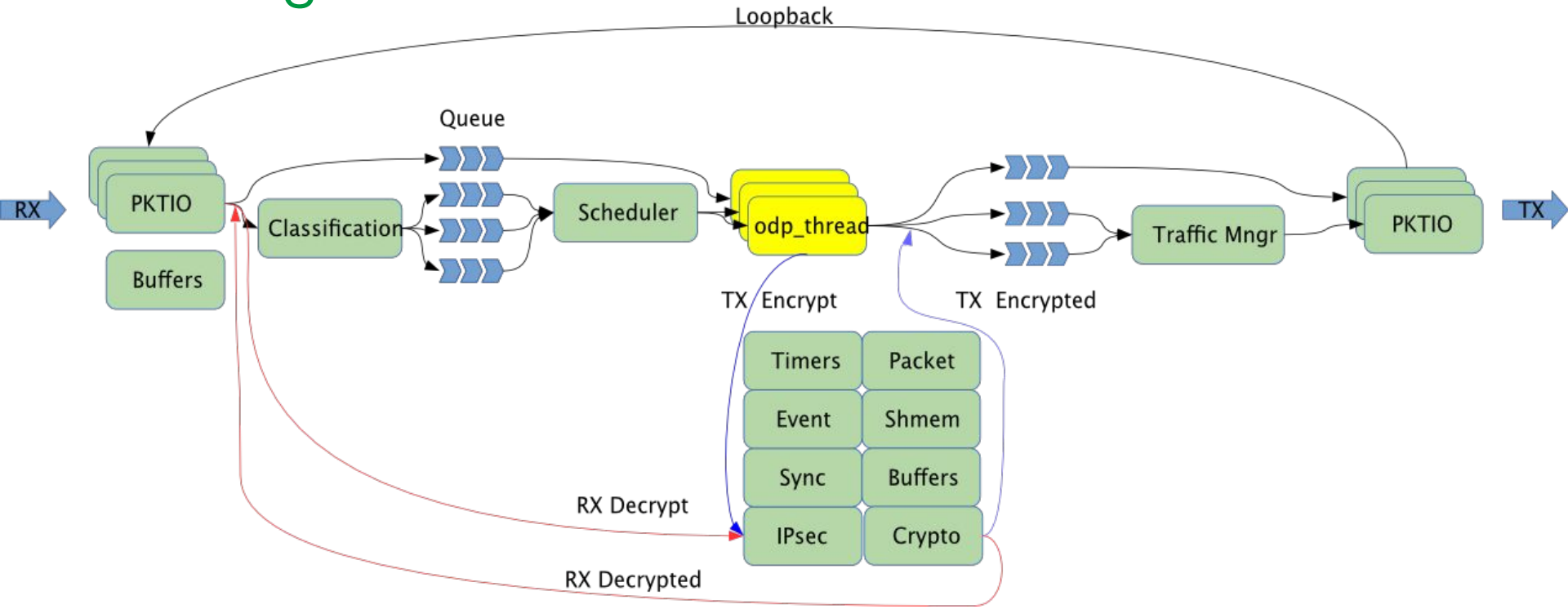
ODP internals: pool, queues and classifier



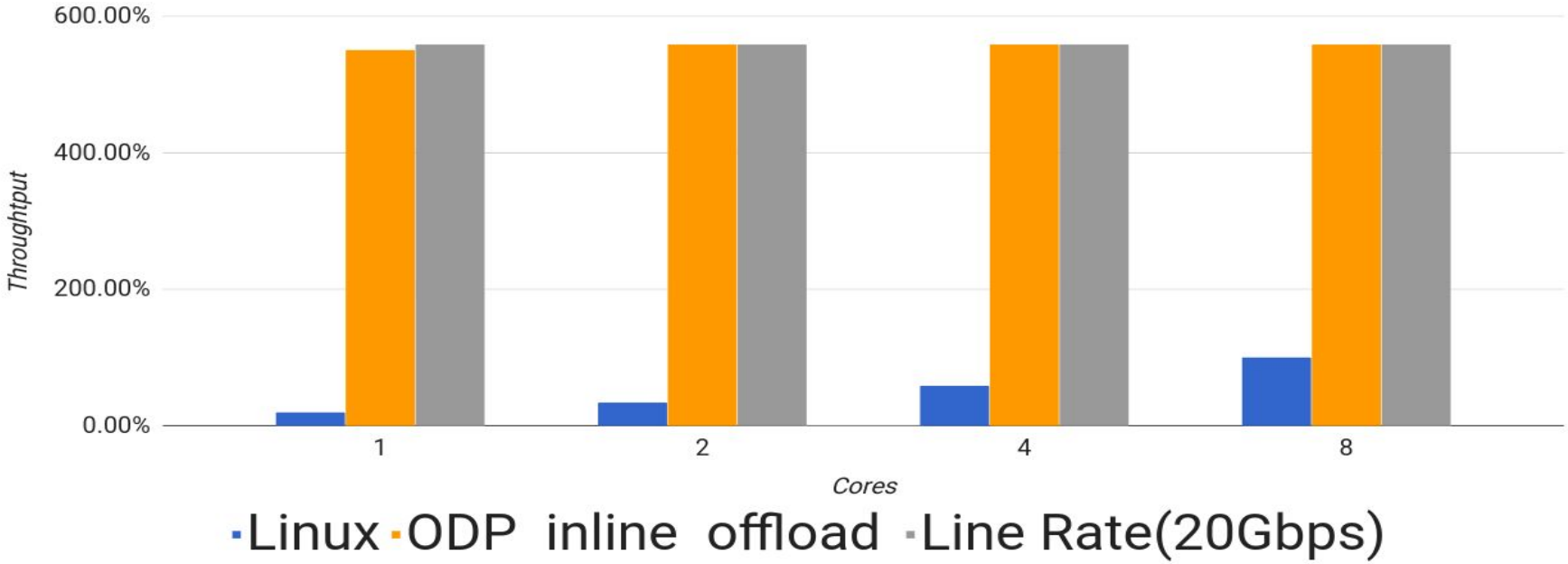
ODP internals: advanced features in short: ipsec look aside offloading



ODP internals: advanced features in short: ipsec inline offloading



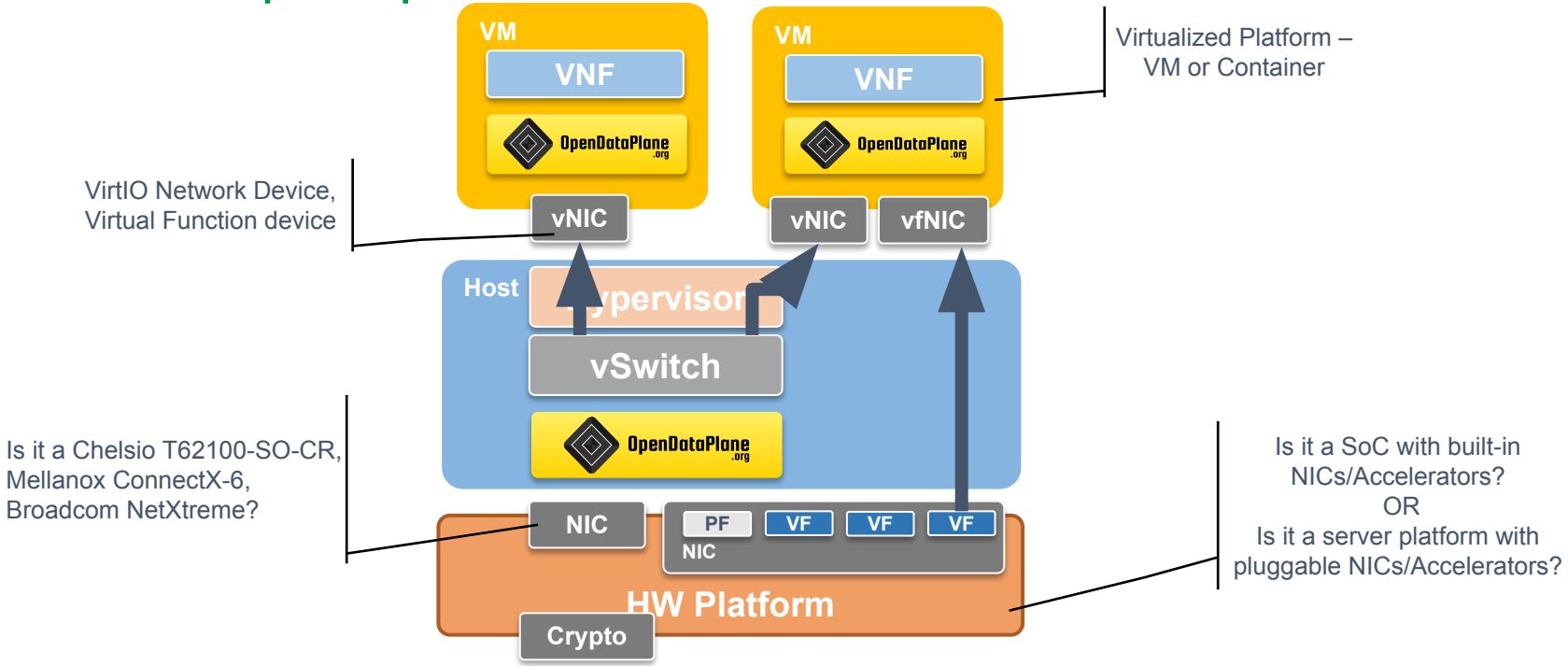
IPSEC IMIX Traffic Performance Comparison
NXP LS2088 board (RFC api)



Linaro

OpenDataPlane .org

NFV - ODP perspective



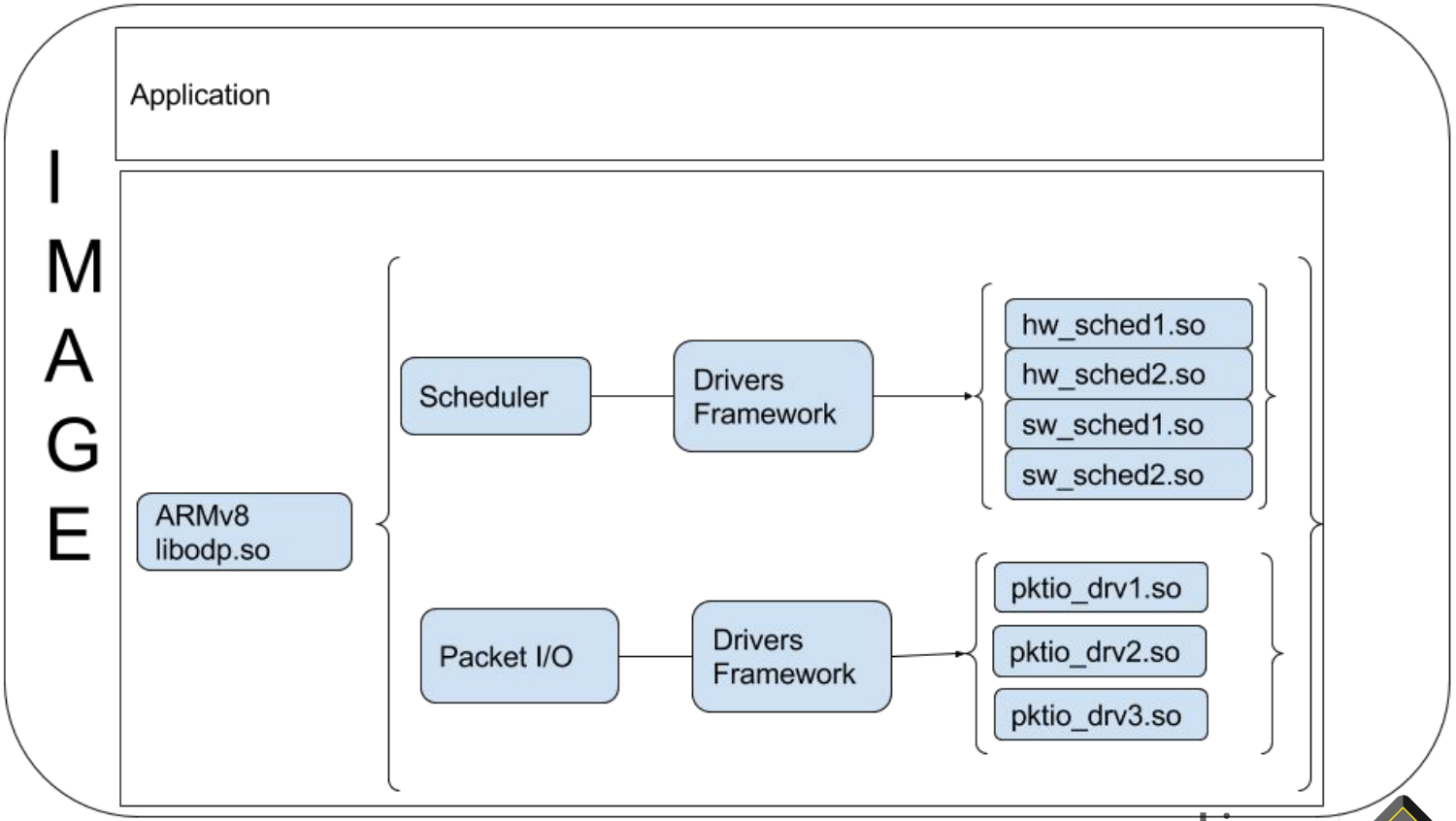
Platform components identified during run time

Single application binary to run on all environments

Linaro

OpenDataPlane .org

ODP for NFV high level design



How to contribute and get involved to ODP

- Everyone can contribute to ODP!

lng-odp@lists.linaro.org http://opendataplane.org

- Member companies of Linaro Networking Group define strategy and priority for project and pay fees to support project maintenance and required equipment. Contact is linaro.org

Other ODP implementations

https://www.opendataplane.org/downloads/

- Cavium (Thunder-X)
- Freescale (based on DPAA)
- Texas Instruments (Keystone II)
- Kalray (MPAA)
- Linaro (based on DPDK)

Questions?

OpenDataPlane
.org