OPNFV SUMMIT 2016 (BERLIN)

DPACC AND DPDK SOLVING NFV ACCELERATION

Keith Wiles

June 2016

Legal Disclaimer

General Disclaimer:

Technology Disclaimer:

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].

Performance Disclaimers:

Cost reduction scenarios described are intended as examples of how a given Intel- based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

DPACC and DPDK overview

DPACC design overview with DPDK

Adding VPP from FD.IO to DPDK to enhance NFV

What is TLDK (Transport Layer Development Kit)?

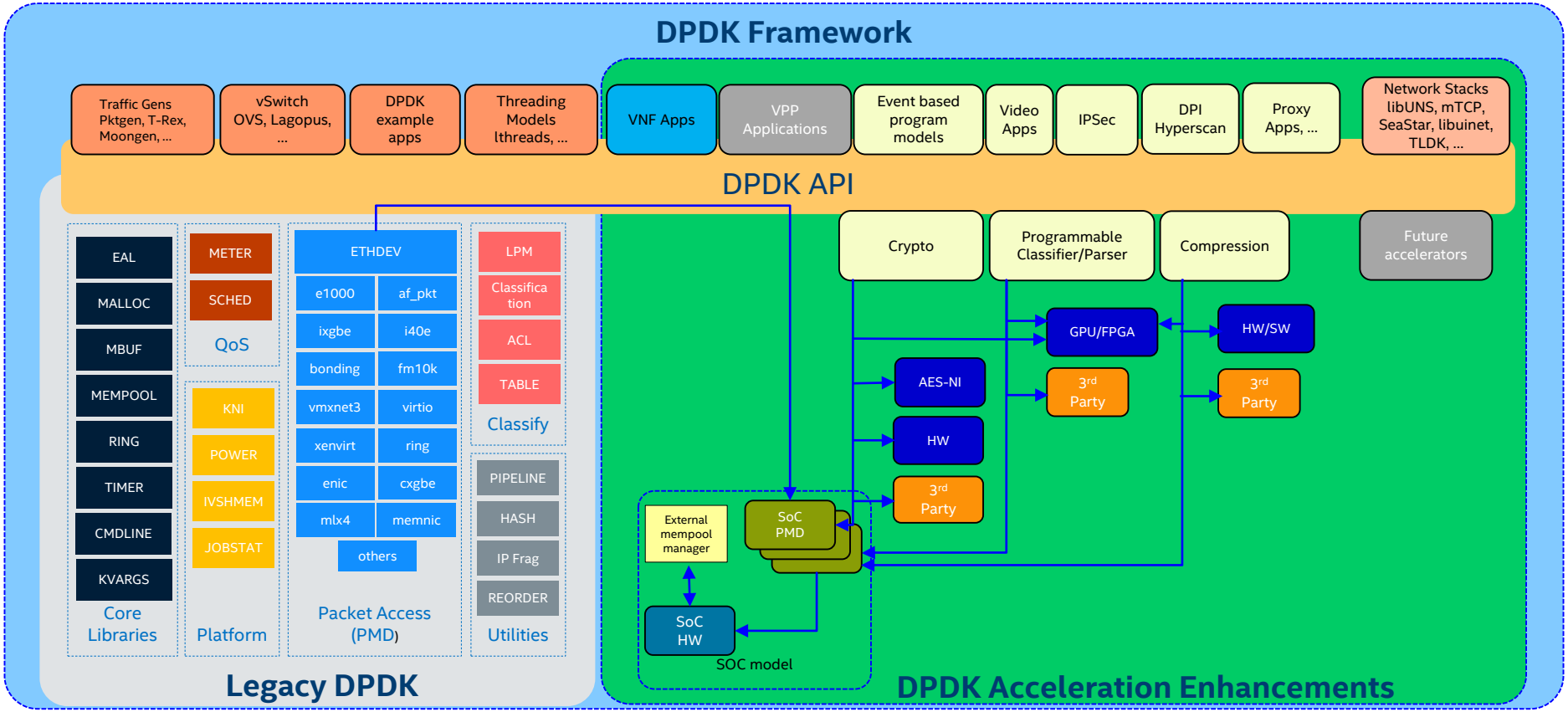
Enhancements to DPDK to support DPACC

Summary

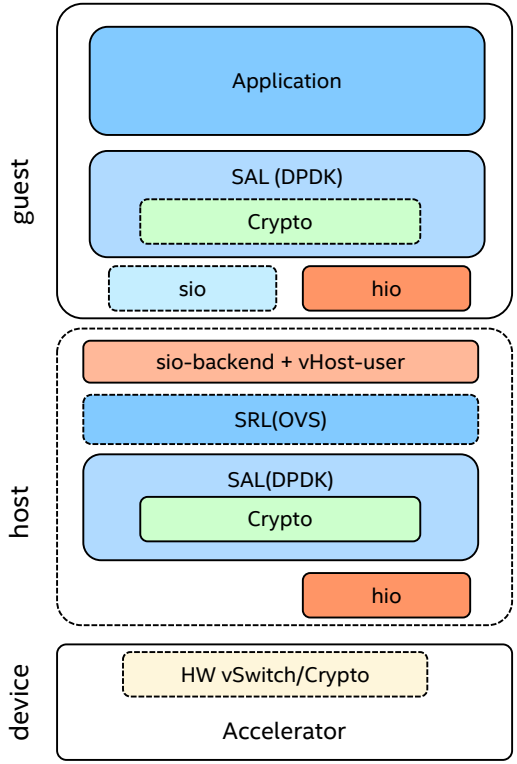
DPDK

Simple overview

DPDK Overview



DPACC and DPDK design



Focus on one design as a basic high level view:

Software Acceleration Layer is the software to hardware abstraction layer

Software Acceleration Layer makes possible additional services which can be controlled by the orchestration layer

sio-backend + vHost-user is normally in the SAL or SRL layer, but shown here to illustrate vHost in the host.

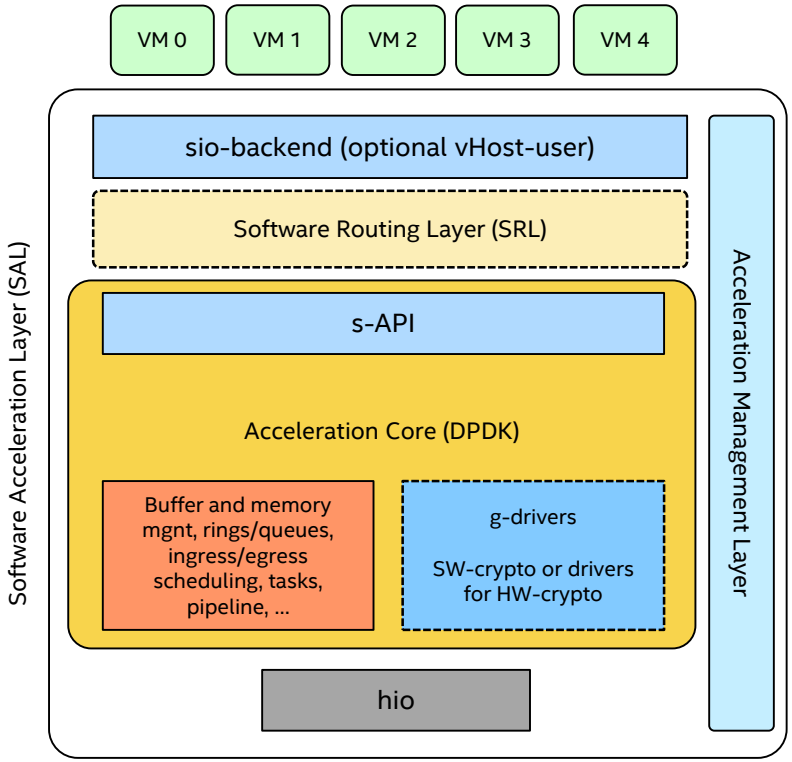
A SAL in the guest allows for the best performance selection

- Direct access to hardware acceleration via SR-IOV, SOC-specific interface or other pass-though
- Able to do software acceleration in the guest

SRL or HW vSwitch adds VM2VM routing or switching of packets

A SAL in the host gives scalability for non-accelerated VMs and/or native applications

DPACC: Acceleration Layer for Host (Hypervisor)



SAL: Software Acceleration Layer

Provides an abstraction between SW and HW, plus able to support multiple devices at the same time

sio-backend: backend of paravirtualized drivers

- vHost-user: User space based VirtIO interface
  - optional for VM ← → host access

s-API: APIs for utilizing an AC (APIs from the AC)

g-drivers: General driver for each device type

- Implemented in software or the frontend to the hardware (may be different for different acceleration functions)

hio: Hardware I/O interface

- Non-virtualized, accessed only by host SAL

AC: Software/Hardware Acceleration Core (DPDK)

SRL: Software Routing Layer (OVS)

AML: Acceleration Management Layer

VPP WITH DPDK

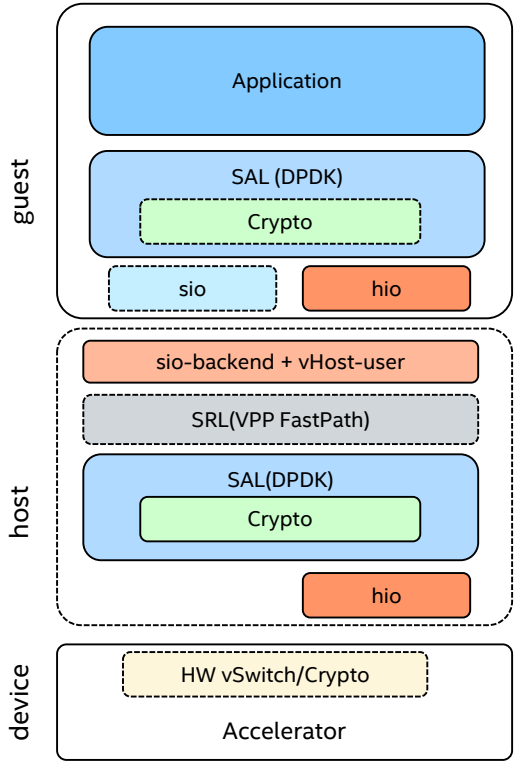
Quick overview

DPDK enhancements for VPP

DPDK contains HW and SW crypto acceleration support

- Add HW/SW crypto support to VPP as a new or modified graph node
  - Adding DPDK crypto support to VPP can transparently support a range of hardware and software solutions
- DPDK contains many accelerators and accelerated software for a number of system architectures, which can be utilized by VPP transparently

DPACC and VPP design



Software Routing Layer (SRL) can be VPP (Vector Packet Processing) code for a FastPath design of L2/L3 forwarding

- VPP can also be in the guess for location fastpath support
- VPP can replace the vSwitch at the SRL layer and provide added functionality to the guess or host

What do we need to do?

- Need to address the patches in VPP to DPDK and reduce the amount of changes required for VPP support
- Modify DPDK to pick up some of the enhancements in VPP
  - One area is around how data is referenced. VPP uses an 32 bit index based on cache line and DPDK uses a pointer, which could be 64 bits
    - Adding more space to some critical structures in DPDK, but we have to understand the performance effect to the system
  - Move the RTE_MBUF 'next' pointer to the first cache line, but this effects performance by having to move other data to the second cache line
- Better support of DPDK devices in VPP can improve the performance and portability of VPP across many different architectures

TLDK (TRANSPORT LAYER DEVELOPMENT KIT)

Quick overview

TLDK (Transport Layer Development Kit)

TLDK is an open source FD.IO project (https://wiki.fd.io/view/TLDK)

- The TLDK project will implement a set of libraries for L4 protocol processing (UDP, TCP etc.) and VPP graph nodes, plugins, etc Using those libraries to implement a host stack
- TLDK works on standalone DPDK or integrated into VPP to provide termination support
- Committers are:
  - Konstantin Ananyev
  - Keith Wiles
  - Ed Warnicke (IRC nick: edwarnicke)
- The first source code push by Konstantin a few weeks ago
  - git clone https://gerrit.fd.io/r/tldk
- Creating a UDP only design with clean scaling and performance
  - About 10 Mpps for 64 byte frames of terminated UDP traffic (Still more tuning needed)

TLDK (Transport Layer Development Kit) High Level View

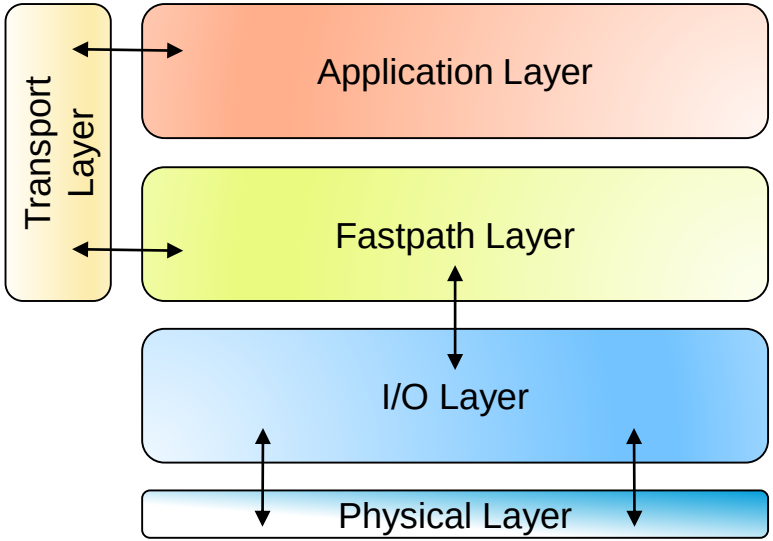
Transport Layer:
- A set of libraries to handle packets needing support for UDP/TCP/… which is TLDK

Physical Layer:
- Ports and other devices like crypto, compression, …

I/O Layer:
- DPDK is contained here as it provides the I/O abstraction to the physical layer

Fastpath Layer:
- The fastpath layer is today VPP and its graph nodes handling moving packets to application or back to the I/O layer after some type of processing

Application Layer:
- The application has a VPP graph node to send/receive packets to/from the application layer
- The application and/or fastpath layer uses TLDK library APIs

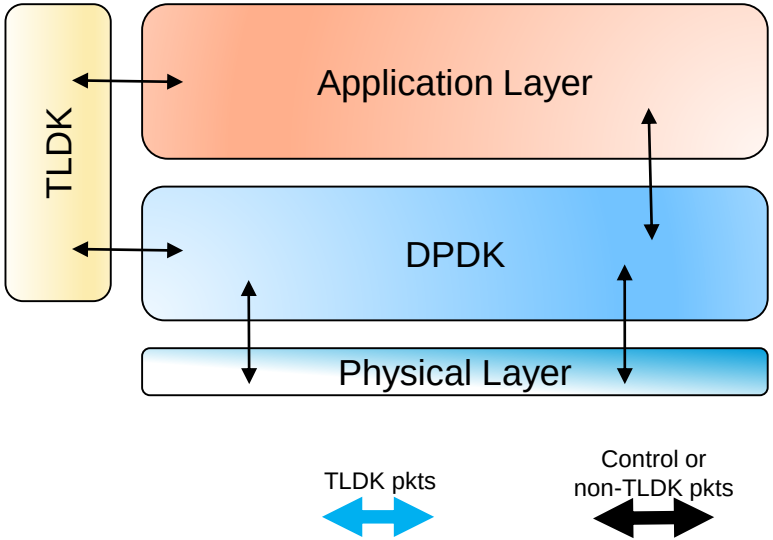


TLDK Uses case #1 with DPDK

TLDK:
- Handles packet I/O and protocol processing of packets
- Application sets up the UDP/TCP protocol contexts and then calls I/O routines in TLDK to start processing packets

Physical Layer:
- Ports and other devices like crypto, compression, …

DPDK:
- DPDK is contained here as it provides the I/O abstraction to the physical layer



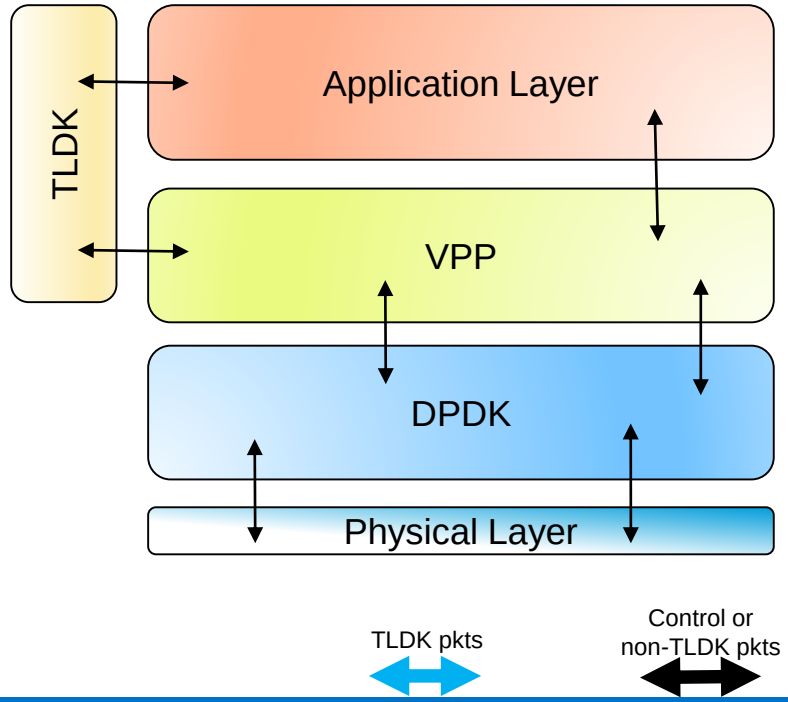
TLDK Uses case #2 with VPP

TLDK:
- Handles packet I/O and protocol processing of packets
- Application sets up the UDP/TCP protocol contexts and then calls I/O routines in TLDK to start processing packets

VPP Fastpath
- Using VPP as the first layer for packet processing before packets are sent to the application layer

Physical Layer:
- Ports and other devices like crypto, compression, …

DPDK:
- DPDK is contained here as it provides the I/O abstraction to the physical layer



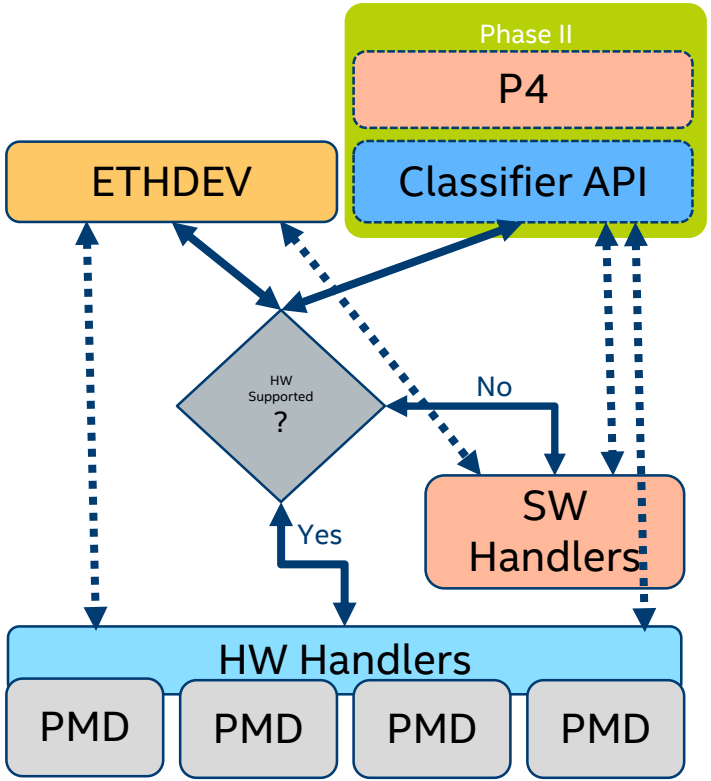
CLASSIFICATION FOR DPDK

WIP on adding classification support to DPDK, plus unifying the current APIs

ETHDEV and PMD layering

ETHDEV to PMD API requirements:

- Reuse existing APIs if possible with little to no change
- If a new API replaces and old API, the old API will be deprecated over time or never TBD
- Add new APIs only when required
- The ETHDEV to PMD layer interface needs to have as little change as possible, but common across all PMDs
- Weather Classifier API is on top of ETHDEV as a library or a new DPDK API along side ETHDEV is TBD
- P4 is shown here to state that P4 is above DPDK API and possible also above Classifier API TBD

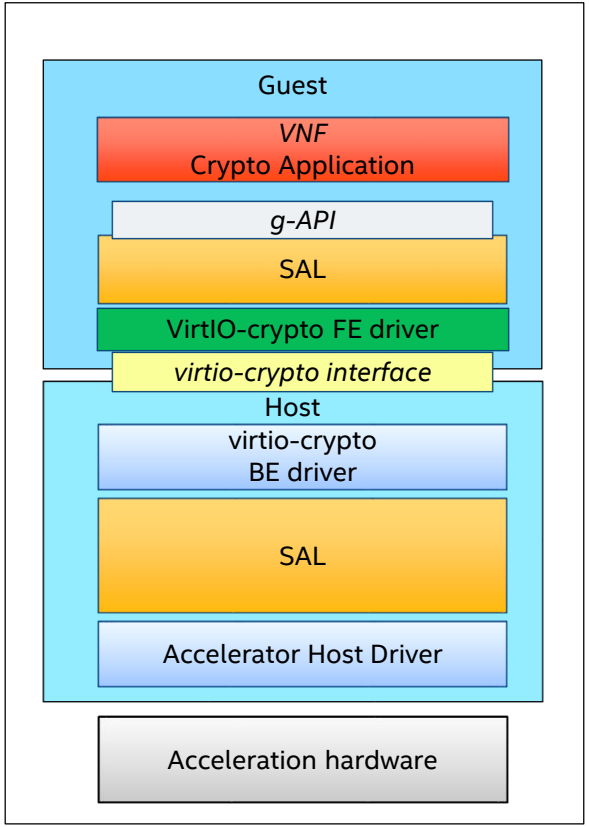


VIRTIO

Update on Intel’s work on Virtio

Lookaside VirtIO-crypto

- Intel has begun to define the specification for virtIO-crypto processing from the VNF to the host
- The specification currently focuses on the definition of symmetric and asymmetric algorithms
  - Specification shall be protocol agnostic but will accelerate compute intensive algorithms
  - Shall offer asynchronous and synchronous API invocation
- Intel shall work with the QEMU mailing list to gain acceptance of the specification
- Request flow:
  - VNF application makes crypto requests using the g-API
  - The SAL translates between the g-API and the virtIO-crypto interface
  - The virtIO-crypto front end driver places requests on a Vring
  - The BE virtIO driver removes the request from the Vring and calls the accelerator driver to submit the request to the accelerator



DPDK ENHANCEMENTS

Current and future enhancements

Enhancements to DPDK

New features:

- External Memory manager support in DPDK 16.11

Future support:

- Event based support for DPDK applications (TBD)
- New Virtio support for crypto devices
- vSwitch using VPP Fastpath, plus we get a lot of extra features
- Add FPGA and GPU support to DPDK for more accelerators
- Add compression support to DPDK for storage and other needs

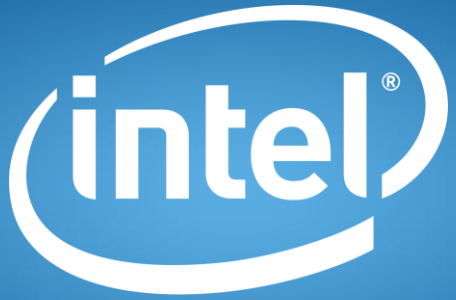
Summary

Adding VPP to DPACC will enhance the overall design with lots of extra protocol support in the fastpath

Reducing the delta between DPDK and VPP to better support FD.io

Adding termination of UDP and TCP support via TLDK to DPDK/VPP will increase native application performance

Enhancing Virtio for more accelerators like Crypto adds better support for NFV applications with better performance



experience
what's inside™