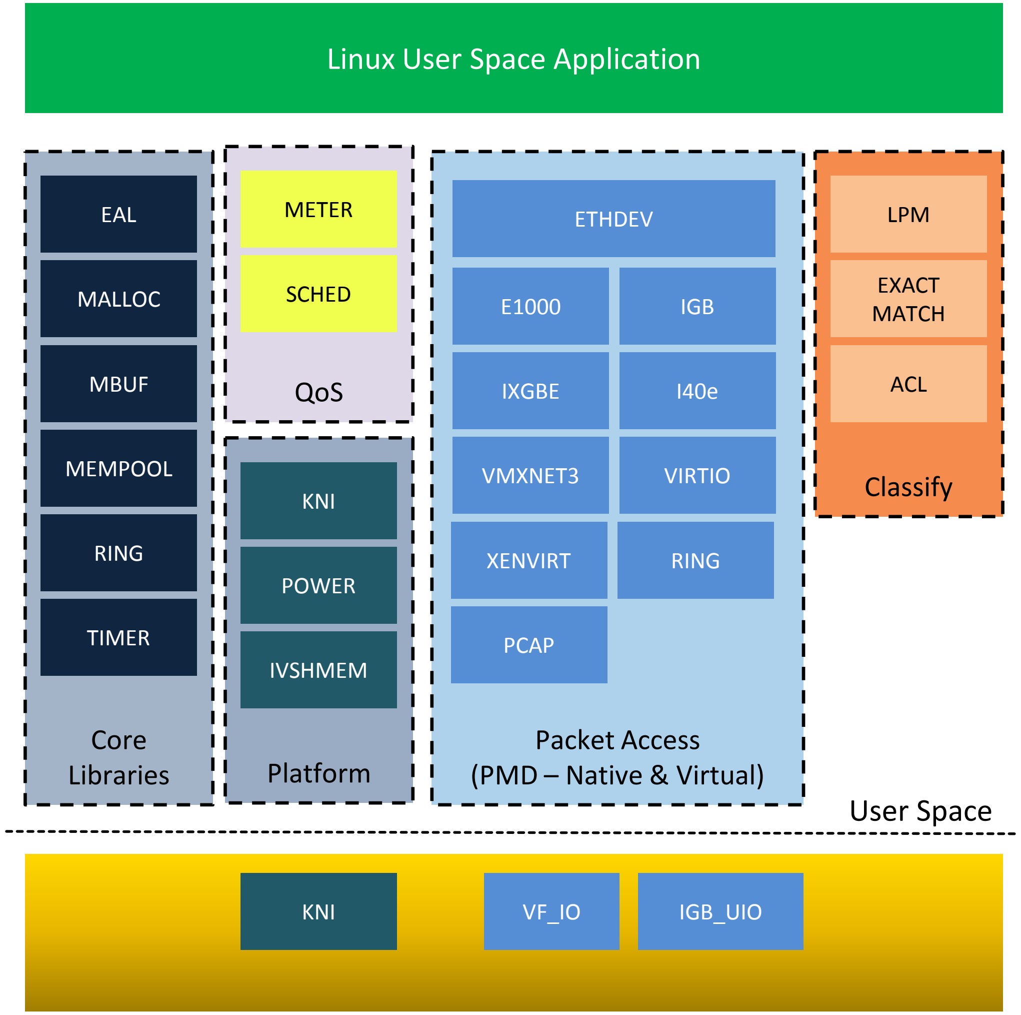
FUTURE ENHANCEMENTS TO DPDK FRAMEWORK

Keith Wiles, Principal Engineer, Intel Corporation

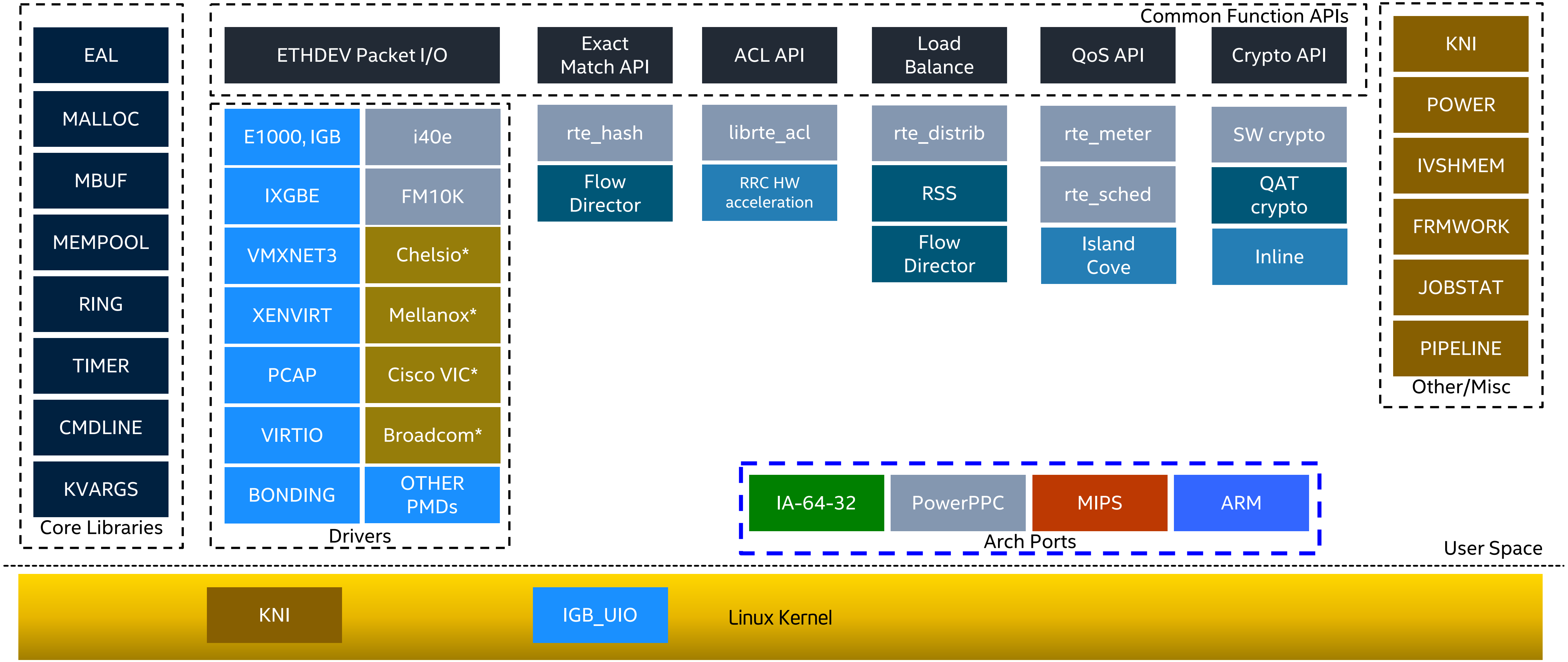
THE NEW CENTER OF POSSIBILITY

PRESERVING APPLICATION INVESTMENT WITH DPDK



- **Open-source (BSD license) community project (5+ years, version 2.1 latest) -- http://dpdk.org/**
  - All code is Open Source including the device drivers or PMDs (Poll Mode Drivers)
  - Optimized Linux User Space Library focused on data plane implementation on general purpose processors
  - Has been Very stable project with ABI versioning for APIs
  - Multi-architecture: x86, IBM, Freescale, EZChip(Tilera) support
- **Encompasses legacy platforms and newer acceleration platforms**
- **DPDK has a large application install base and included in Linux Distro's CentOS, Ubuntu, Red Hat, ...(Fedora)**
  - Adopted by standard OS distributions (FreeBSD, Linux) and many platform frameworks including VirtIO/Vhost and OpenvSwitch
- **Scalable solution to meet different NFV use cases**
- **Hardware acceleration complemented by software implementations for consistent set of services to applications**
- **Supports a large number of features like lockless rings, hash keys, ACL, Crypto, Match Action, buffer management and many others**
- **Has a large number of example applications and growing**
- **Supports any number of devices at the same time, using a 2 layer device model**

DATA PLANE DEVELOPMENT KIT ARCHITECTURE



* Other names and brands may be claimed as the property of others.

DPDK

DATA PLANE DEVELOPMENT KIT

DPDK - AE

What is Acceleration Enhancements for DPDK?

DPDK – WHAT DOES THE FUTURE HOLD?

Here are a few items we are thinking about and need help

- DPDK-AE (Acceleration Enhancements)
- What type of acceleration device types?
  - Crypto via hardware and software acceleration
  - DPI engine
  - Compression
  - Match Action and Flow Director APIs
- Adding support for SoC hardware
  - hardware memory management and event handling
- Network Stacks, light weight threading and other applications
- Focus on VirtIO performance and enhancements
- Support other language bindings

DPDK

DATA PLANE DEVELOPMENT KIT

DPDK – CRYPTO API

Overview of proposed Crypto API for DPDK

DPDK – CRYPTO USING HARDWARE AND SOFTWARE

Doing hardware and/or software crypto has some good advantages

- Hardware crypto can handle the large packets
- Software crypto can handle the smaller packets

Added advantages are:

- better performance over a range of packet sizes
- parallel execution with software and hardware crypto
- Abstracts the packet handling making it transparent to the application

CRYPTO DEVICE CONFIGURATION (RFC)

```
uint8_t dev_id; /* Crypto Device identifier */

/* Crypto Device Configuration */
struct rte_cryptodev_config dev_conf = {
        .socket_id = SOCKET_ID_ANY,
        .nb_queue_pairs = RTE_CRYPTODEV_DEFAULT_NB_QUEUE_PAIRS,
        .session_mp = {
                .nb_objs = RTE_CRYPTODEV_DEFAULT_NB_SESSIONS,
                .cache_size = RTE_CRYPTODEV_DEFAULT_CACHE_SIZE
        }
};

/* Configure device */
rte_cryptodev_configure(dev_id, &dev_config);

/* Crypto Device Queue Pair Configuration */
struct rte_cryptodev_qp_conf qp_conf = {
        .nb_descriptors = RTE_CRYPTODEV_DEFAULT_NB_DECS_PER_QUEUE_PAIR
}

/* Configure device queue pairs */
uint16_t qp_id;
for (qp_id = 0; qp_id < RTE_CRYPTODEV_DEFAULT_NB_QUEUE_PAIRS; qp_id++) {
        rte_cryptodev_queue_pair_setup(dev_id, qp_id, &qp_conf, rte_cryptodev_socket_id(dev_id));
}
```

CRYPTO SESSION MANAGEMENT (RFC)

```
uint8_t aes128_cbc_key[CIPHER_KEY_LENGTH_AES128_CBC] = { … };        /* AES128 cipher key */
uint8_t hmac_sha256_key[HMAC_KEY_LENGTH_SHA256] = { … };    /* HMAC SHA256 authentication key */

/* Cipher Parameters */
struct rte_crypto_cipher_params cipher_params = {
         .algo = RTE_CRYPTO_SYM_CIPHER_AES_CBC,
         .op = RTE_CRYPTO_SYM_CIPHER_OP_ENCRYPT,
         .key = {
                 .data = aes128_cbc_key,
                 .length = CIPHER_KEY_LENGTH_AES128_CBC
         }
}

/* Authentication Parameters */
struct rte_crypto_hash_params hash_params = {
         .op = RTE_CRYPTO_SYM_HASH_OP_DIGEST_GENERATE,
         .algo = RTE_CRYPTO_SYM_HASH_SHA256_HMAC,
         .auth_key = {
                 .data = hmac_sha256_key,
                 .length = HMAC_KEY_LENGTH_SHA256
         },
         .digest_length = DIGEST_BYTE_LENGTH_SHA256
}

/* Session Creation */
struct rte_cryptodev_session *crypto_session = rte_cryptodev_session_create(dev_id,
                          &cipher_params, &hash_params, RTE_CRYPTO_SYM_OPCHAIN_CIPHER_HASH);
```

CRYPTO OPERATION (RFC)

```
struct rte_mbuf *m;

/* Create a crypto op mempool */
struct rte_mempool *crypto_op_pool = rte_crypto_op_pool_create(“crypto_op_mempool",
                                            NB_CRYPTO_OPS, CRYPTO_OP_CACHE_SIZE, rte_socket_id());
/* Generate Crypto op data structure */
struct rte_crypto_op_data *crypto_op = rte_crypto_op_alloc(crypto_op_pool);

/* Set crypto operation data parameters, in this example, we are appending the generated digest to the
end of the mbuf data buffer and prepending the IV to the start of mbuf data buffer, these parameter do
not need to be within the mbuf */
rte_crypto_op_attach_session(crypto_op, crypto_session);

crypto_op->digest.data = (uint8_t *)rte_pktmbuf_append(m, DIGEST_BYTE_LENGTH_SHA224);
crypto_op->digest.length = DIGEST_BYTE_LENGTH_SHA224;

crypto_op->iv.data = (uint8_t *)rte_pktmbuf_prepend(m, CIPHER_IV_LENGTH_AES128_CBC);
crypto_op->iv.length = CIPHER_IV_LENGTH_AES_CBC;

crypto_op->data.to_cipher.offset = CIPHER_IV_LENGTH_AES_CBC;
crypto_op->data.to_cipher.length = DATA_LENGTH;

crypto_op->data.to_hash.offset = CIPHER_IV_LENGTH_AES_CBC;
crypto_op->data.to_hash.length = DATA_LENGTH;

/* Attach the completed crypto op data structure to the mbuf */
rte_pktmbuf_attach_crypto_op(m, crypto_op);
```

CRYPTO PACKET PROCESSING (RFC)

```
uint8_t dev_id;  /* Crypto device identifier */
uint16_t qp_id;  /* Queue pair identifier */

struct rte_mbuf *pkts[] = { m, … };                 /* Array of mbufs for crypto processing */
unsigned nb_pkts = sizeof(pkts)/sizeof(*pkts);      /* Number of valid mbufs in pkts array */

/* Enqueue mbufs for crypto processing on the specified crypto device queue */
unsigned pkts_enqueued = rte_cryptodev_enqueue_burst(dev_id, qp_id, pkts, nb_pkts);

/* Create an array of mbuf pointers to receive processed packets */
struct rte_mbuf *ppkts[MAX_BURST_SIZE];

/* Dequeue all available processed mbufs up to MAX_BURST_SIZE on the specified crypto device queue */
unsigned pkts_dequeued = rte_cryptodev_dequeue_burst(dev_id, qp_id, ppkts, MAX_BURST_SIZE);
```

DPDK

DATA PLANE DEVELOPMENT KIT

DPDK – FLOW CLASSIFICATION

Proposed flow classification support in DPDK

DPDK – FLOW CLASSIFICATION WITH HARDWARE

DPDK uses Flow Director APIs to manage flows

Match-Action API is a superset of APIs for flow classification

- The code is open source at https://github.com/match-interface

Match-Action API has a much large set of APIs to handle more flow classification needs, which we need to expose in the future

The Match-Action API is used under the Flow Director API for backward compatibility with current applications, while extending the applications to new hardware or software designs

DPDK – FLOW CLASSIFICATION WITH HARDWARE

DPDK uses Flow Director APIs to manage flows

The flows are currently managed in NIC devices, but we can extend FDIR APIs to support other hardware devices using Match-Action

Later we can continue to extend FDIR API to allow for more complex configurations and hardware designs using the full set of APIs with Match-Action APIs

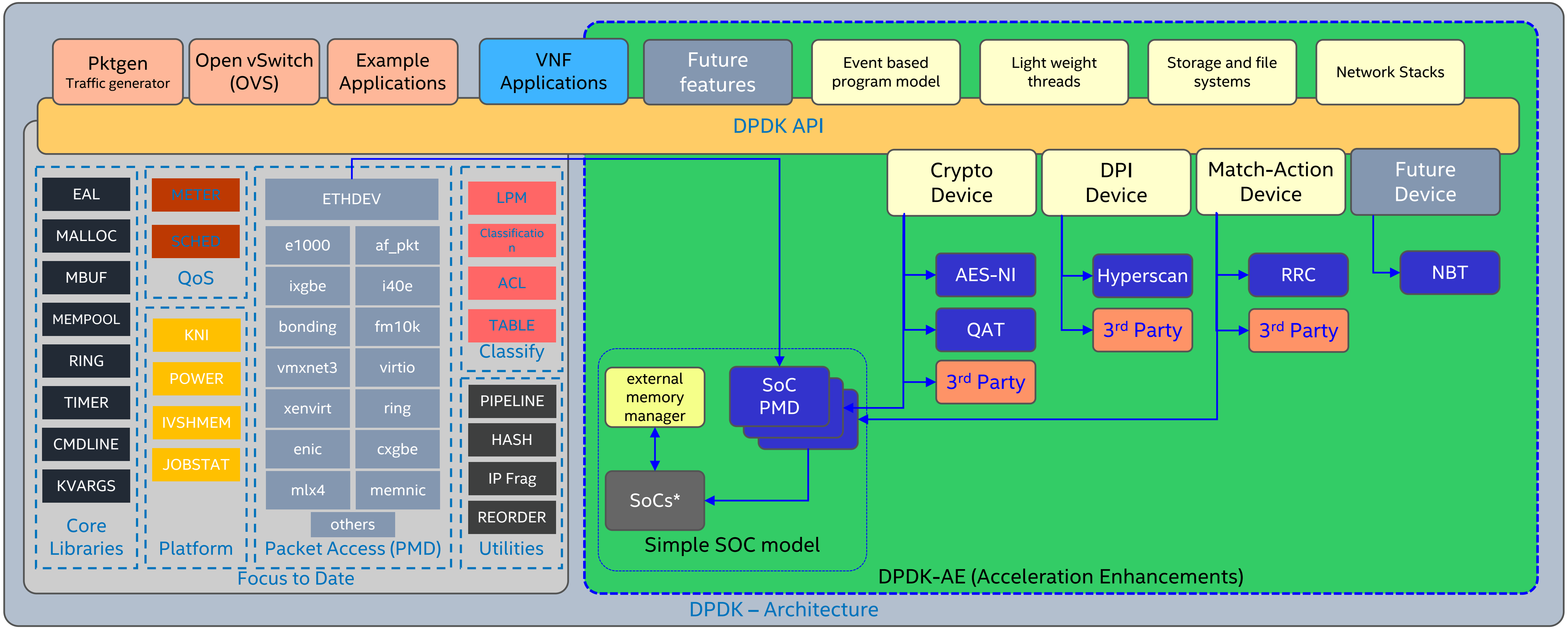
DPDK

DATA PLANE DEVELOPMENT KIT

DPDK – SOC SUPPORT

Proposed suggestion to add SoC support to DPDK

DPDK-AE (DPDK- ACCELERATION ENHANCEMENTS)



Pktgen
Traffic generator

Open vSwitch (OVS)

Example Applications

VNF Applications

Future features

Event based program model

Light weight threads

Storage and file systems

Network Stacks

DPDK API

EAL
MALLOC
MBUF
MEMPOOL
RING
TIMER
CMDLINE
KVARGS
Core Libraries

METER
SCHED
QoS

KNI
POWER
IVSHMEM
JOBSTAT
Platform

ETHDEV
e1000
af_pkt
ixgbe
i40e
bonding
fm10k
vmxnet3
virtio
xenvirt
ring
enic
cxgbe
mlx4
memnic
others
Packet Access (PMD)

LPM
Classification
ACL
TABLE
Classify

PIPELINE
HASH
IP Frag
REORDER
Utilities

Focus to Date

Crypto Device
AES-NI
QAT
3rd Party

DPI Device
Hyperscan
3rd Party

Match-Action Device
RRC
3rd Party

Future Device
NBT

external memory manager

SoC PMD

SoCs*

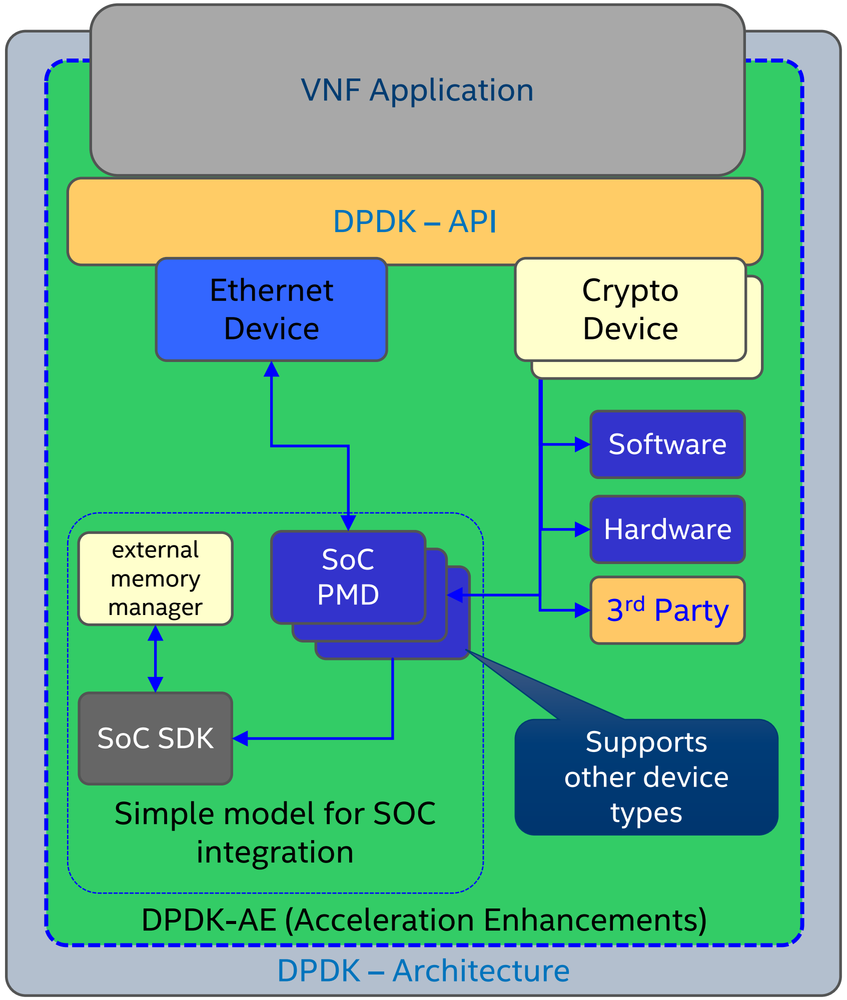
Simple SOC model

DPDK-AE (Acceleration Enhancements)

DPDK – Architecture

* We can adapt the SoC SDK via a DPDK PMD to maintain the highest performance

DPDK EXTENDING ACCELERATORS VIA SOC HARDWARE



SoC PMD: Poll Mode driver model for SoC devices

Provides a clean integration of SoC via a PMD in DPDK

- Hardware abstraction in DPDK is at the PMD layer
- DPDK-API: A generic API extended to support SoCs
  - DPDK provides a two layer device model to support many devices at the same time/binary, which can include SoC devices
  - Need to enhance DPDK with some SoC specific needs or features to support SoC hardware
    - Non-PCI configuration
    - External memory manager(s) (for hardware based memory)
    - Event based programming model
- SoC-PMD: Poll Mode Driver model for SoC
  - Allows SoC SDK's to remain private
- Supports ARM and MIPS DPDK ports to utilize these SoC designs

DPDK – CHANGES TO SUPPORT SOC HARDWARE

Enabling SoC hardware in DPDK requires a few enhancements

- Need a way to configure these non-PCIe devices
- Add support to DPDK mempool's to allow for external or hardware memory managers
- Add support for event based applications
  - e.g. Open Event Machine or others to utilize an event based programming model

Enlisting input for other enhancements to DPDK for SoC devices

DPDK

DATA PLANE DEVELOPMENT KIT

DPDK – NETWORK STACK

We need a stack the only question is how many?

DPDK – NETWORK STACKS AND OTHERS

DPDK needs to add network stack support for applications

- One network stack will not fill everyone’s needs today
- DPDK needs a clean API for multiple network stacks

Some examples for network stack types

- Network stack with very high performance with UDP and TCP
- Low latency network protocol support for High Frequency Trading
- IPv6 support and SCTP are a few protocols needed
- Light weight threading model for more then one thread per core to manage applications built using a multi-threaded design

What else needs to be supported?

DPDK – NETWORK STACK(S)

- Need a standalone Socket solution
  - For linked and non-linked applications
  - Need to support native applications using LD_PRELOAD
- Integration with the host platform network stack
  - Does the design appear as one stack or multiple stacks?
  - Need to support integration with the host stack
    - Netlink messages can be used to update the DPDK stack
    - Synchronize all stacks for routing and application data
  - Must be able to route packets to the local host stack from the accelerated stacks
    - Exception path or normal traffic like management

DPDK – NETWORK STACK(S)

How can we build these network stack?

- FreeBSD based stacks allow for up to date protocols
- Purpose built stacks are also needed for a given use case
- High performance and/or low latency stack are needed

- One option is to maintain up to date protocols/stacks
  - We can pull FreeBSD code from FreeBSD on demand then apply patches to allow us to build FreeBSD in DPDK style
  - Support as much of the protocols in FreeBSD as possible should be a goal IMO
  - Support File systems types using DPDK as a fast disk device

DPDK

DATA PLANE DEVELOPMENT KIT

DPDK – VIRTIO

Improve performance and enhancements

DPDK – VIRTIO

- Virtio is one of the primary interfaces for VM ← → Host
  - Needs to be enhanced to support more devices
  - Need to enhance performance of VirtIO
- SR-IOV is good for some cases in a VM
  - But does not scale to many VMs or containers
  - Very good in the host users pace to gain direct access to the devices
- Not all devices support SR-IOV and VirtIO is the only solution we have today as a standard

DPDK

DATA PLANE DEVELOPMENT KIT

DPDK – LANGUAGE BINDINGS

Adding other language bindings

DPDK – LANGUAGE SUGGESTIONS

DPDK needs other language bindings other than ‘C’

- Go from Google : golang.org
  - Go concurrency mechanisms make it easy to write programs and get the most out of multicore machines
- Swift will become Open Source later this year
  - Swift is easy to program and has some nice performance
- Java is also another good language

- These language bindings can add cleaner application development
  - Adding object oriented language designs
  - Give better support for multicore programming for the developer
  - Cleaner and faster development
- These compiled languages could provide very good performance with some trade offs

DPDK – SCRIPTING LANGUAGES

Interpreted languages like Lua (lua.org) or Python are reasonable for configuration

- Lua is very simple to support DPDK bindings to the language
- Python is object oriented, but harder/bigger to integrate

Using a scripting language can enhance configuration and writing applications like network protocol testers where performance is not the main requirement

What other language bindings would be reasonable?

DPDK – SUMMARY

- We need to add more acceleration supported hardware
  - Review and comment on the Crypto RFC
- Lets contribute other architecture types like ARM to DPDK.org
- Adding SoC enhancements to DPDK for more devices
- Lets move forward by adding more network stacks
- Add support to enhance native applications
- Lets add other language bindings to enhance configuration and application designs

- Lets collaborate on these and more…

DPDK

DATA PLANE DEVELOPMENT KIT

BUILDING A COMMON PLATFORM
FOR EVERYTHING AND EVERYWHERE!

THANK YOU