# QorIQ DPAA Enablement Software

## EUF-NET-T1309

**Shlomi Gridish**
SW Architect

November 2013

# Session Introduction

- As developers of Networking solutions over Multiple-core devices, having a suitable software base is crucial to your success and competiveness.
- Attending this session, will help you gain understanding of how Freescale enables your wins in this highly complex, highly competitive arena by freely supplying you with the industry's leading NetComm Software.
- As you migrate between different Freescale DPAA-based devices, often in a single system, you want to know that simple migration, scalability and portability are inherent inside the software supplied by your Silicon vendor. Well, the good news are that it is – come and learn how!

# Session Objectives

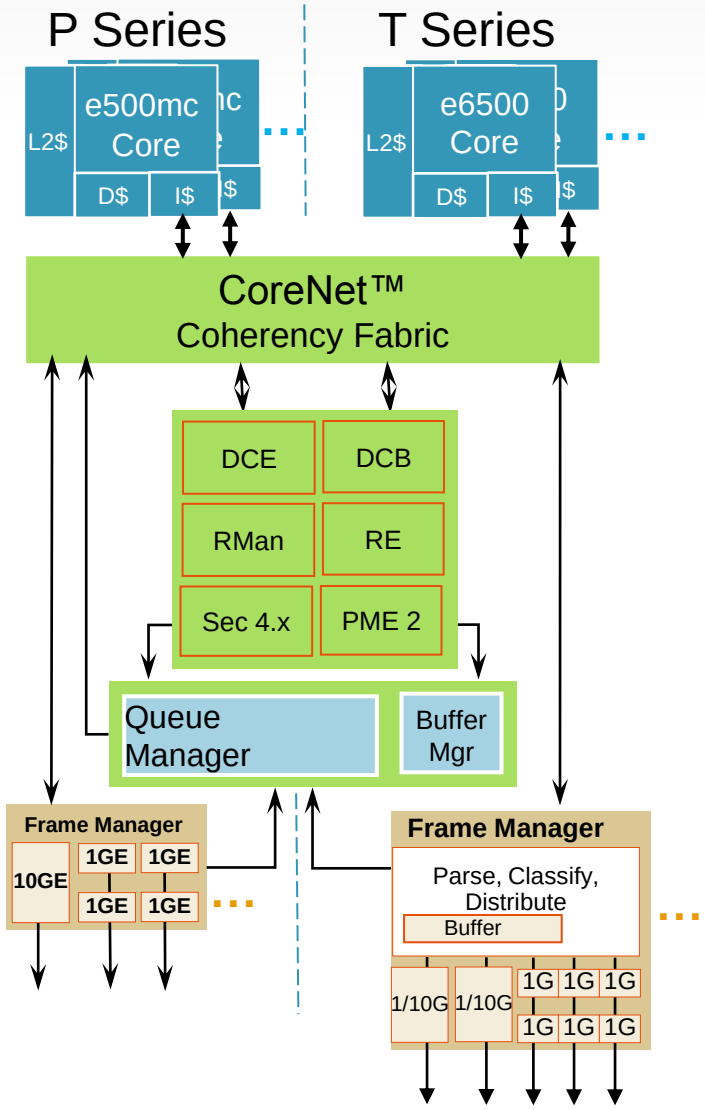
- The HW (brief review)
- The Software packages
- DPAA SW components
- DPAA SW in a System
- “PCD” Graphs & PCD Processing Elements
- Advanced Data Flows & DPAA Domains
- Looking into the future - LayerScape

# QorIQ DPAA 1.x

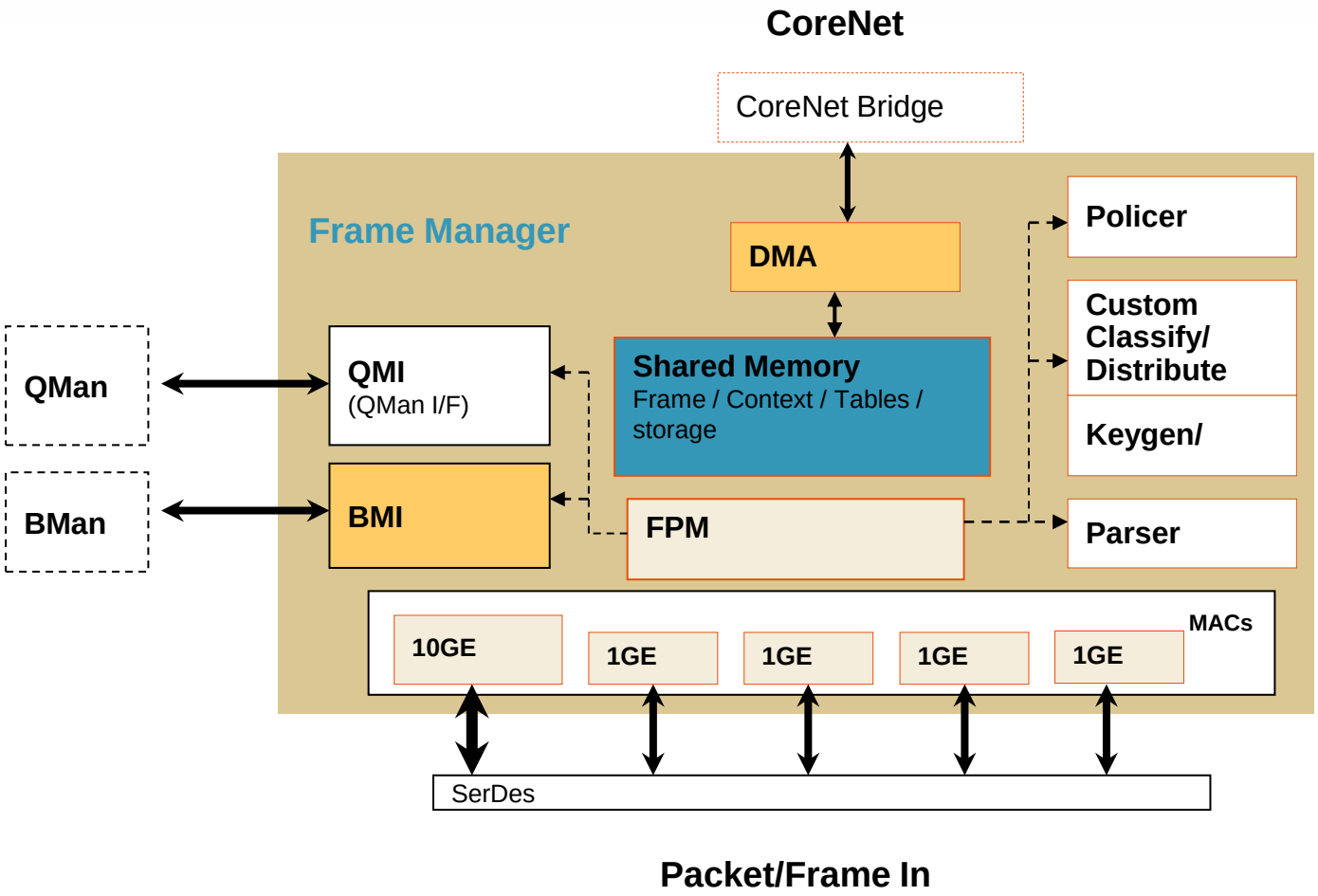
## HW overview

# Data Path Acceleration Architecture Philosophy

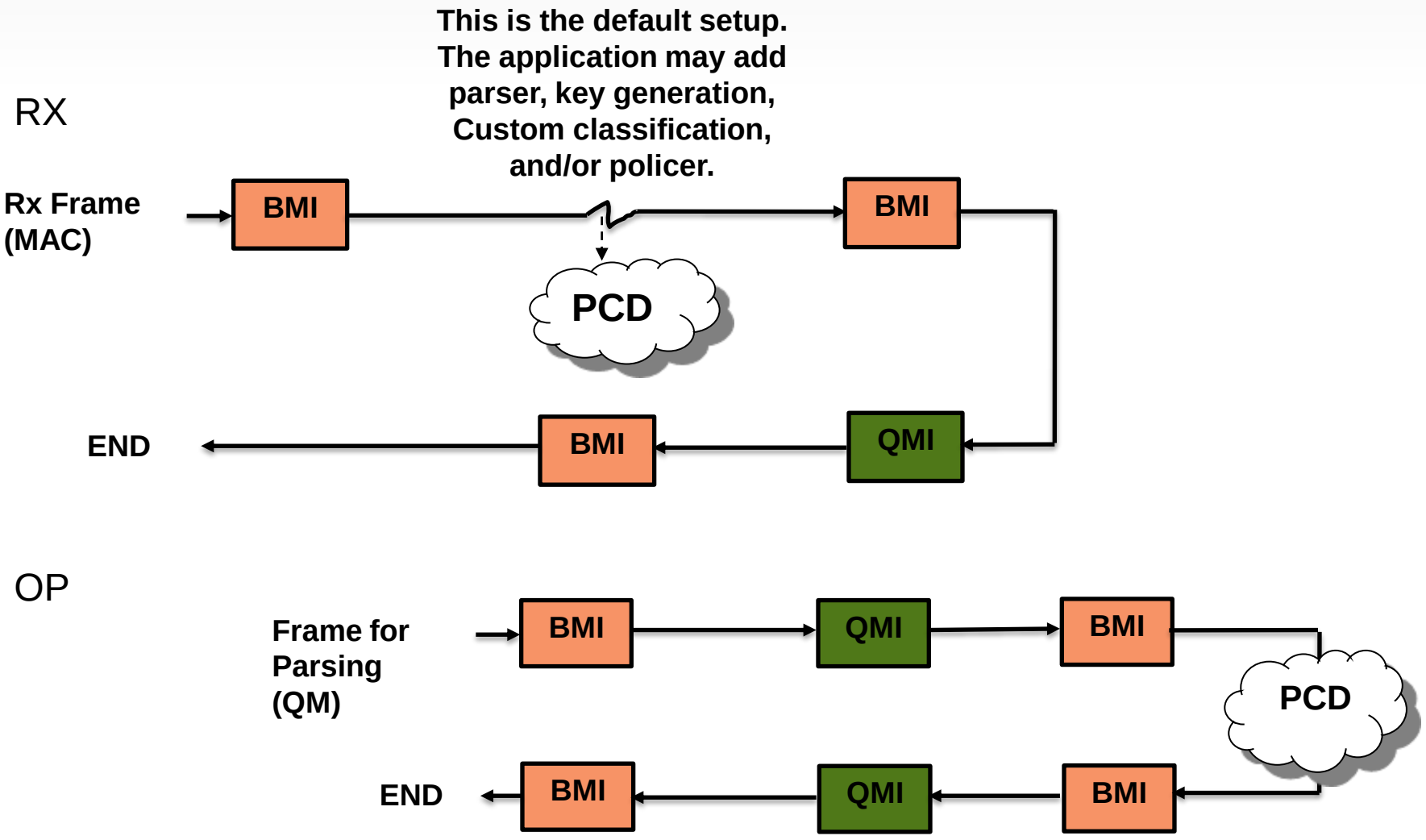
- DPAA is design to balance the performance of Accelerators with seamless Integrations
  - ANY packet to ANY core to ANY accelerator or network interface efficiently WITHOUT locks or semaphores.
- “Infrastructure” components
  - Queue Manager (QMan)
  - Buffer Manager (BMan)
- “Accelerator” Components
  - Cores
  - Frame Manager (FMan)
  - RapidIO Message Manager (RMan)
  - Cryptographic accelerator (SEC)
  - Pattern matching engine (PME)
  - Decompression/Compression Engine (DCE)
  - DCB (Data Center Bridging)
  - RAID Engine (RE)
- CoreNet
  - Provides the interconnect between the cores and the DPAA infrastructure as well as access to memory.



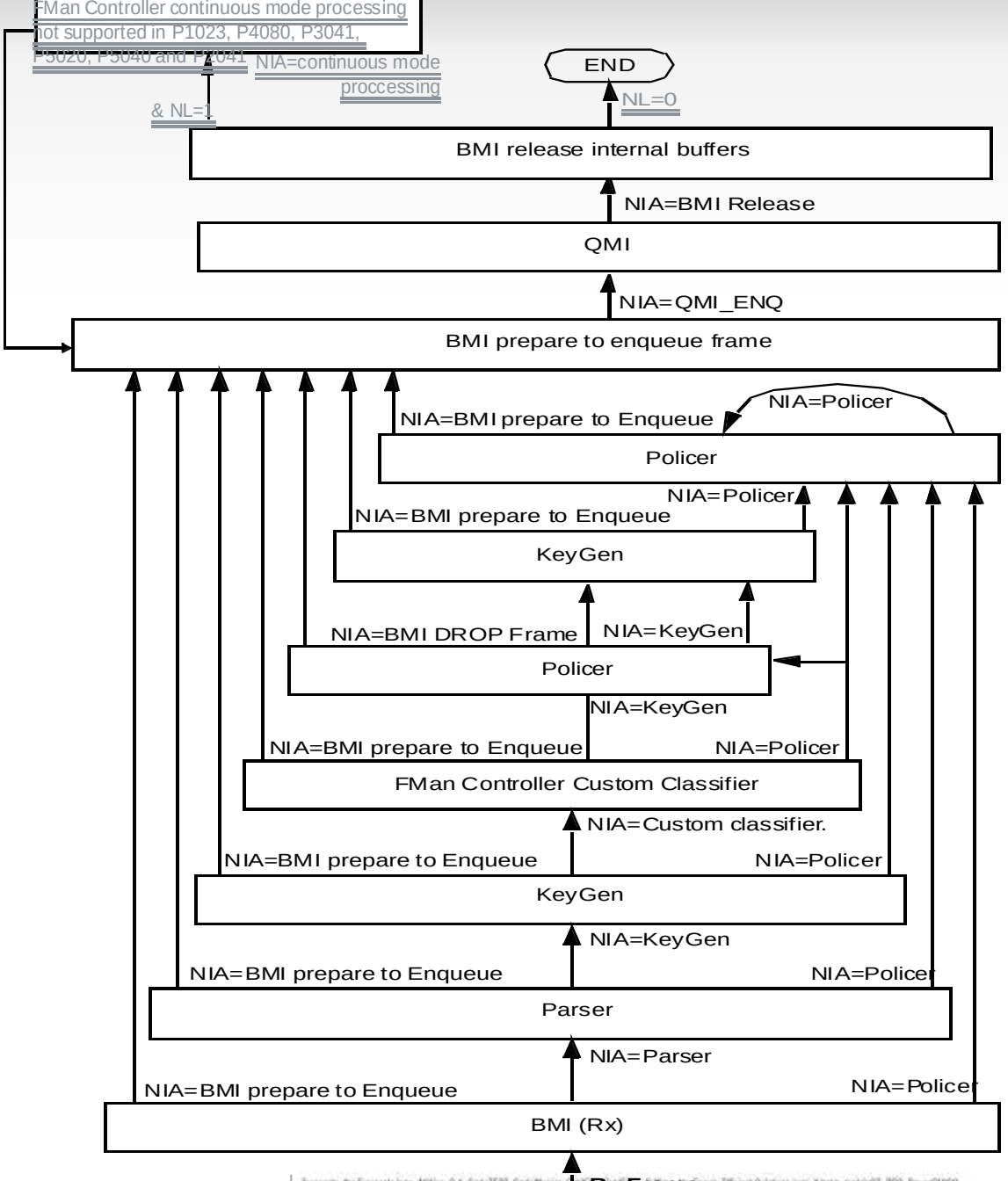
# FMan Internal Architecture



# Basic RX/OP Data Flow



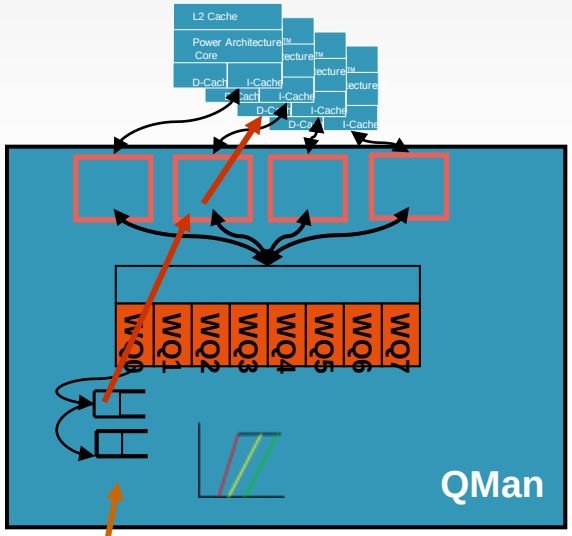
# FMan PCD –
## Data Flows

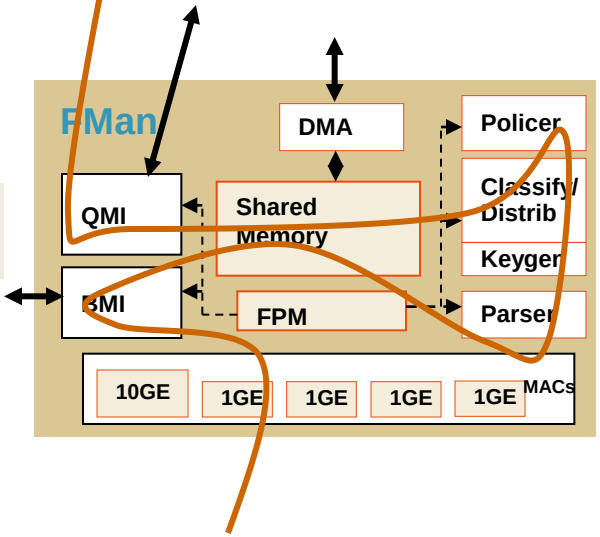


# Ingress/Egress Pipeline

## Ingress

- Core dequeue & processing
- QMan active queue mngt and scheduling (WRED)
- QMI instructs QMan to enqueues FD
  BMI releases internal buffer on completion
- BMI instructs DMA to write frame IC and header to ext. Buffer
- Per-group policing (RFC2698)
- Multiple stages: Parse / Classify / Distribute → Determine queue ID#
- BMI instructs DMA to transfer frame to external buffer
- Based upon Layer-2 packet size, BMI requests "right sized" buffer to BMan
- BMI streams and allocate internal buffer for incoming frame + IC (Internal Context)
  Calculate raw L4 checksum for parser
- MAC Rx and validate
- Rx




## Egress

- Core Enqueue frame
- BMI allocates internal buffer
- BMI copies frame IC (Internal Context) to internal buffer
- BMI instructs DMA to copy frame from external buffer, calculates TCP/UDP checksum on the fly for later insertion
- BMI checks physical port Tx rate (< predefined rate)
- Inserts checksum and transmits frame to MAC
- Releases internal and external buffers (returned to BMan pool)
- Tx

freescale™

# QorIQ DPAA 1.x

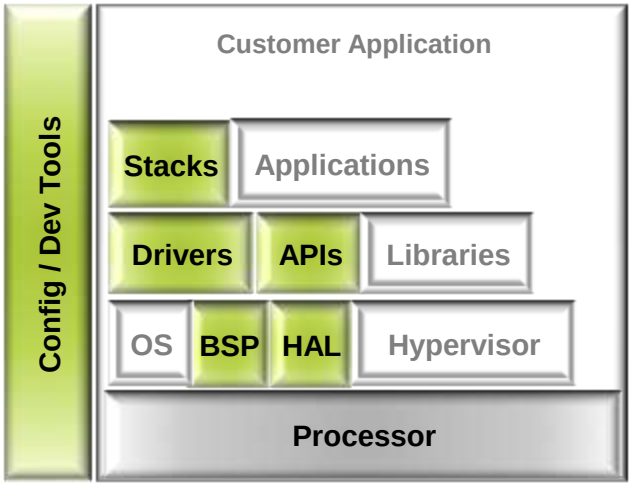
## The Software Packages

# Freescale SDK

- U-boot
- Linux
  - KVM
  - USDPAA (User Space UIO)
- Topaz (HV)

# NetComm Software Solution

- NCSW is a baremetal environment
- NCSW contains drivers that are written in “Agnostic architecture”. I.e. driver may easily be ported to any embedded OS and/or compiler;
  - Useful Porting-Guide for easier integration process.
- Support SMP and non-SMP (i.e. AMP) Systems



# NetComm Software Solution

- Successfully ported, by vendor/in-house to many OS

ENEA
OSE

QNX SOFTWARE SYSTEMS
Neutrino

Green Hills SOFTWARE
Integrity

freescale™
Linux SDK

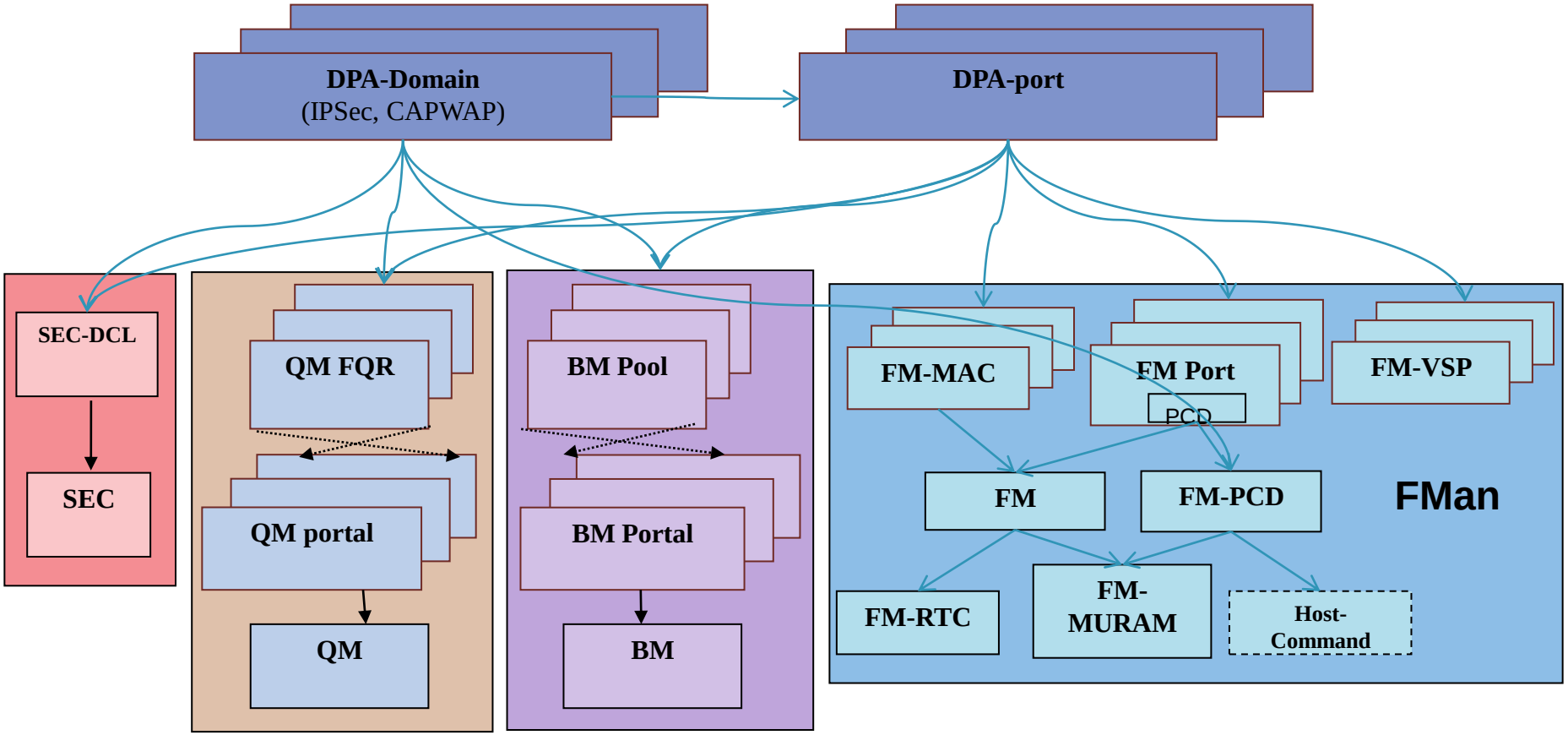
WIND RIVER
VxWorks™
Linux

- Already integrated with many compilers
  - GNU GCC, DIAB, FSL-PPCEABI, GHS
- Provided with a  non-GPL Freescale EULA
- Freescale SDK includes NCSW FMD driver
- NetComm Software Web home: http://www.freescale.com/netcommsw

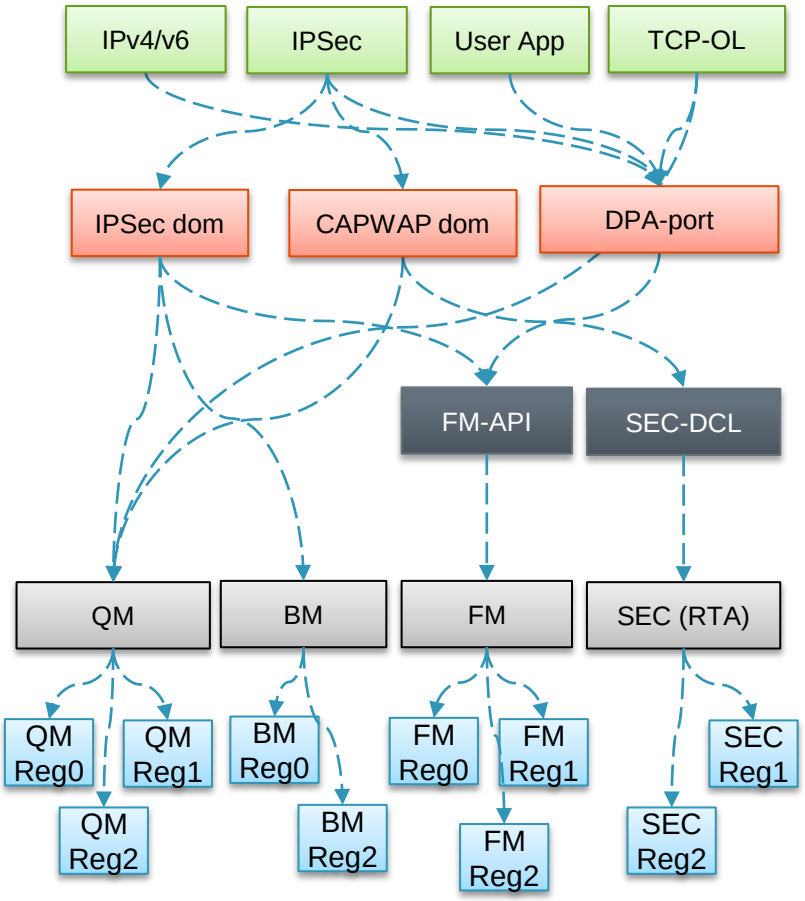
# QorIQ DPAA 1.x

## Software Components

# DPAA1.x - Software Modules



# DPAA1.x - Software Layers




**Layer-4**
*Application Specific*
e.g. ASF

**Layer-3**
*Network Component*

**Layer-2**
*Abstracted Component*
a.k.a API

**Layer-1**
*Direct Component*
a.k.a FLib

**Layer-0**
*HW Access*

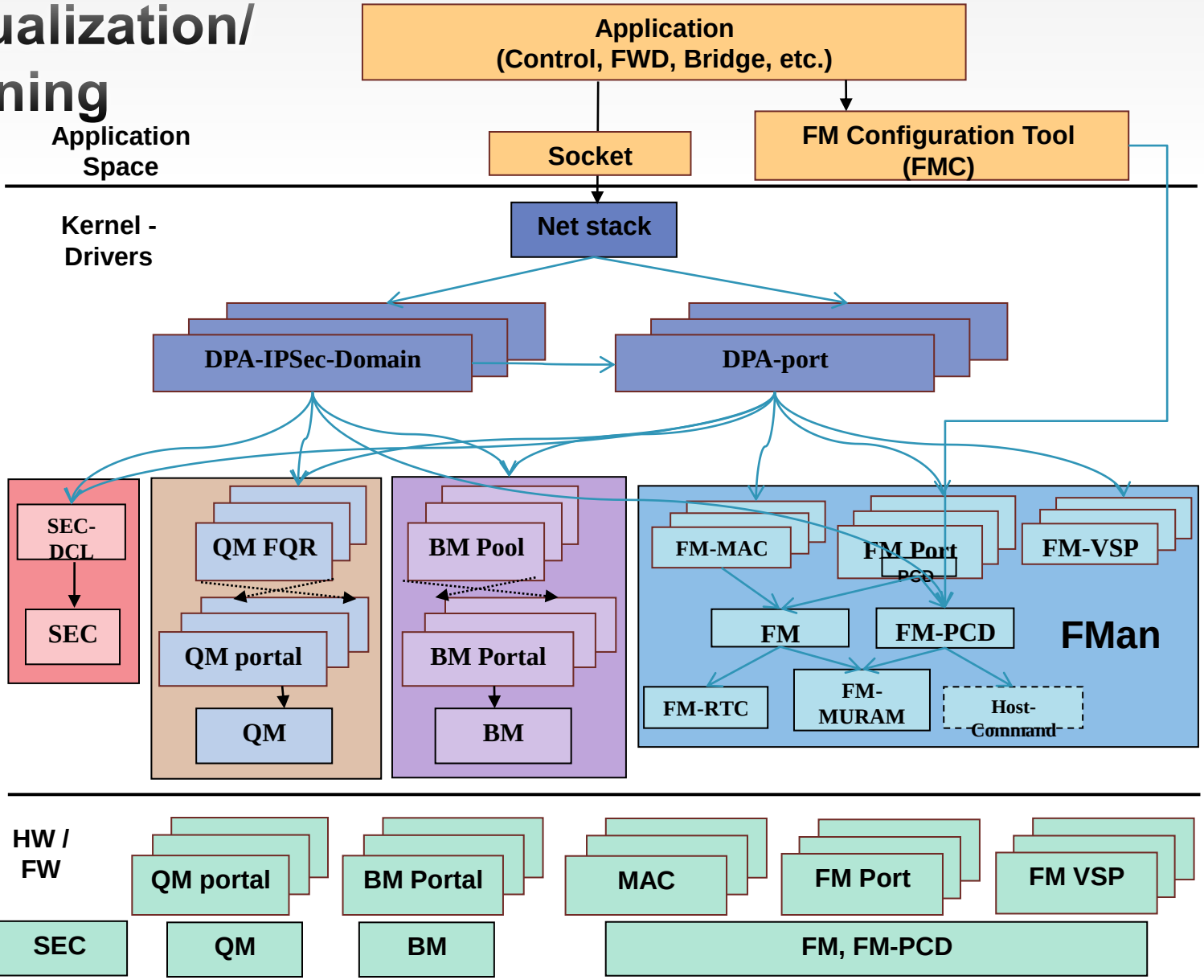
FLIBS

# QorIQ DPAA 1.x

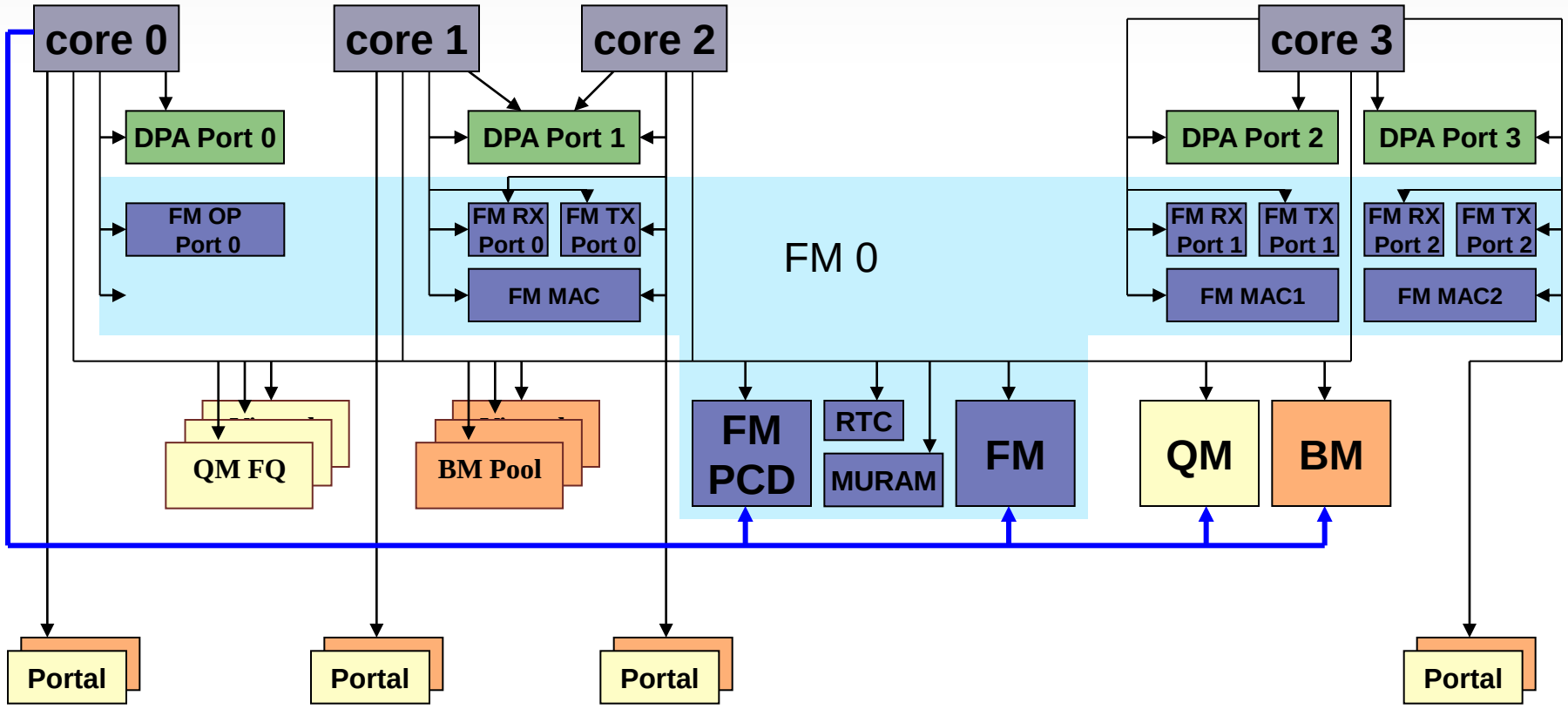
## The Software in a System

# assic SMP environment

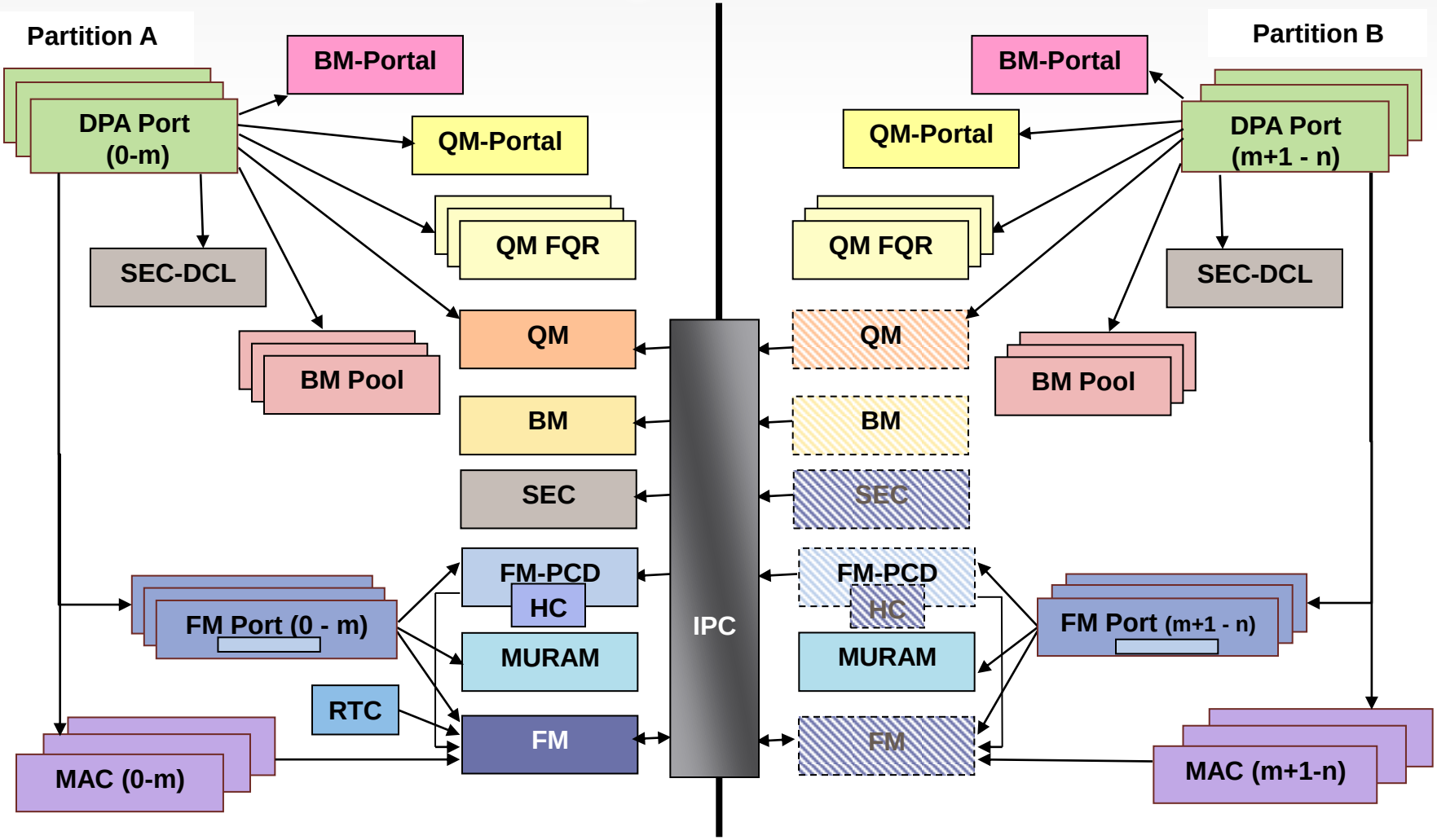
## No Virtualization/ Partitioning



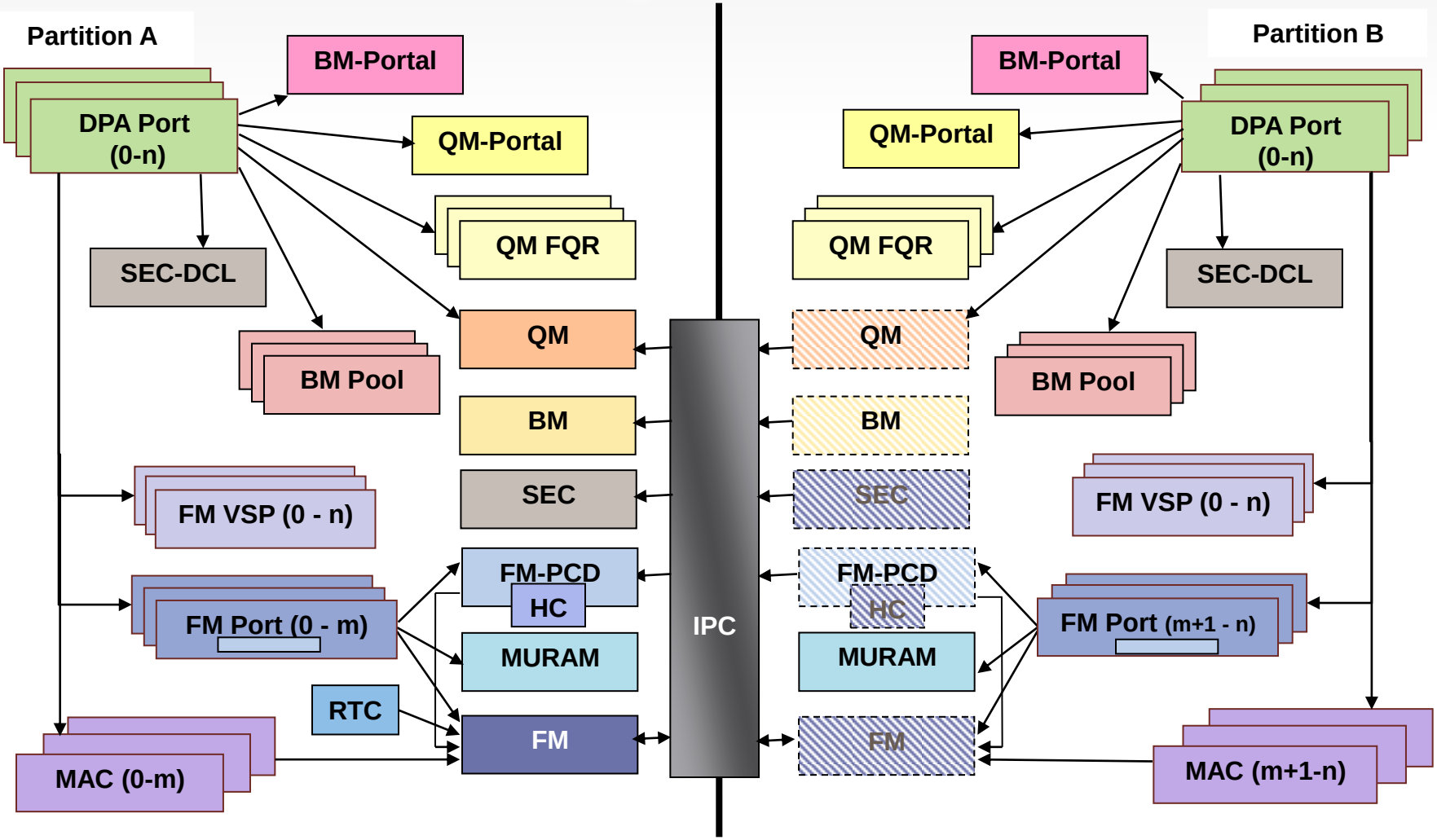
# Multi-core in SMP example (Single Partition)



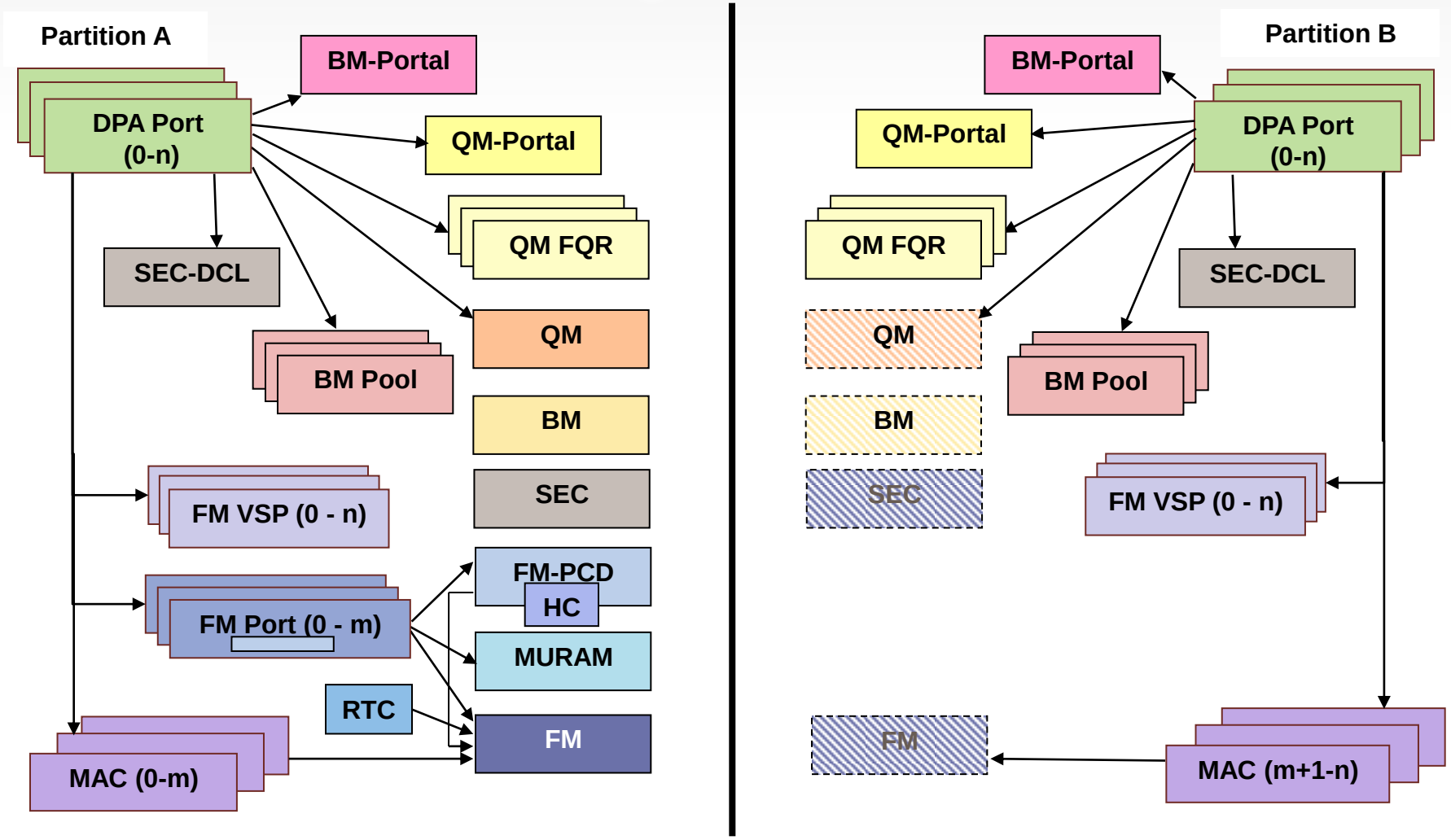
# Multi-core Partitioning with IPC – DPAA1.0



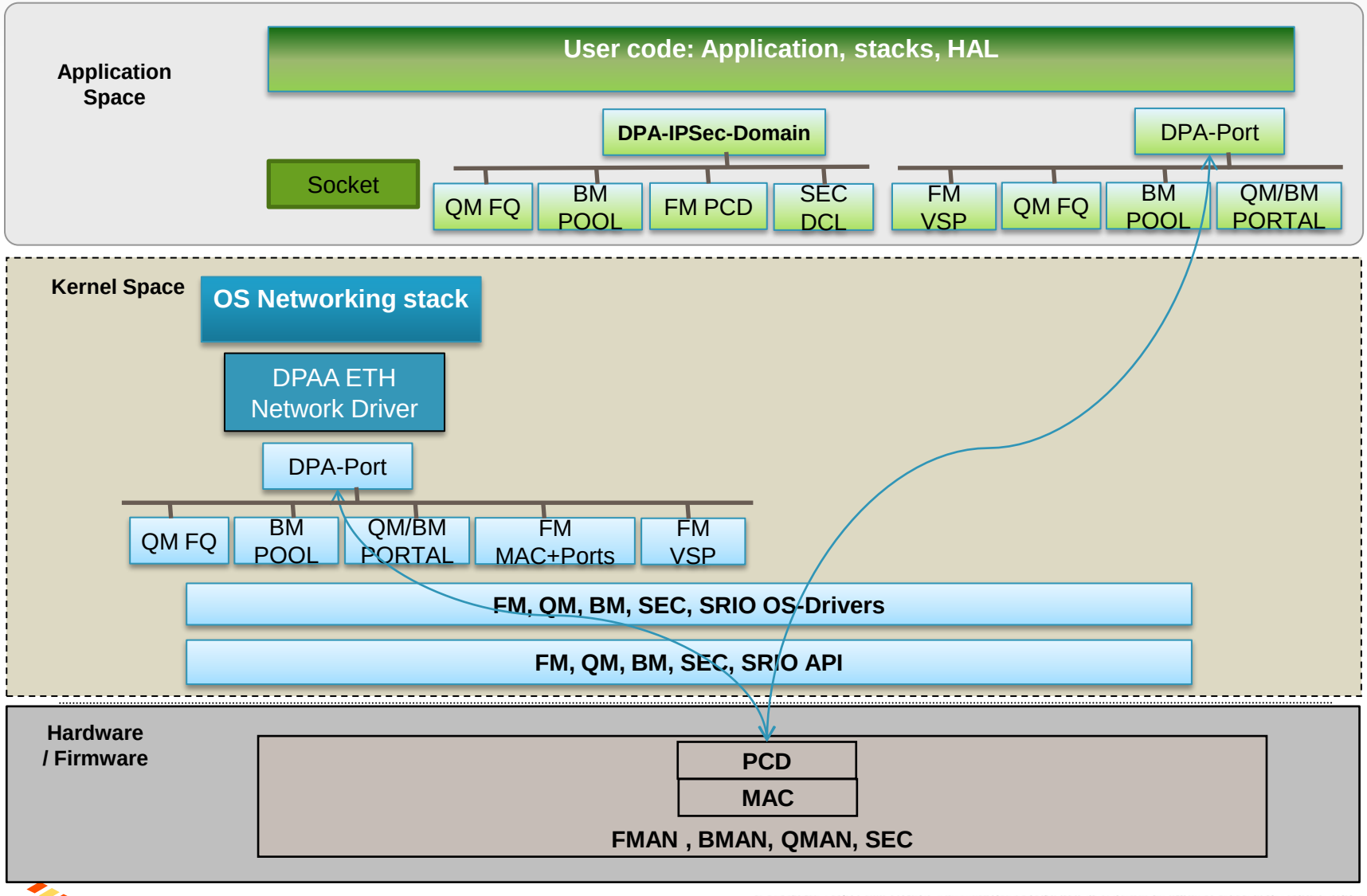
# Multi-core Partitioning with IPC – DPAA1.1



# Multi-core Partitioning without IPC – DPAA1.1



# Virtualization



# QorIQ DPAA 1.x

## “PCD” Graphs &
## PCD Processing Elements

# PCD Processing Elements

- [FMan] Physical ports
- [FMan] “Policy” – based on Classification-plans and parser (and SW-parser for user extensions)
- [FMan] Policer (Coloring)
- [FMan] Distribute
- [FMan FW] Tables – Match (TCAM based), “Indexed”, Hash
- [FMan FW] IP-Reassmebly, IP-Fragmentation
- [FMan FW] CAPWAP-Reassembly, CAPWAP-Fragmentation
- [FMan FW] L2 header encap/decap – ETH, VLAN
- [FMan FW] L3 header encap/decap – IPv4, IPv6 (partially)
- [FMan FW] L4 header encap/decap – UDP

# PCD Processing Elements – cont.

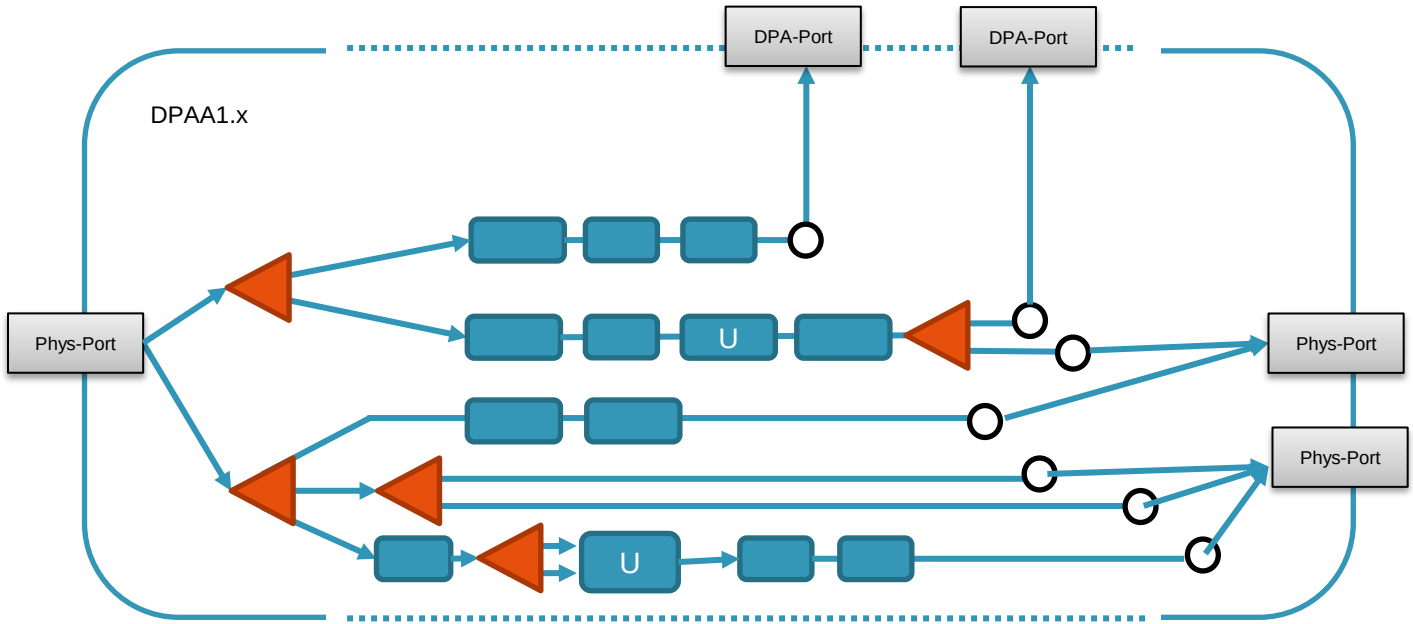
- [FMan FW] Frame-Replication
- [QMan] "Policing" and congestion-management – tail-drop, WRED
- [QMan] Class based Shaping (CEETM)
- [CAMM] Security Elements:
  - IPSec
  - DTLS
  - Kasumi
  - Etc.
- [SW] DPA-Port – consists of Frame-Queue and "Virtual-Storage-Profiles"

# DPAA “PCD” Graph concept example

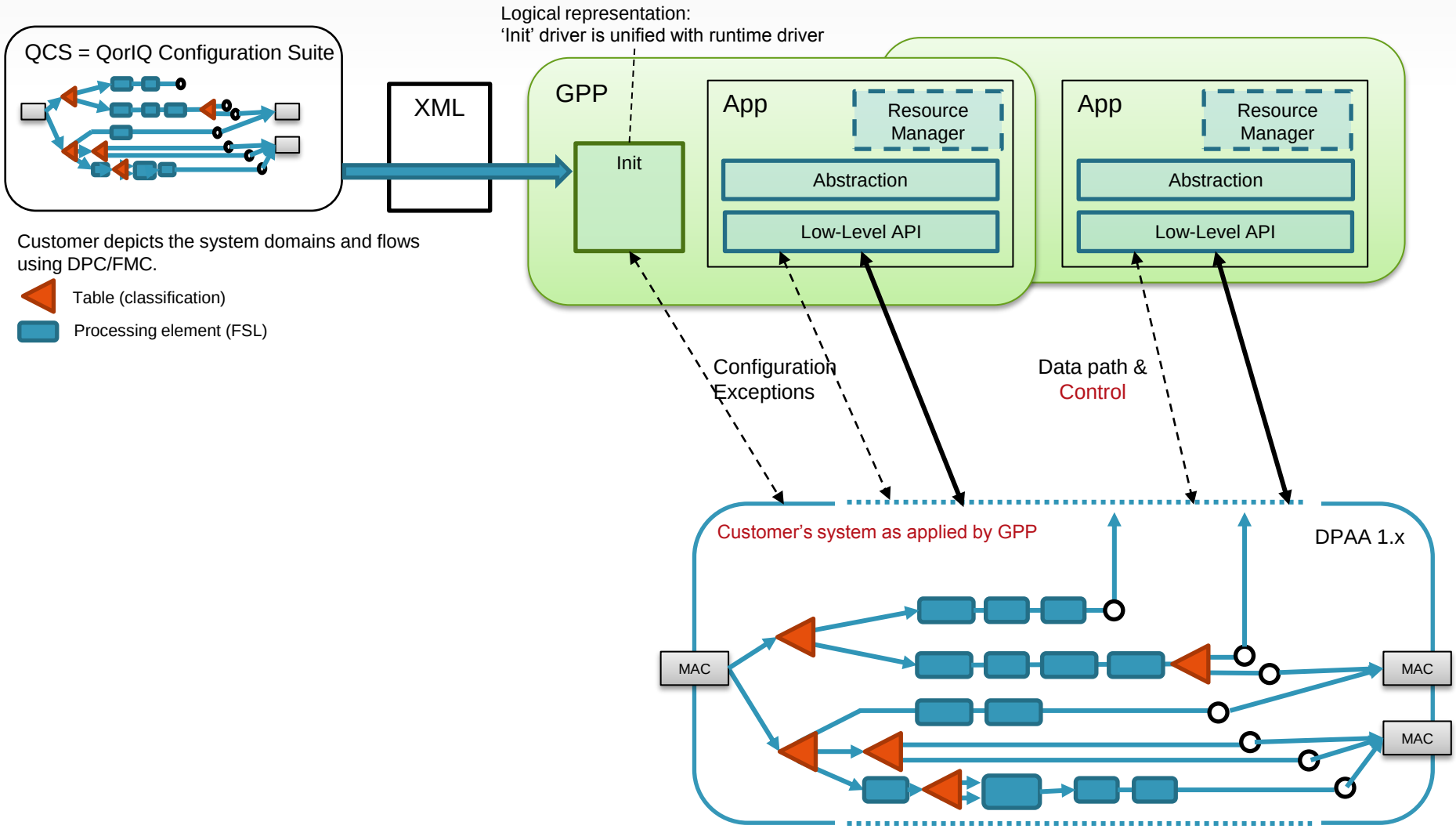
Customer depicts the system domains and flows,

Table (classification)

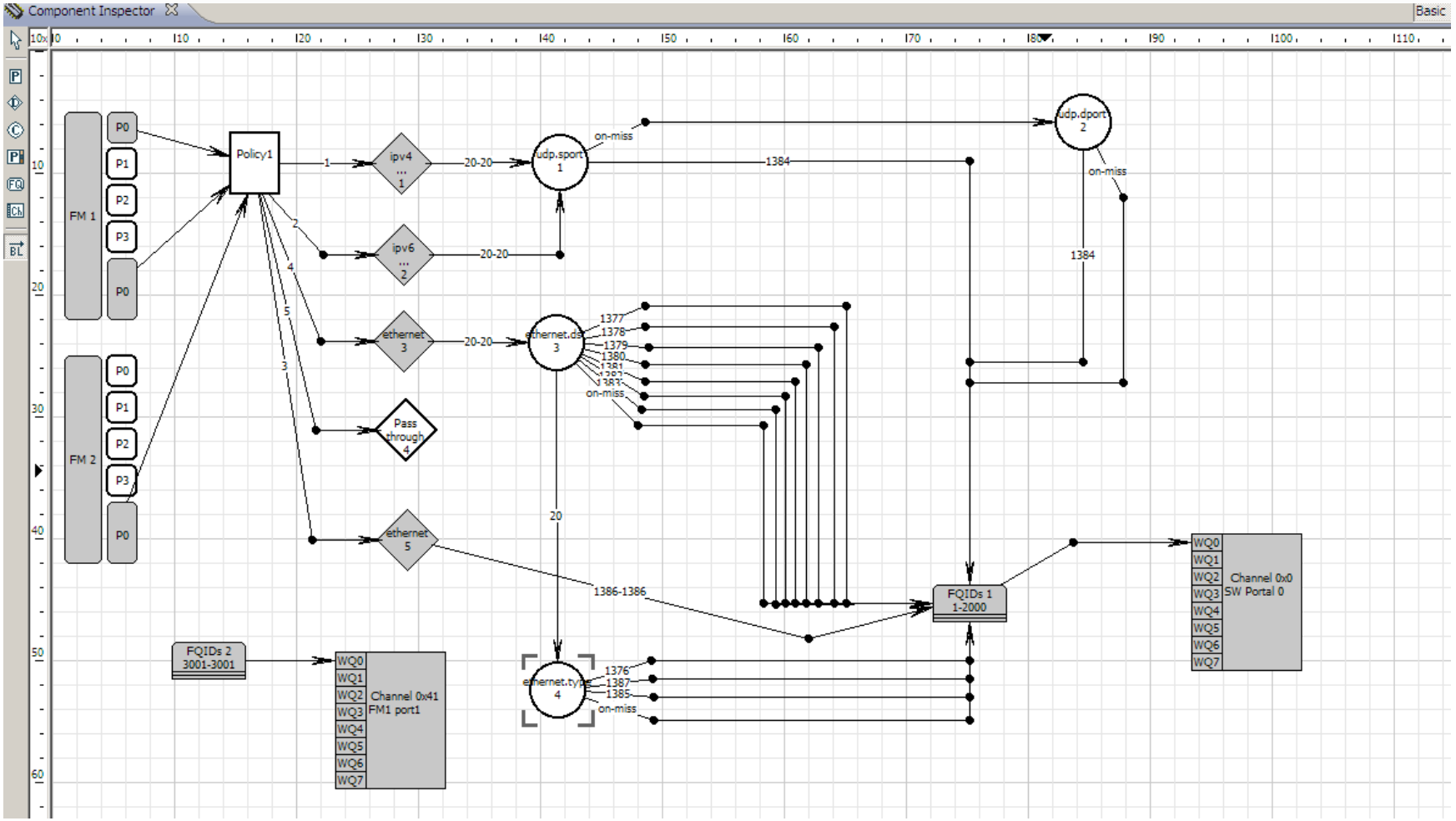
Processing element (FSL / User)



# DPAA 1.x: PCDs in a system



# QorIQ Configuration Suit (QCS) - Example



# QorIQ DPAA 1.x

## Advanced Data Flows & DPAA Domains

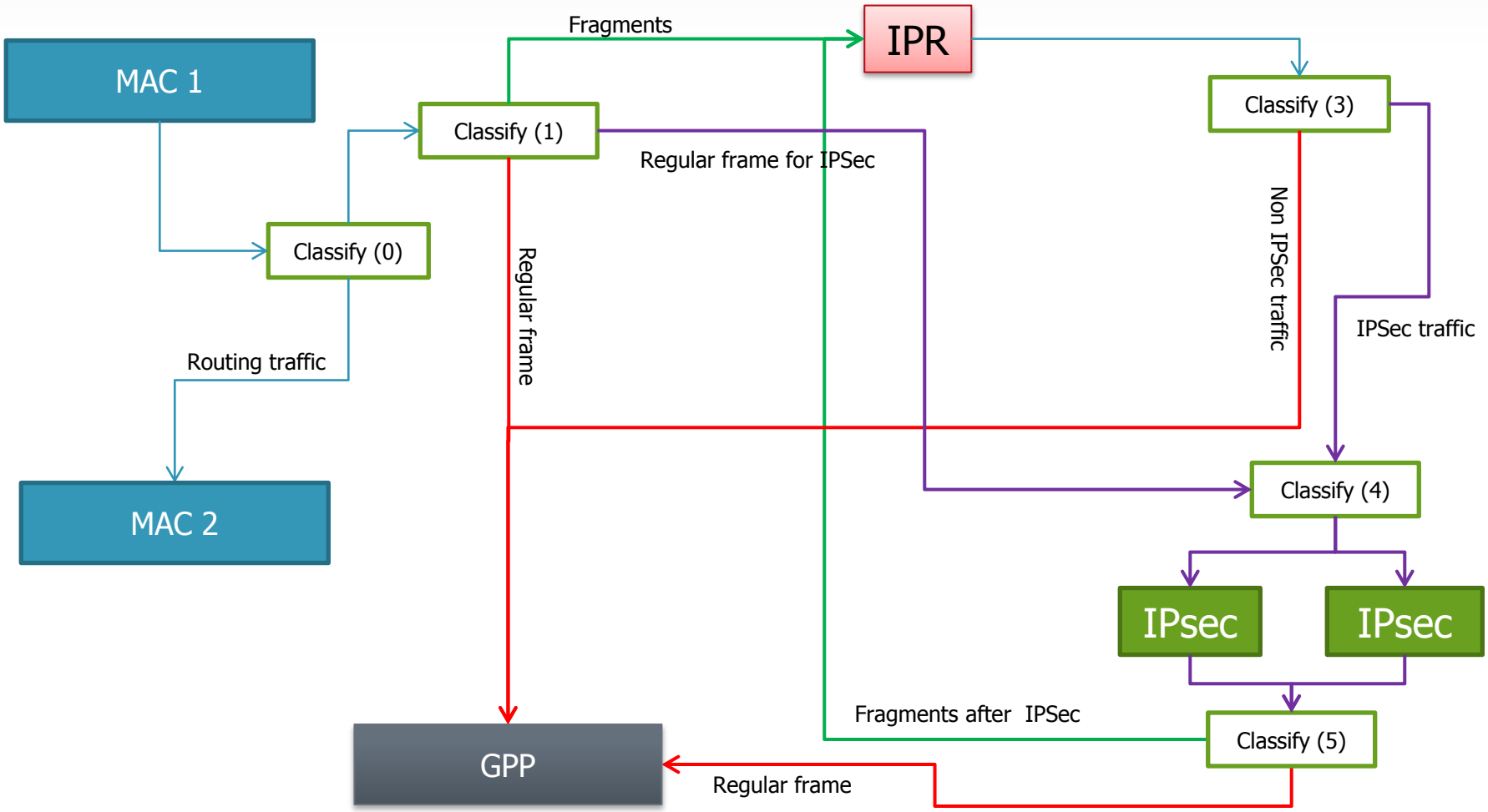
# The Problem (and the solution)

- Questions:
  - What can we do when Processing Elements are “crossing” the boundaries of FMan (i.e. advance offload that need to involve HW such as CAAM or even several “FMan paths”)?
  - What can we do when the offload as dynamic (not only table updates. Examples – adding/removing IPSec tunnels)?

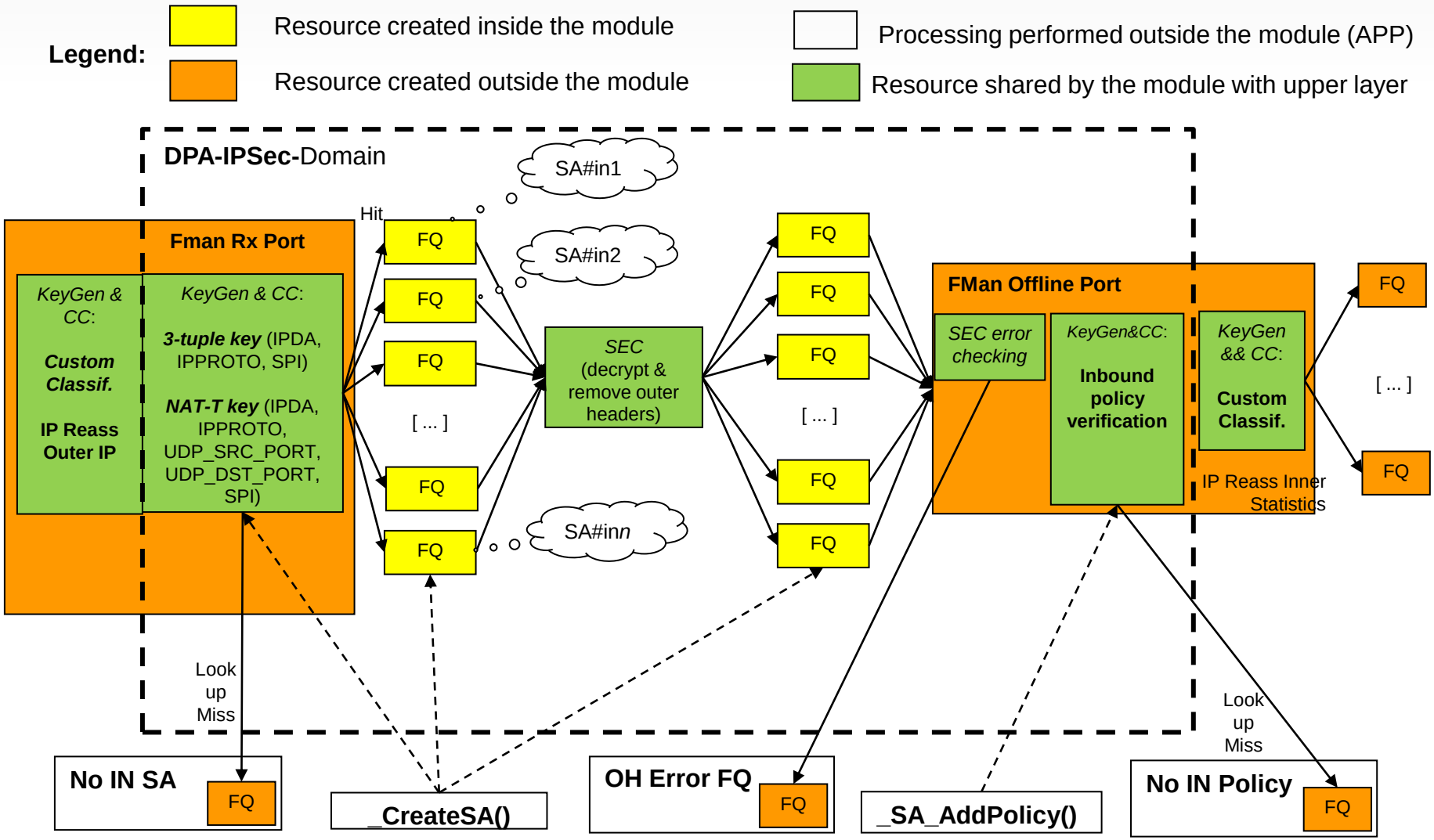
- Answer:

  “DPAA Domains”

# "Real life" IPSec "Inbound" side – Logical Flow (example)



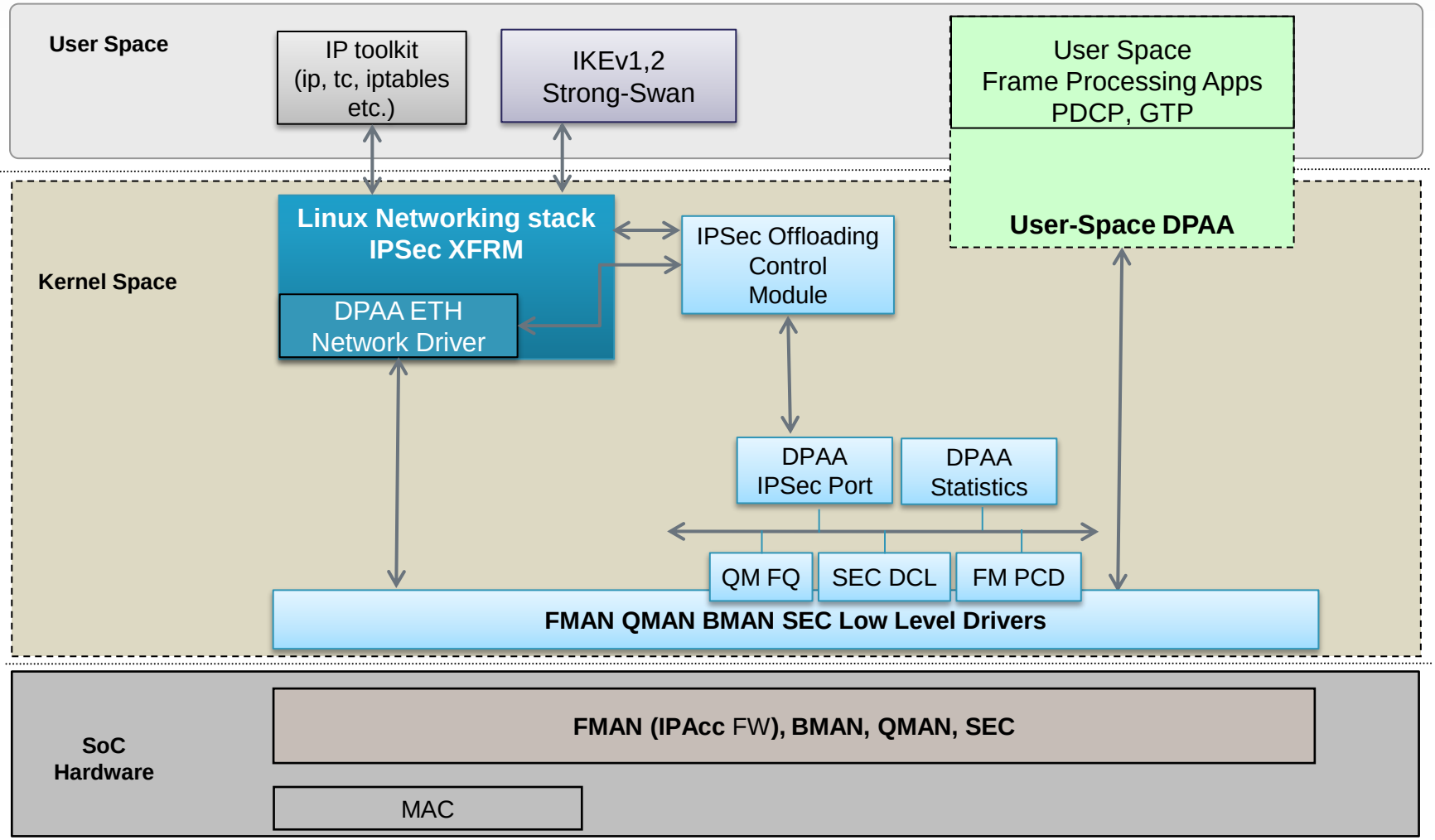
# DPAA IPSec Port - Inbound Flow Processing



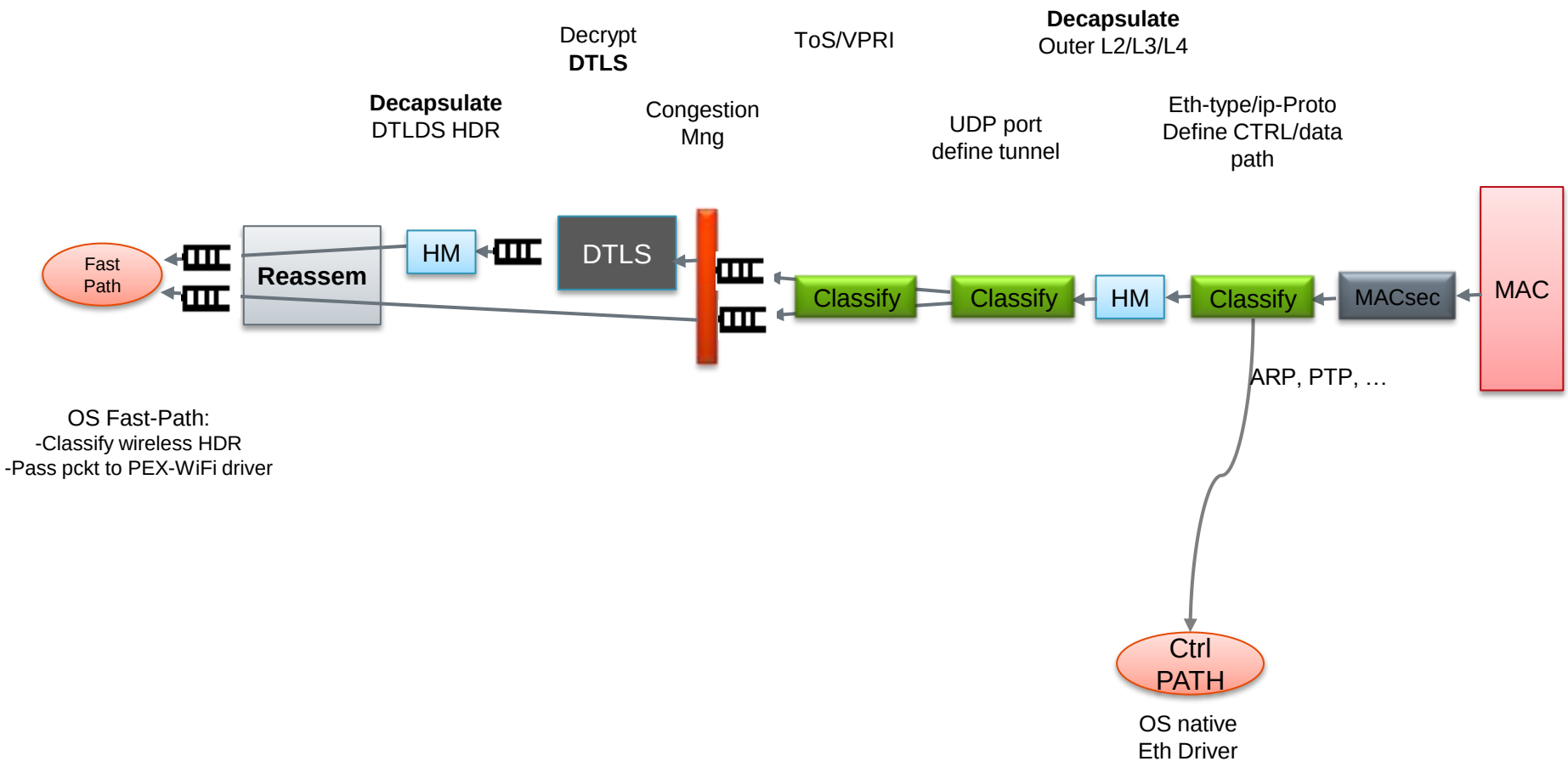
# DPA-IPSec-Domain API

```
t_Handle DPAA_IPSEC_DOMAIN_Config(t_DpaaIPSecDomainPara  *p_DpaaIPSecDomainParams);
t_Error DPAA_IPSEC_DOMAIN_Init(t_Handle h_DpaaIPSecDomain);

t_Handle DPAA_IPSEC_DOMAIN_CreateSA(t_Handle                    h_DpaaIPSecDomain,
                                    t_DpaaIPSecDomainSAParams   *p_SaParams);
t_Error DPAA_IPSEC_DOMAIN_DeleteSA(t_Handle h_Sa);

t_Error DPAA_IPSEC_DOMAIN_SA_AddPolicy(t_Handle                     h_Sa,
                                       t_DpaaIPSecDomainPolicyParams *p_PolicyParams,
                                       uint32_t                      *p_PolicyId);
t_Error DPAA_IPSEC_DOMAIN_SA_RemovePolicy(t_Handle  h_Sa,
                                          uint32_t   policyId);
```

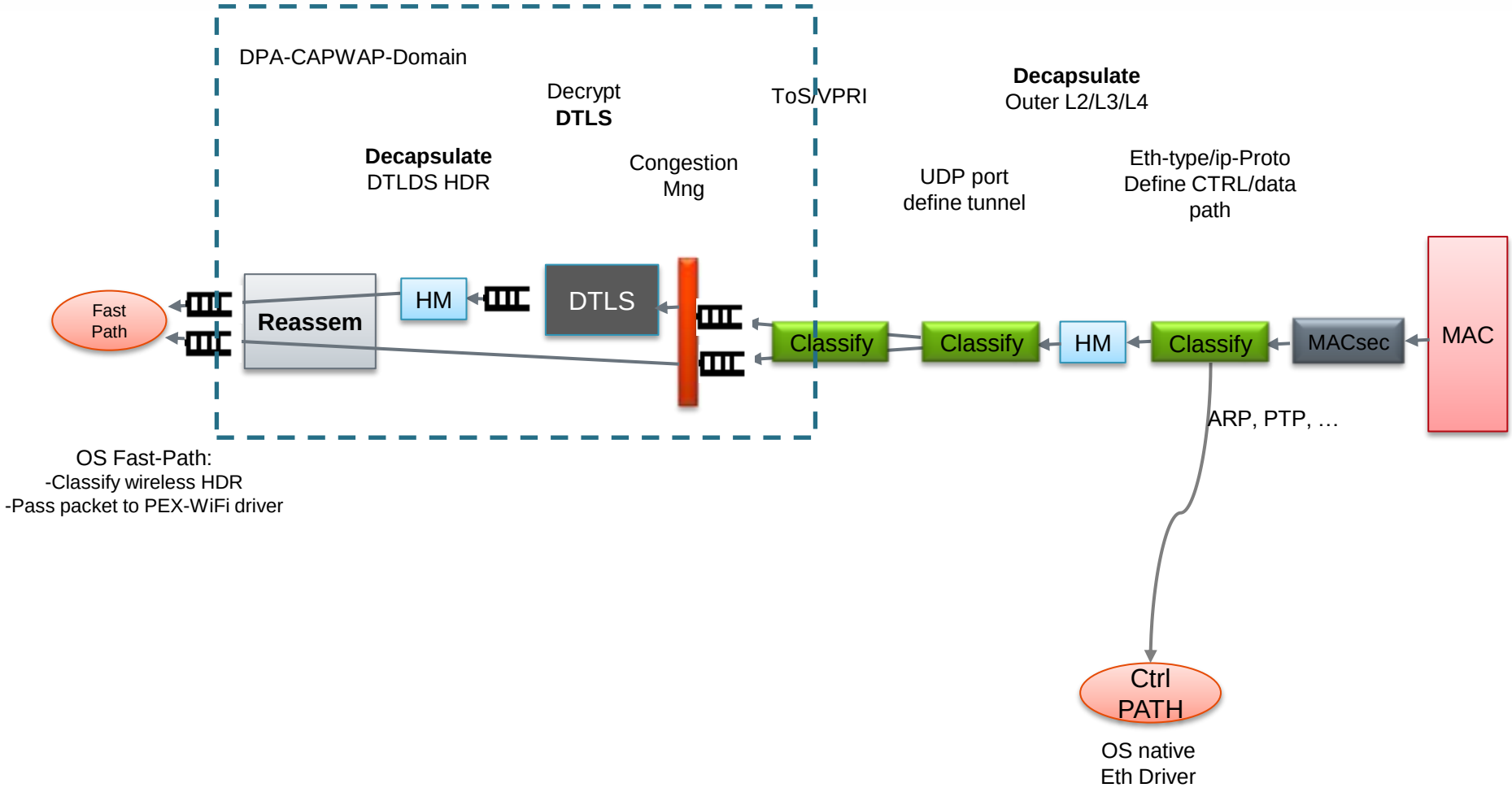
# DPA-IPSec-Domain in Linux – Integration example



# "Real life" CAPWAP-DTLS Downlink side – Logical Flow (example)



# CAPWAP-DTLS Downlink side – Domain boundaries



# DPA-CAPWAP-Domain API

*t_Handle DPAA_CAPWAP_DOMAIN_Config(t_DpaaCAPWAPDomainParams *p_DpaaCAPWAPDomainParams);*

*t_Error DPAA_CAPWAP_DOMAIN_Init(t_Handle h_DpaaCAPWAPDomain);*

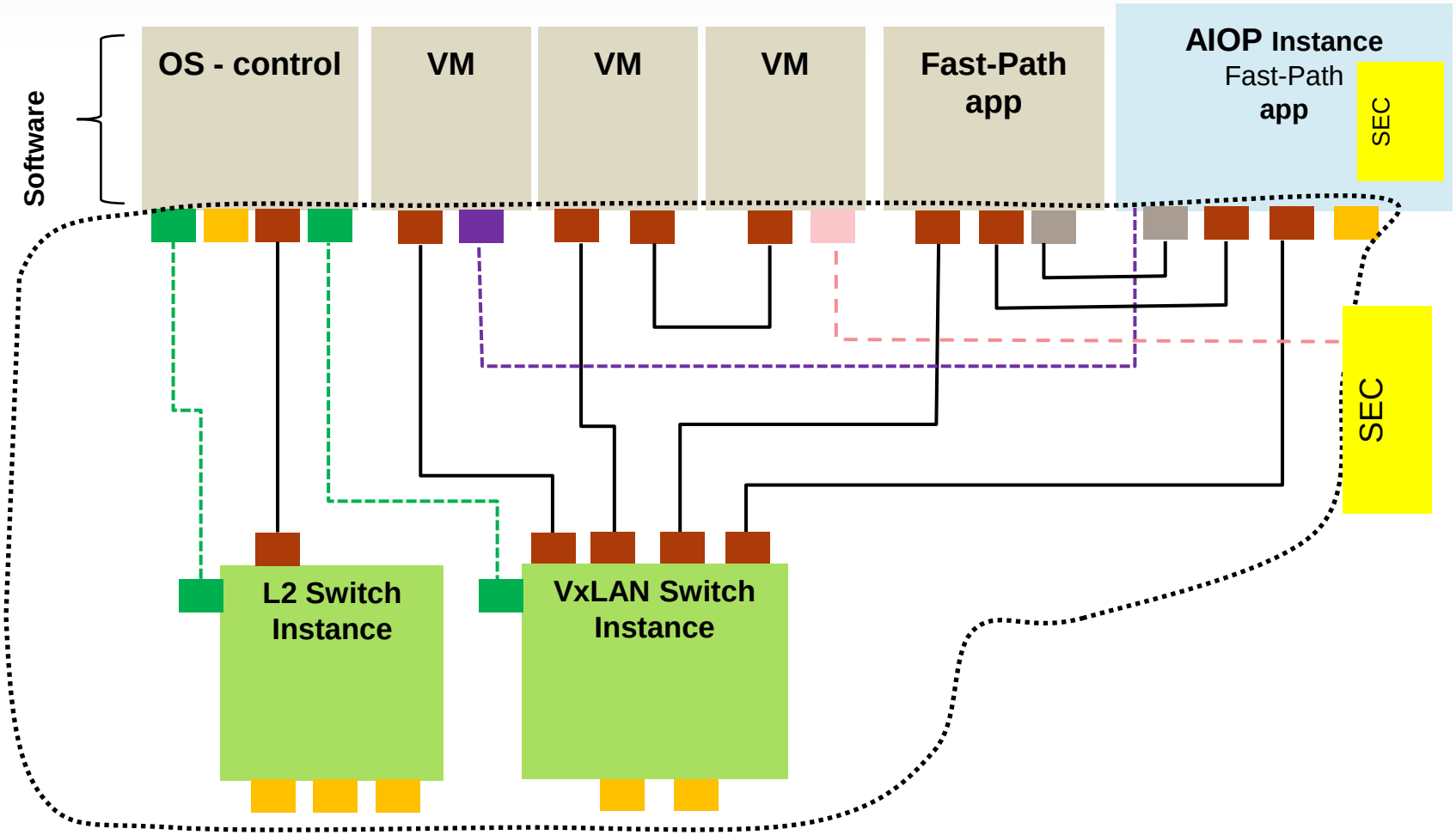
*t_Handle DPAA_CAPWAP_DOMAIN_CreateTunnel(t_Handle h_DpaaCAPWAPDomain,*
*t_DpaaCAPWAPDomainTunnelParams *p_TunnelParams);*

*t_Error DPAA_CAPWAP_DOMAIN_DeleteTunnel(t_Handle h_Tunnel);*

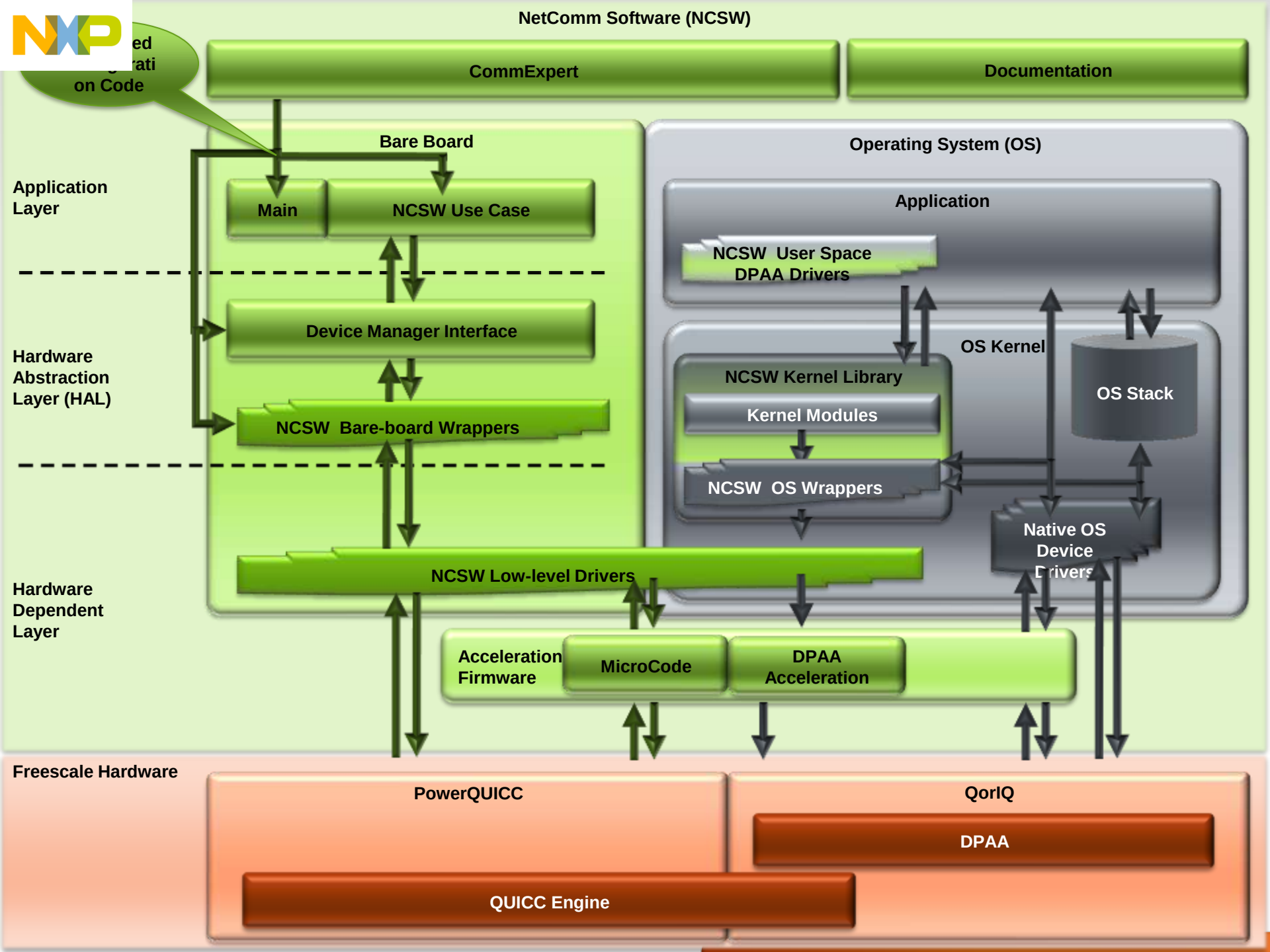
# QorIQ DPAA 1.x

## Looking into the future – LayerScape

# LayerScape – High-Level Logical view




freescale™


NetComm Software (NCSW)
ed
rati
on Code
CommExpert
Documentation
Bare Board
Operating System (OS)
Application Layer
Main
NCSW Use Case
Application
NCSW User Space DPAA Drivers
Device Manager Interface
Hardware Abstraction Layer (HAL)
OS Kernel
NCSW Kernel Library
Kernel Modules
OS Stack
NCSW Bare-board Wrappers
NCSW OS Wrappers
Native OS Device Drivers
NCSW Low-level Drivers
Hardware Dependent Layer
Acceleration Firmware
MicroCode
DPAA Acceleration
Freescale Hardware
PowerQUICC
QorIQ
DPAA
QUICC Engine

# Custom Protocols Support

- Hardware parser permits to process known protocols with no additional configuration

- For new and custom protocols, software parser can be involved.
  - Here goes complete example of L2TPv3 Protocol description used for further packet classification and distribution (NetPDL is used as the input language):

# Example: Description of L2TPv3 Protocol

```
<netpdl name="L2TPv3" description="L2TPv3 Protocol definition">
    <protocol name="L2TPv3" longname="L2TPv3 Protocol" prevproto="ipv4">
        <format>
            <fields>
                <field type="fixed" name="sessionID" longname="SessionID"          size="4"/>
                <field type="fixed" name="header"    longname="Sub-layer header" size="4"/>
            </fields>
        </format>

        <execute-code>
            <before>
                <!-- Check what the next protocol is -->
                <switch expr="ipv4.nextp">
                    <!-- The "Protocol" in the IP header for L2TPv3 is 115 -->
                    <case value="115">
                        <!-- Store the shim header position -->
                        <assign-variable name="$shimoffset_1" value="$nxtHdrOffset"/>
                    </case>
                    <default>
                        <!-- For other protocol values - just continue normally -->
                        <action type="exit" nextproto="after_ip"/>
                    </default>
                </switch>
            </before>
            <after>
                <!-- Move nxtHdrOffset to a position after L2TPv3 -->
                 <assign-variable name="$nxtHdrOffset" value="$shimoffset_1 + 8"/>
                <!-- Just confirm existance of shim1 protocol     -->
                <action type="exit" confirmcustom="shim1" advance="yes" nextproto="end_parse"/>
            </after>
        </execute-code>
    </protocol>
</netpdl>
```


freescale™