Logical Abstraction and Resource Management Using the **Layerscape Management Complex**

EUF-NET-T0969

Nir Erez | Software R&D

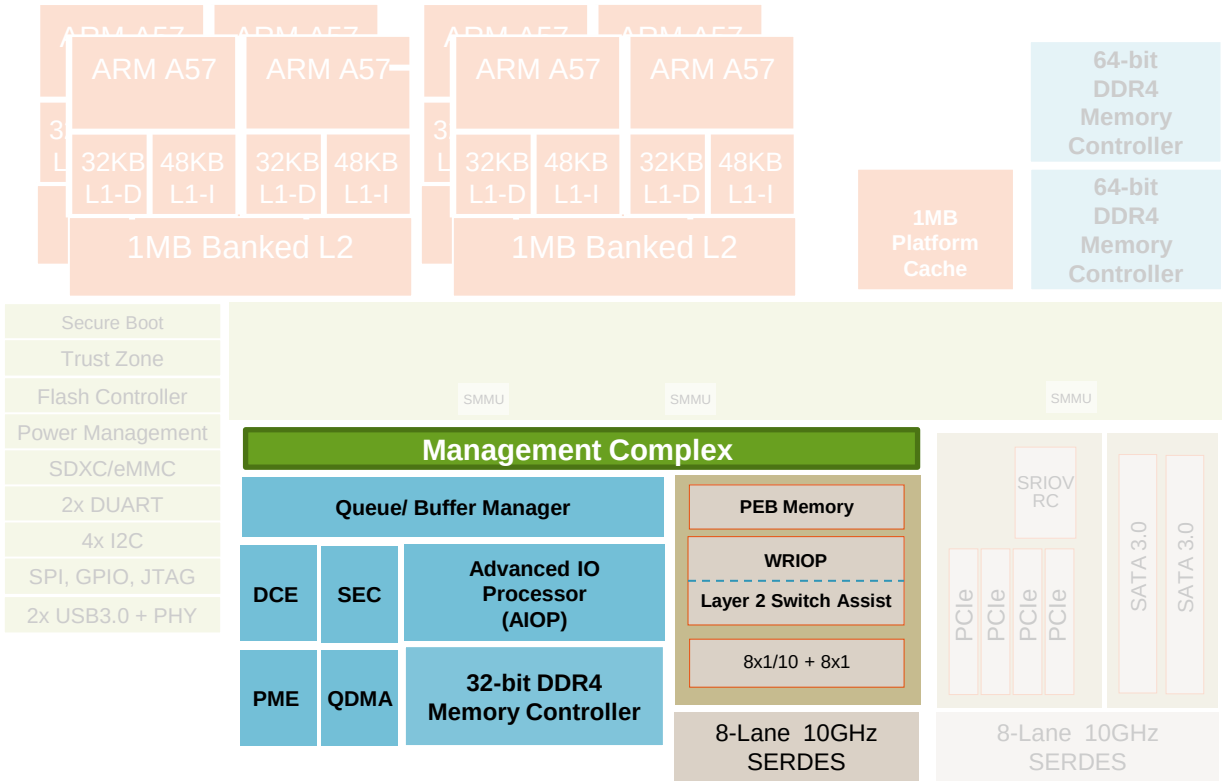
J A N . 2 0 1 5

External Use

Agenda

- Why Management Complex?
- Creating Network-on-Chip
- Resource Management
- MC Logical Objects
- Data Path Network Interface (DPNI)
- Data Path Switch (DPSW)

LS2085A Platform



General Purpose Processing

- 8x ARM® A57 CPUs, 64 b, 2.0 GHz
  - 1 MB L2 cache
- HW L1 & L2 Prefetch Engines
- Neon SIMD in all CPUs
- 1 MB L3 platform cache w/ECC
- 4 MB Coherent Cache
- 2x64 b DDR4 up to 2.4GT/s

Express Packet IO

- Supports1x8, 4x4, 4x2, 4x1 PCIe Gen3 controllers
  - SR-IOV support, Root Complex
- 2 x SATA 3.0, 2 x USB 3.0 with PHY

Accelerated Packet Processing

- 20 Gbps SEC - crypto acceleration
- 10 Gbps Pattern Match/RegEx
- 20 Gbps Data Compression Engine

Datapath Acceleration

- **SEC** – crypto acceleration
- **DCE** – Data Compression Engine
- **PME** – Pattern Matching Engine
- **QDMA** – Queue-enabled DMA Engine
- **L2 Switching** -- via Datapath Acceleration Hardware
- **Management Complex** – Configuration and Control Abstraction

Network IO

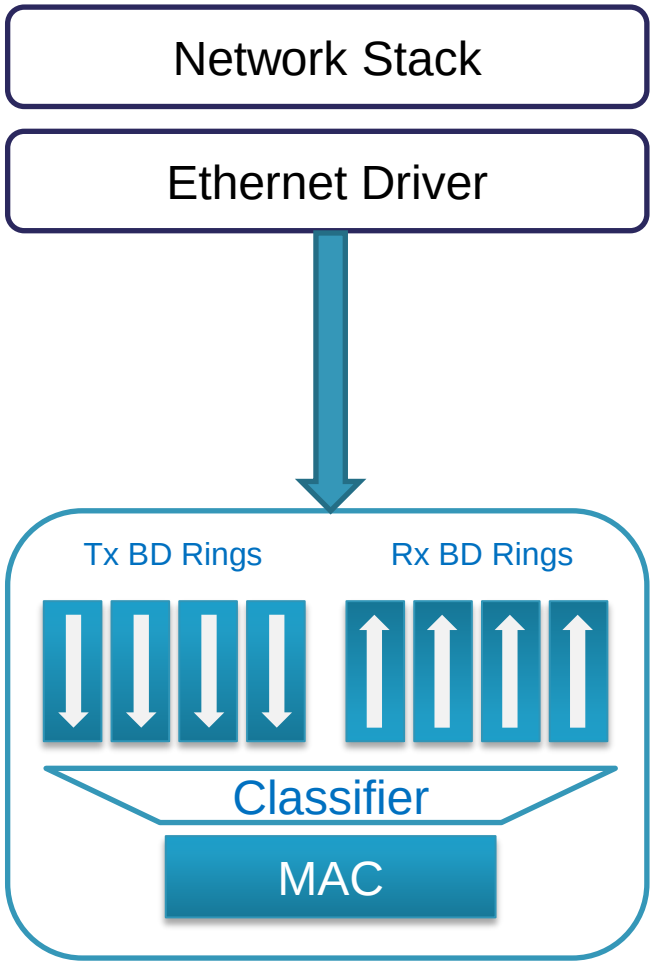
- Wire Rate IO Processor:
  - 8x1/10GbE + 8x1G
  - XAUI/XFI/KR and SGMII
  - MACSec on up to 4x 1/10GbE
  - Layer 2 Switch Assist

…Why Management Complex?

Legacy Ethernet Controller

… translates between network stack's standard features and HW implementation.

… owns all hardware resources needed to operate the Ethernet device.

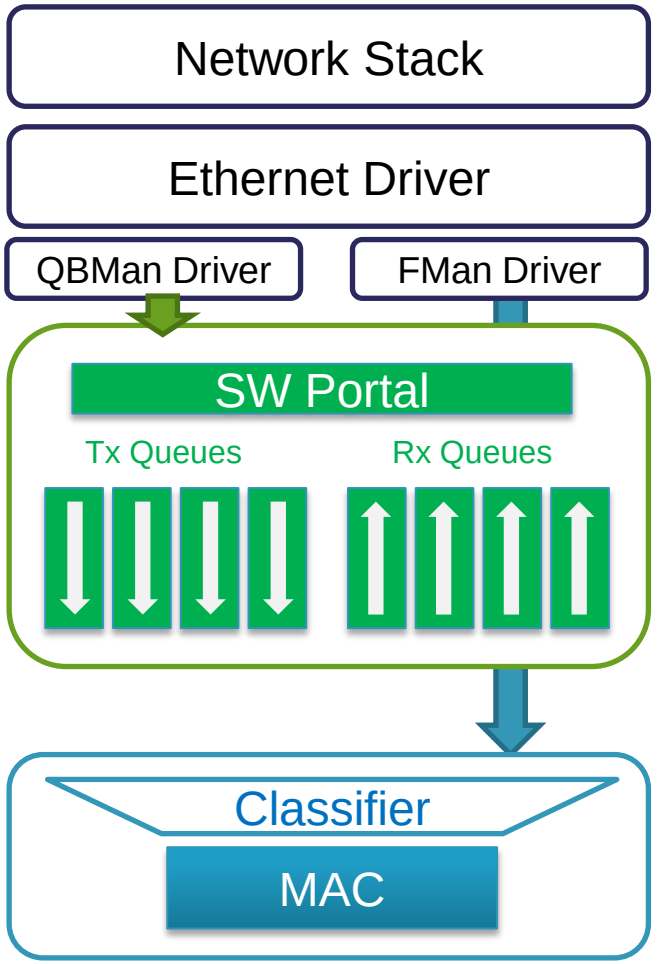


All functions are in a single HW block.

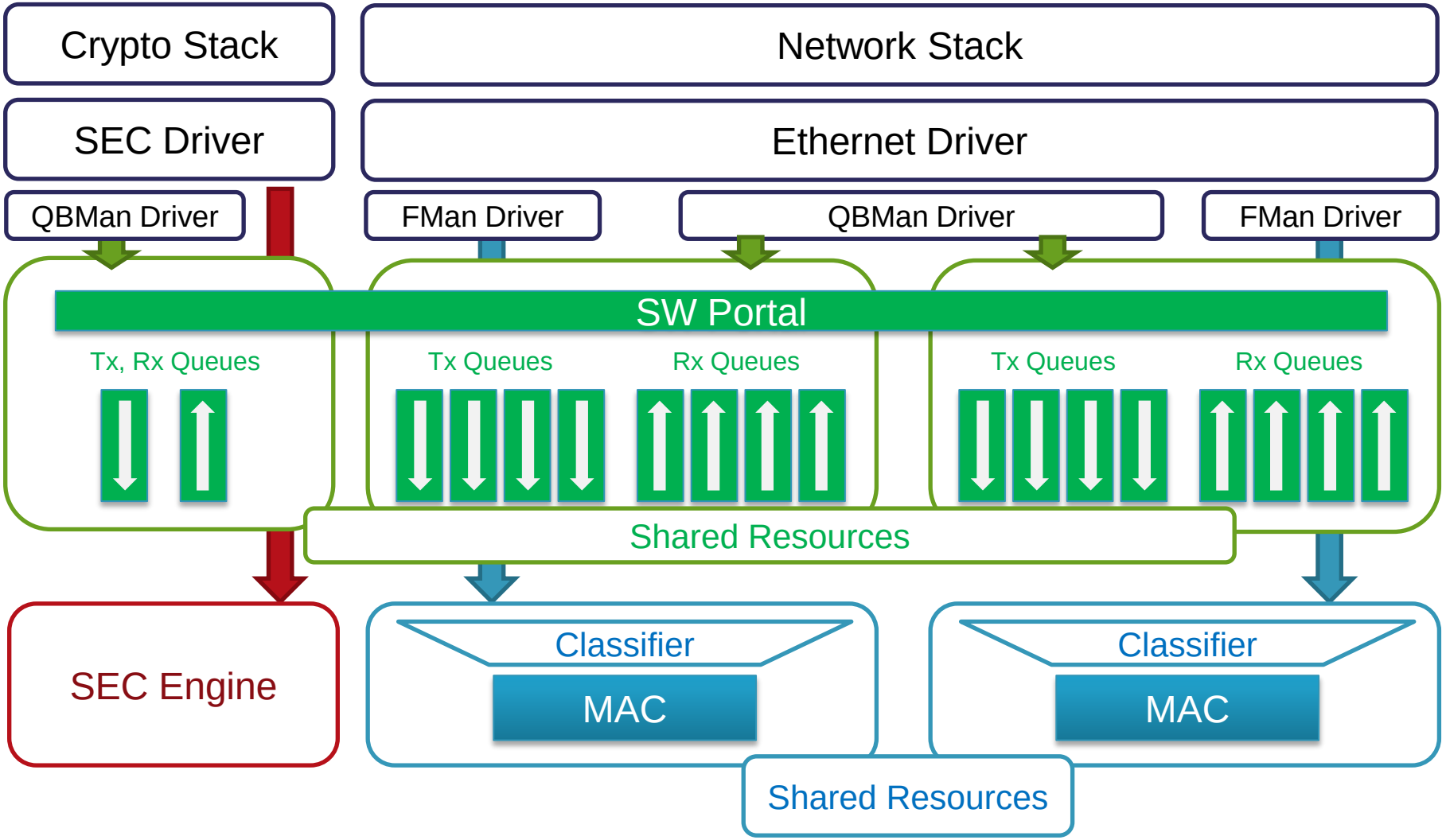
Configuration is mostly independent of other blocks.

DPAA 1.x Ethernet Controller

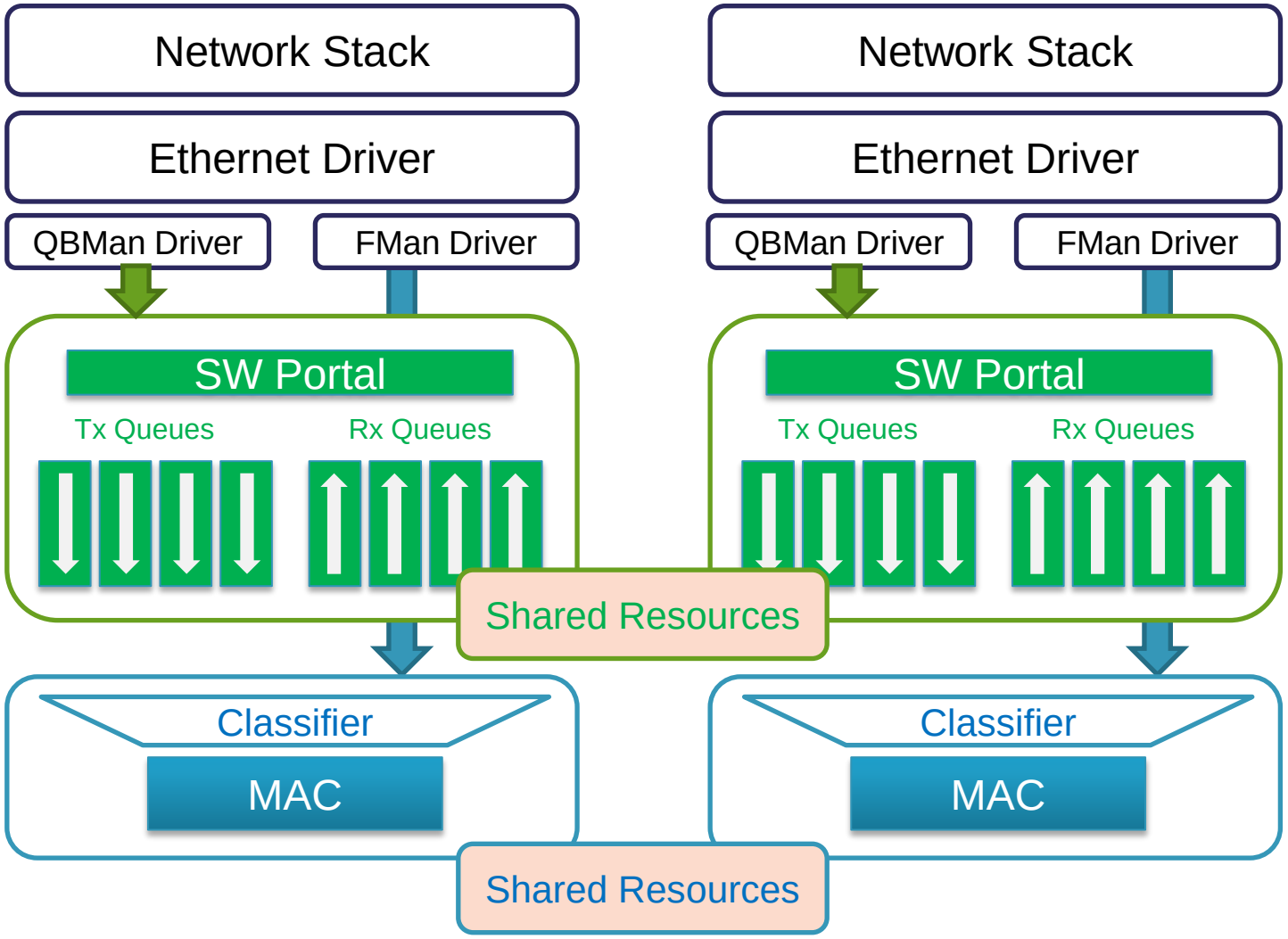
- The Ethernet driver does not own all hardware resources needed to operate the device.
- QBMan resources may serve also other drivers or instances
- Ethernet Controller functions are achieved by multiple HW blocks, and configuration has several dependencies.



DPAA 1.x Ethernet and Crypto Functions (SMP System)

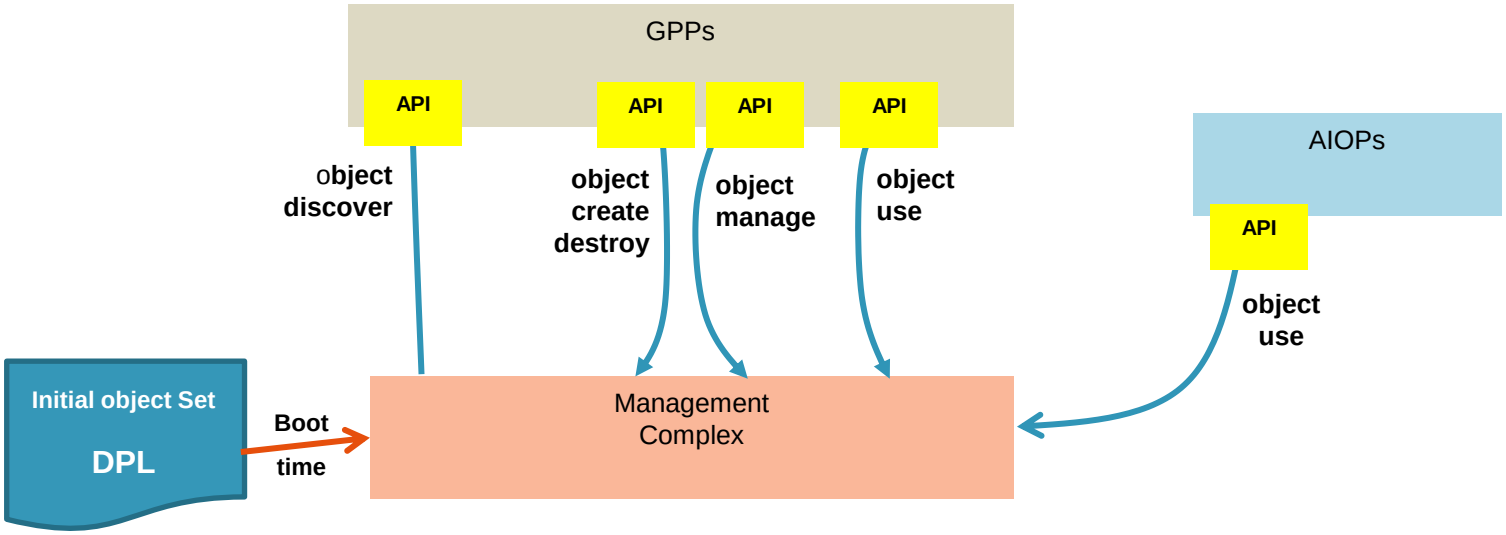


DPAA 1.x Ethernet Controllers (Partitioned System)



Management Complex (MC) – General Concept

- The Management Complex provides **Freescale-owned abstraction and control firmware**.
- MC exports **software-defined and standard-oriented interfaces** to GPP and AIOP software, and thus hides configuration complexity from customers.
- **MC is a trusted entity and only executes Freescale-supplied trusted firmware.**
  It is isolated from the rest of the SoC so it cannot be compromised by malicious or buggy software running on the GPPs.



Management Complex Roles

- **DPAA boot and global initialization**
  - Global initialization of DPAA hardware blocks (QBMan, WRIOP, AIOP, SEC, etc.)
- **Configuration and abstraction of logical objects** – MC firmware allocates the right set of low-level resources and configures them to compose a logical object:
  - Network interfaces (basic or high-function interfaces)
  - L2 switches and Demux objects (MAC partitioning, VEB/VEPA)
  - Link aggregation groups
  - Accelerator interfaces (SEC, DCE, PME, QDMA)
  - Inter-Partition Communication interfaces (GPP ↔ AIOP, GPP ↔ GPP)
- **DPAA objects discovery per software context**
- **DPAA resource management**
  - Allocation, tracking and recovery in fault scenarios
- **Support DPAA hardware virtualization**

Goal: Easy-to-Use Logical Objects

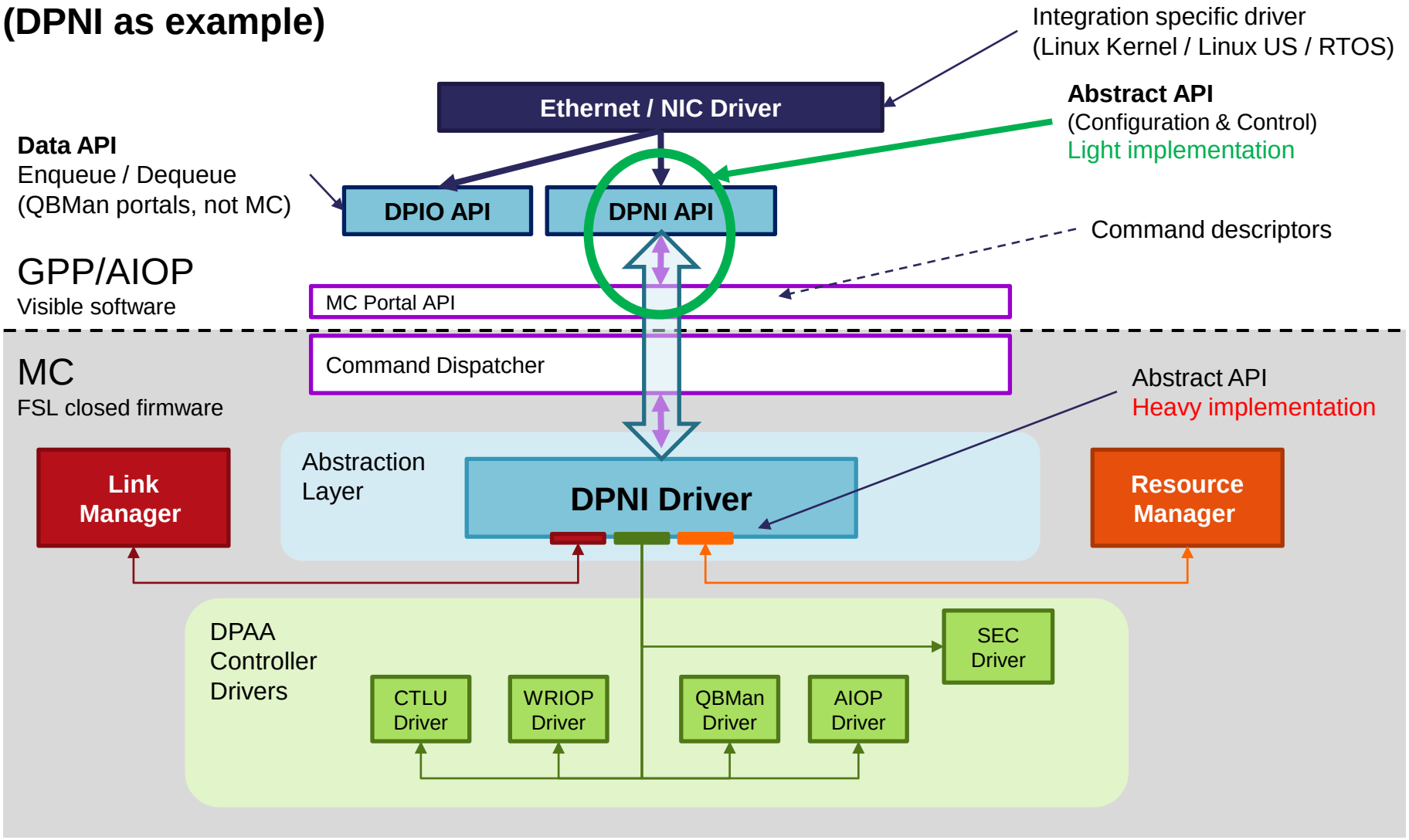
Lack of virtualization

- Centralized resource piles
- Sharing needed but complex
- Requires Hypervisor or IPC

Complex sequences

- Driver dependencies
- Resource cleanup
- Performance tuning

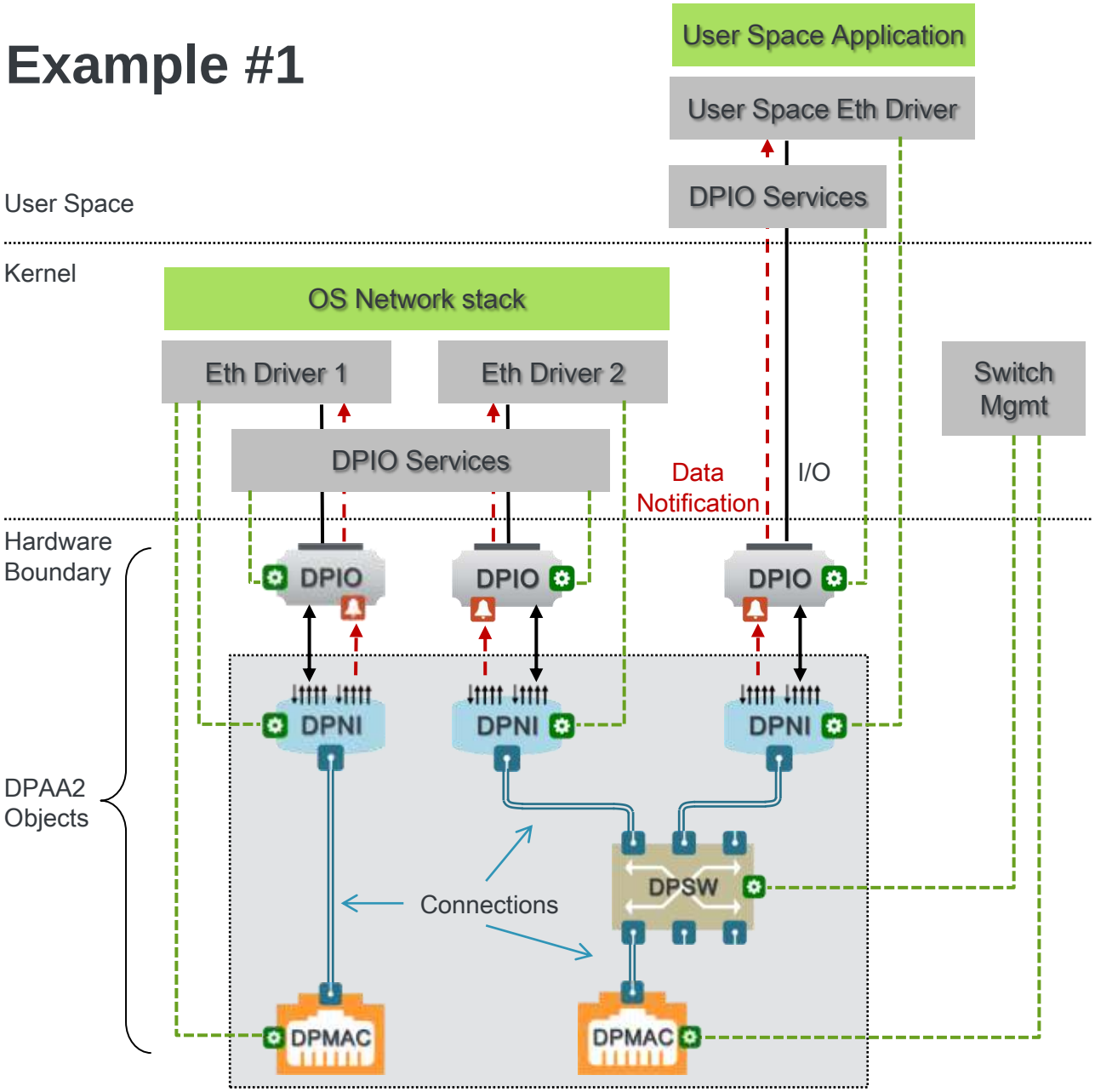
MC Ease of Use
(DPNI as example)



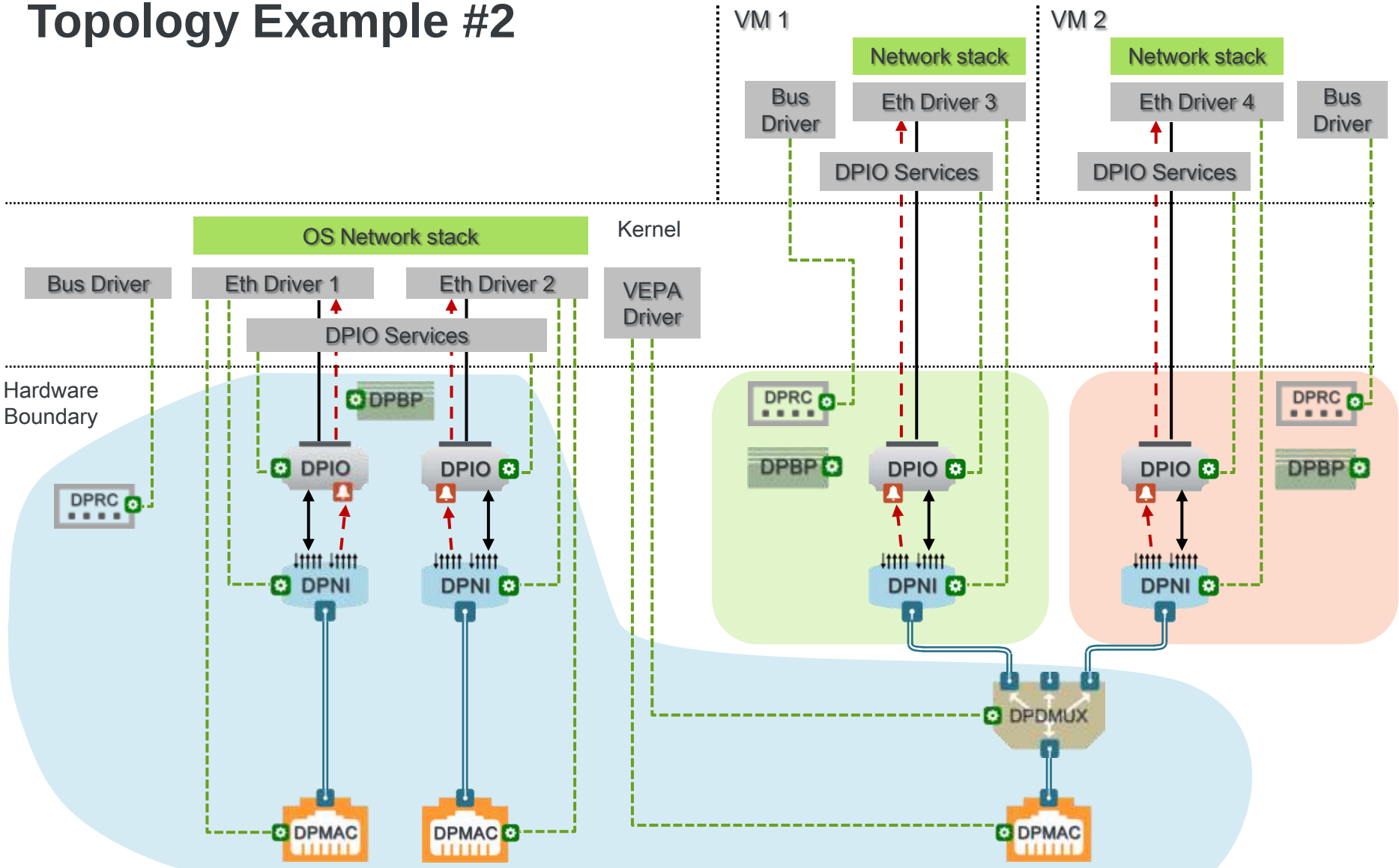
Create Your Network-on-Chip

As easy as 1-2-3

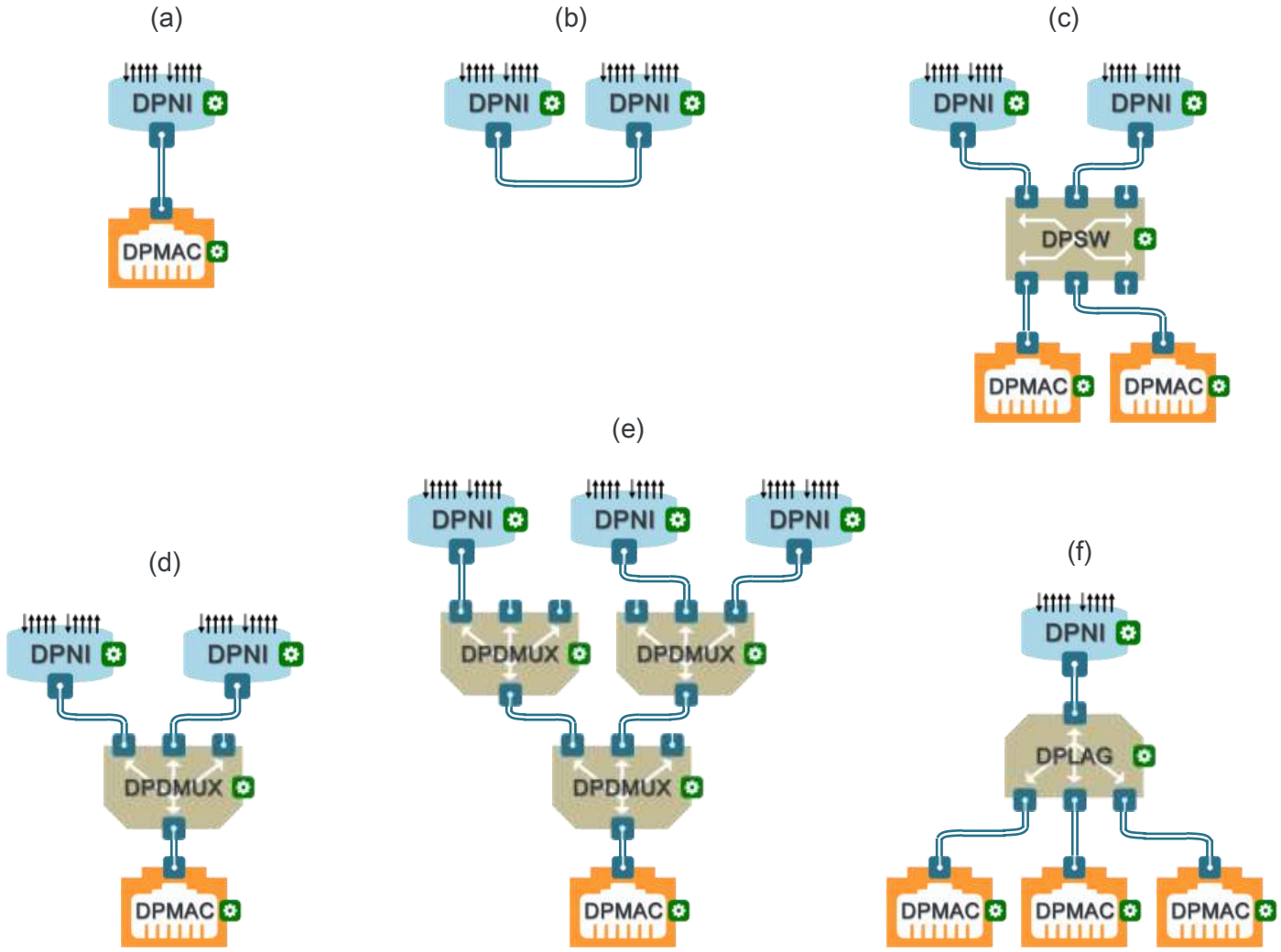
Topology Example #1



Topology Example #2



Typical Network Connections

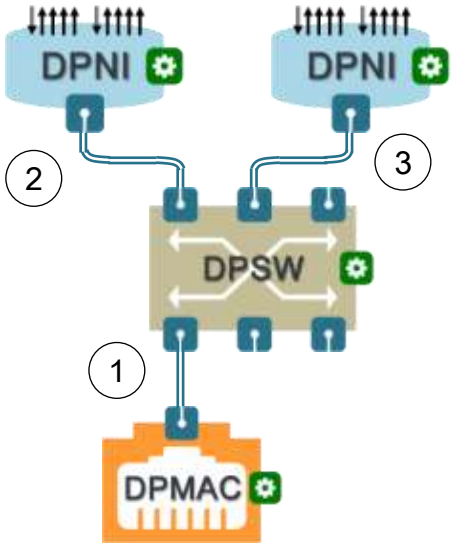


Connecting Network Objects

Connections can be declared in the DPL file:

```
connections {
        connection@1{
                endpoint1 = "dpsw@0/if@0";
                endpoint2 = "dpmac@0";
        };
        connection@2{
                endpoint1 = "dpsw@0/if@5";
                endpoint2 = "dpni@1";
        };
        connection@3{
                endpoint1 = "dpsw@0/if@4";
                endpoint2 = "dpni@2";
        };
};
```

Network Topology View

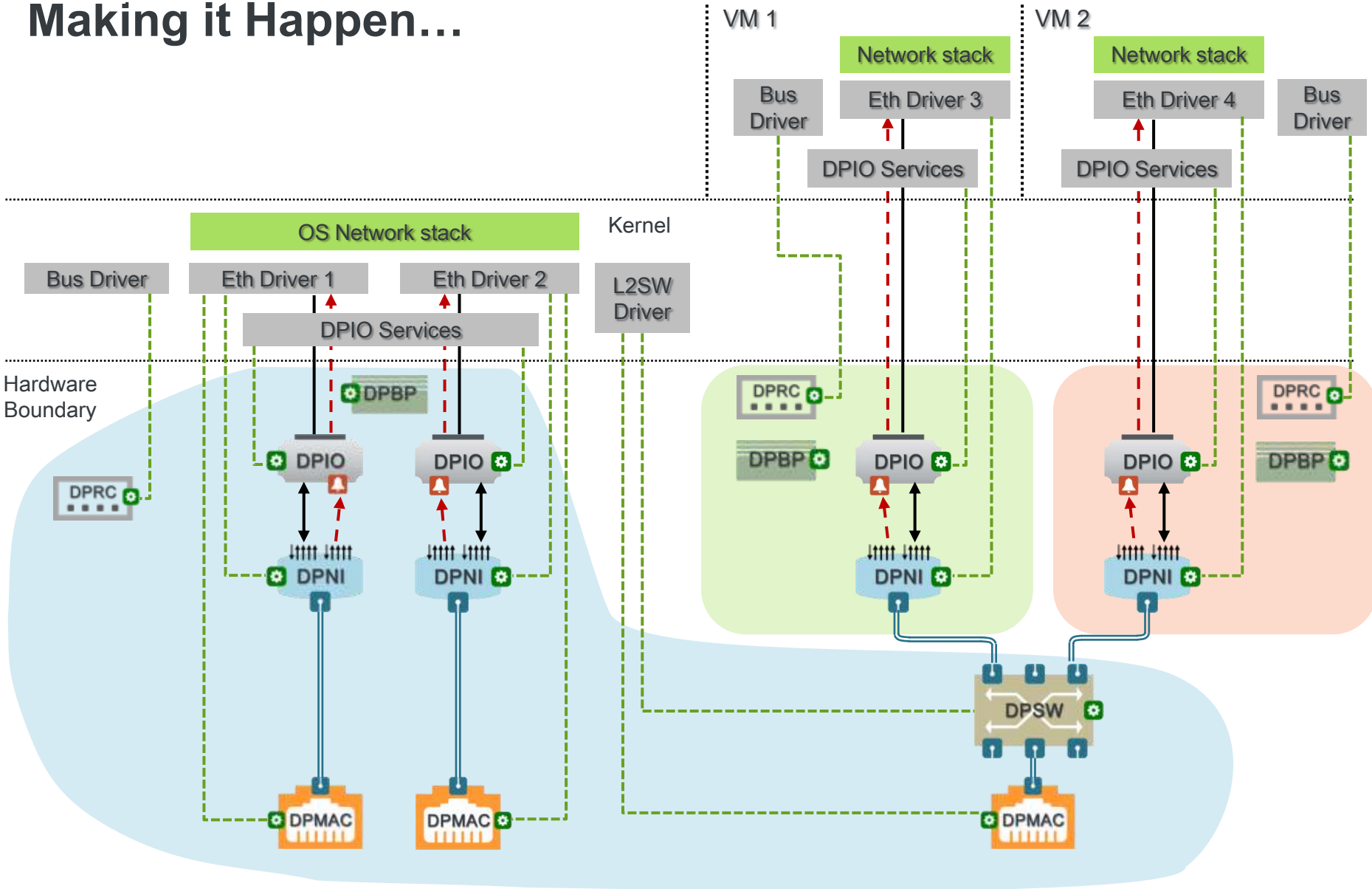


…or created dynamically via DPRC API:

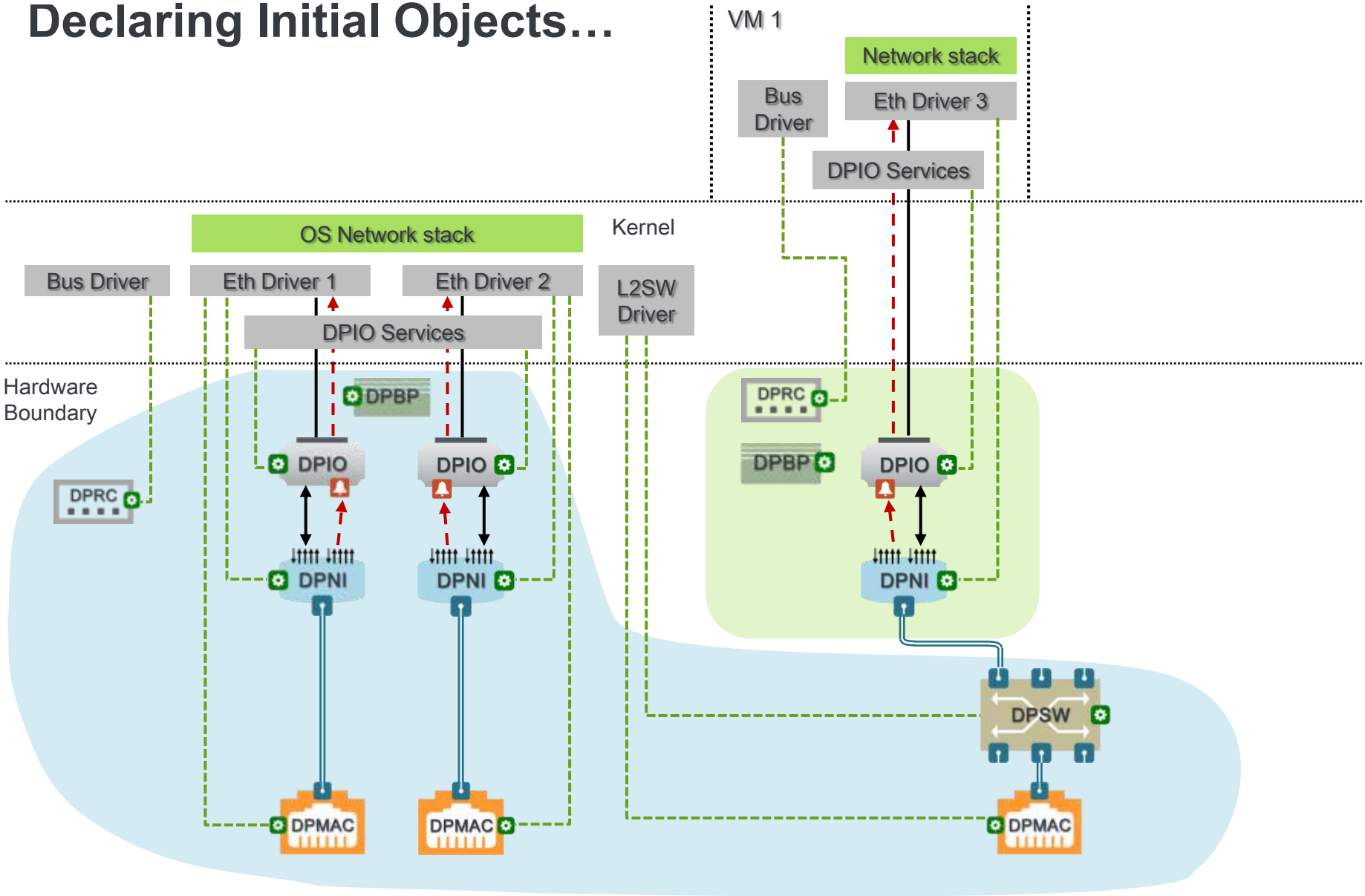
```
1) dprc_connect(dprc, <dpsw-0/0>, <dpmac-0>)

2) dprc_connect(dprc, <dpni-1>, <dpsw-0/5>)

3) dprc_connect(dprc, <dpni-2>, <dpsw-0/4>)
```

Resource Management

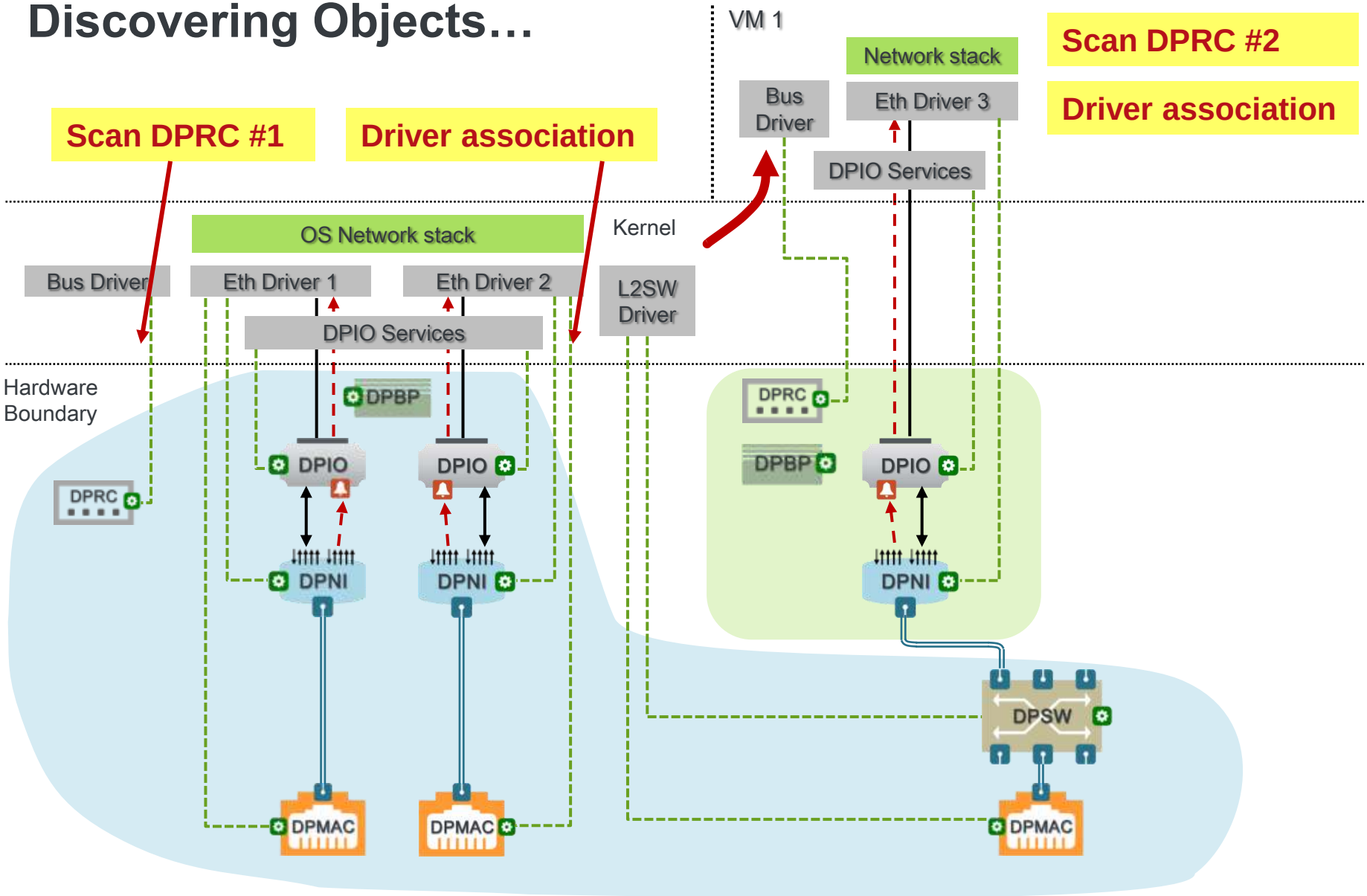
Making it Happen…



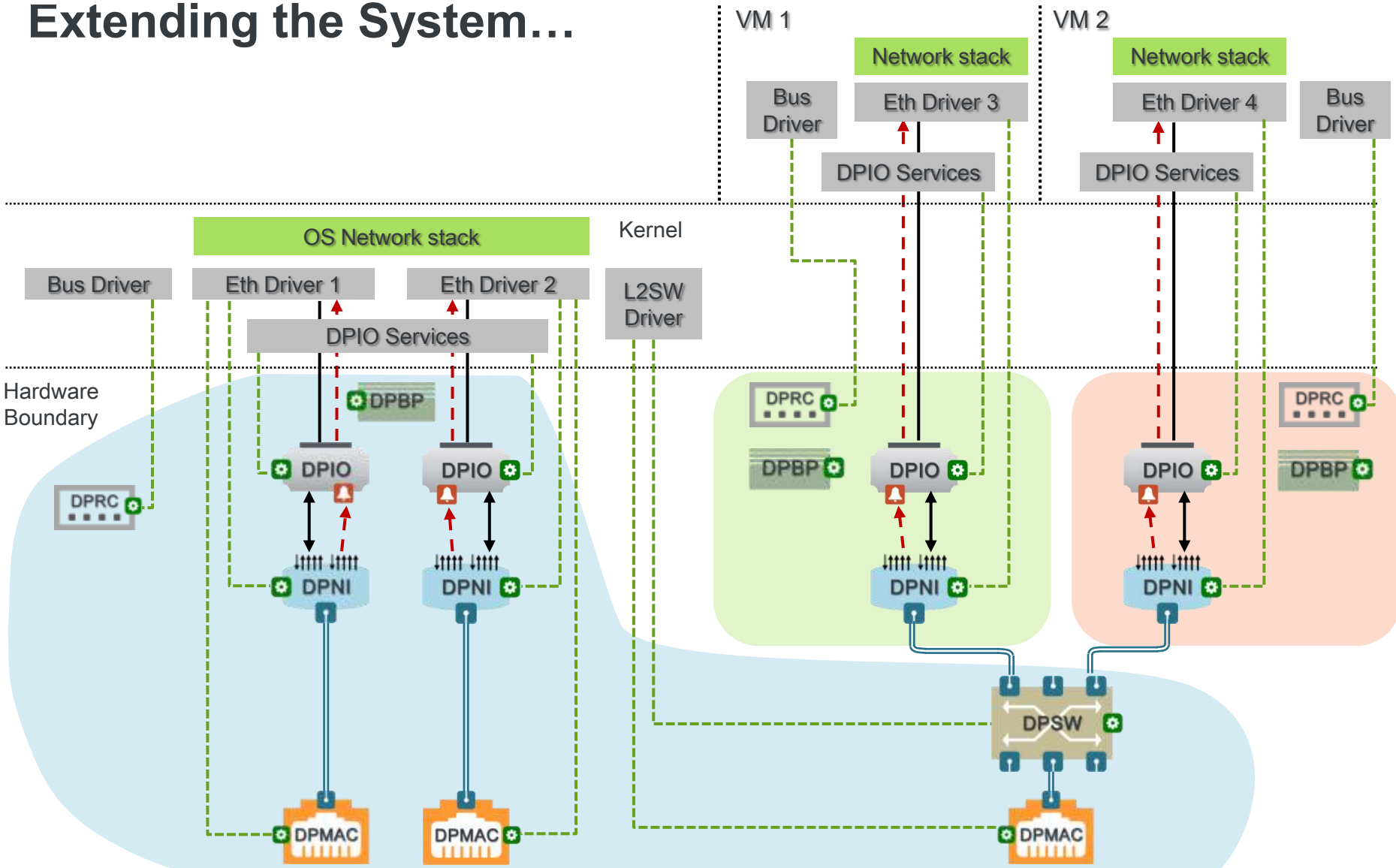
Declaring Initial Objects…



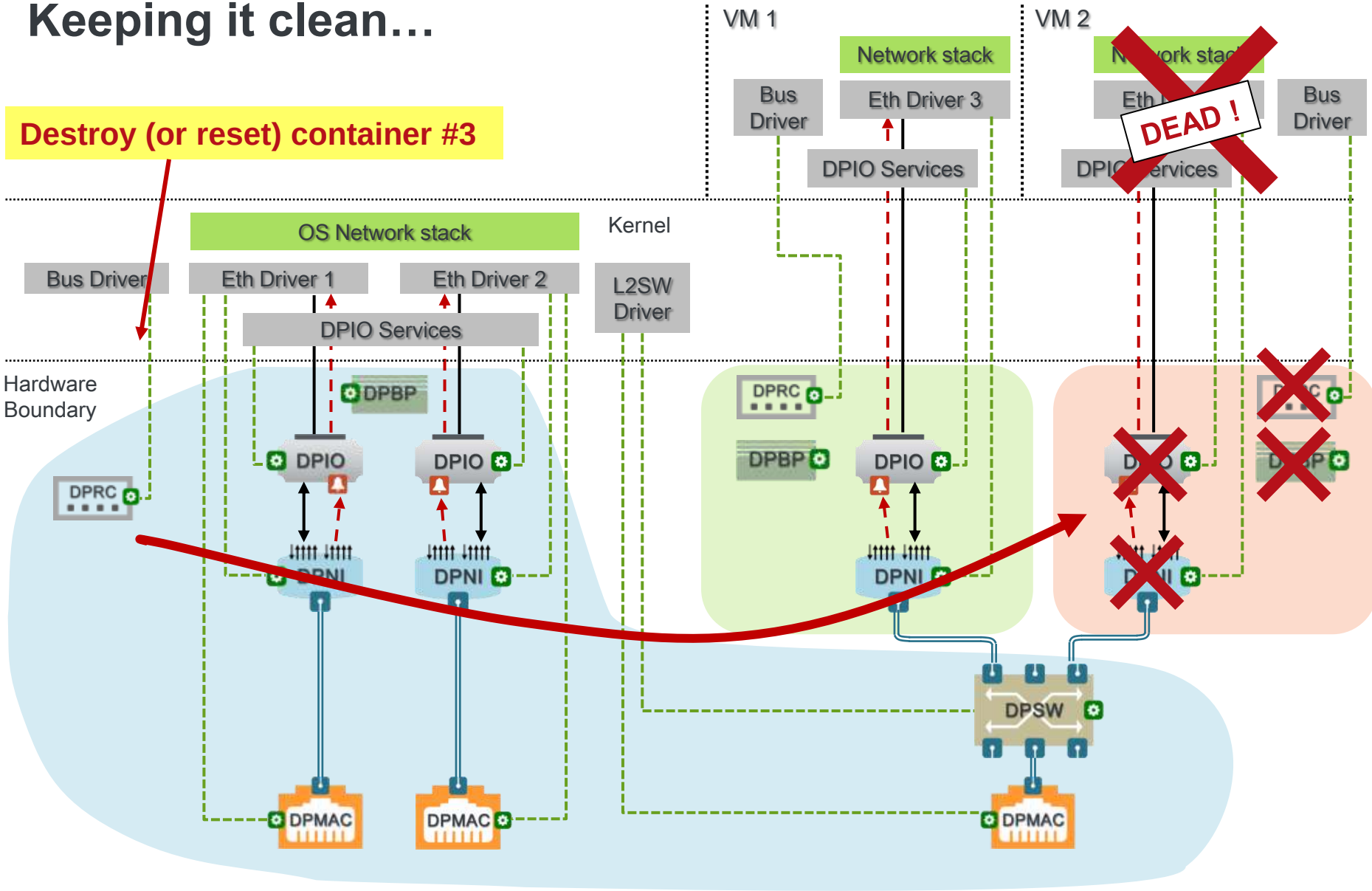
Discovering Objects…



Extending the System…



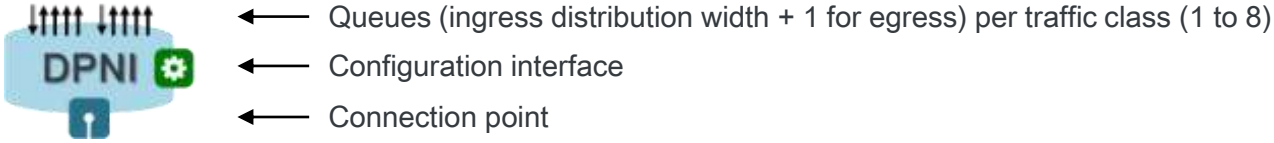
Keeping it clean…



MC Logical Objects

MC Logical Objects

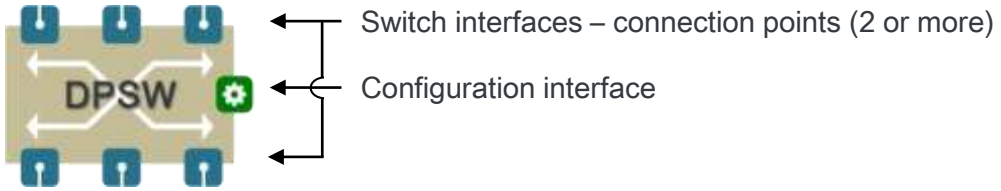
- **Network Interfaces:**
  - **Data Path Network Interface (DPNI)**
    - A standard network interface (L2 and up), as expected by standard network stacks/applications.
    - Offers a wide range of standard offloads:
      MAC & VLAN Filtering, QoS, checksums, time-stamping, policing, IPR, IPF, IPSec, RSC, GSO, etc.
    - Configurable as a tunnel/fast-path interface (non-L2 packet), suitable for GPP-AIOP interaction.



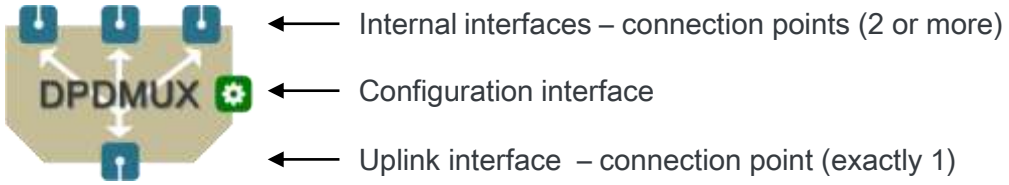
- **Physical Interfaces:**
  - **Data Path MAC (DPMAC)**
    - Serves for physical MAC and MDIO control

MC Logical Objects (cont.)

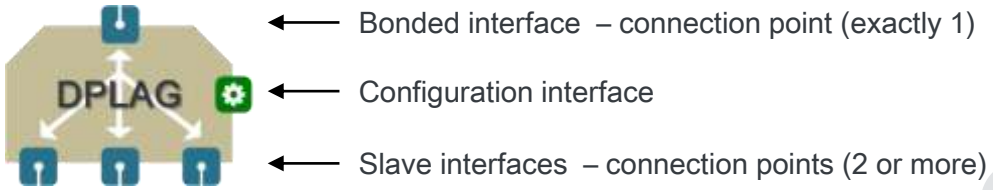
- **Switching and Aggregation:**
  - **Data Path Switch (DPSW) –** Standard implementation of L2 Switch.



  - **Data Path Demux (DPDMUX) –** Allows partitioning of a physical interface into multiple (isolated) logical interfaces. May be used for setting up different Ethernet Virtual Bridging (EVB) objects, such as VEB, VEPA, or S-Component.



  - **Data Path Link Aggregator (DPLAG) –** aggregates multiple physical links into a single logical link. (NOT available in LS2085 rev-1)



MC Logical Objects (cont.)

- **Supporting Infrastructure objects:**
  - **Data Path Buffer Pool (DPBP)** – An abstraction of BMan buffer pool

← Configuration interface

  - **Data Path I/O (DPIO)** – Enables enqueue/dequeue via QMan portals and getting ingress notifications

← GPP interface (QBMan software portal)
← Configuration interface
← Notification channel (optional: notifications from ingress queues or DPCON)

  - **Data Path Concentrator (DPCON)** – Scheduling object for advanced scheduling of ingress packets from multiple interfaces.

← QBMan channel for dequeue
← Configuration interface
← Priorities for scheduling ingress queues (2-8)

MC Logical Objects (cont.)

- **Accelerator Interfaces:**
  - **SEC Accelerator Interface (DPSECI)**

← Queues (1 for ingress + 1 for egress) per priority (1 or 2)
← Configuration interface

  - **DCE Accelerator Interface (DPDCEI)**

← Queues (1 for ingress + 1 for egress)
← Configuration interface

  - **PME Accelerator Interface (DPPMEI) + PME database (DPPME)**

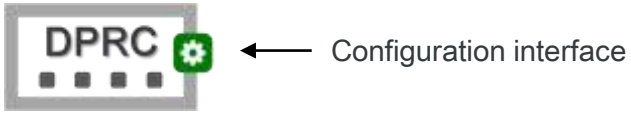
← Queues (1 for ingress + 1 for egress)
← Configuration interface

  - **QDMA Accelerator Interface (DPDMAI)**

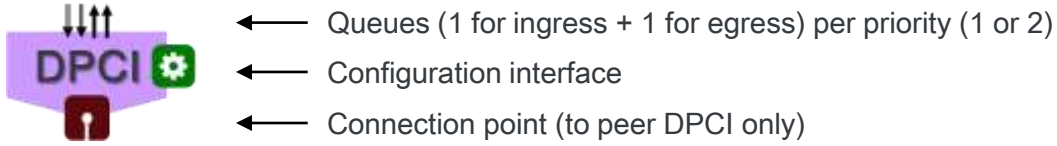
← Queues (1 for ingress + 1 for egress) per priority
← Configuration interface

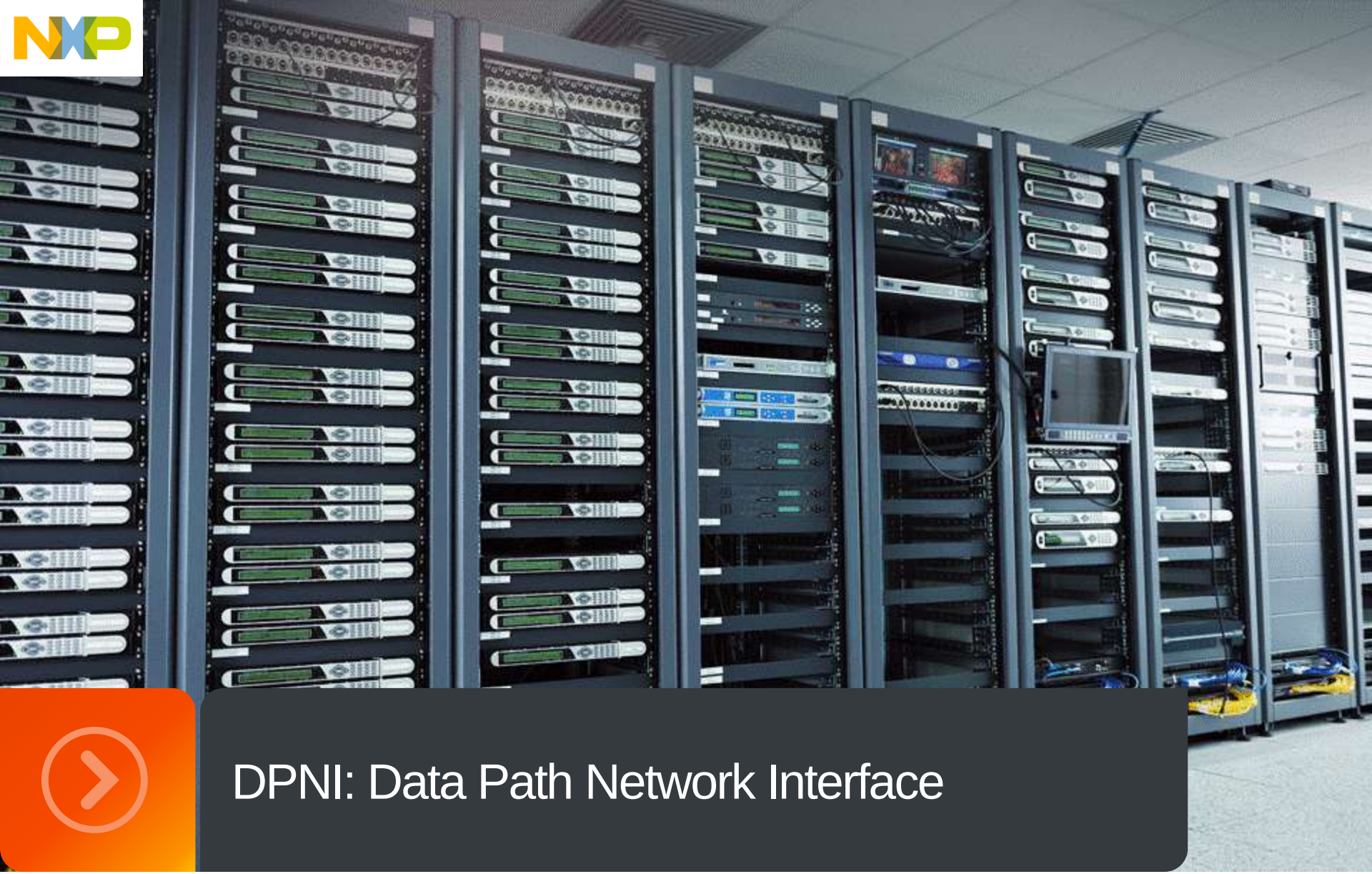
MC Logical Objects (cont.)

- **Management objects:**
  - **Data Path Resource Container (DPRC):**
    - Allows software context to assign DPAA objects and resources.
    - Allows software context to create network topology by connecting network objects.
    - Functions as virtual bus, so software context may query DPAA objects and associate with OS device drivers.



- **Inter-Partition Communication:**
  - **Data Path Communication Interface (DPCI)** – allows communication between different software contexts through QMan infrastructure, which is not limited to network packet format. Useful for IPC between two GPP software entities, or between GPP and AIOP entities. The communication protocol is user-defined.





DPNI: Data Path Network Interface

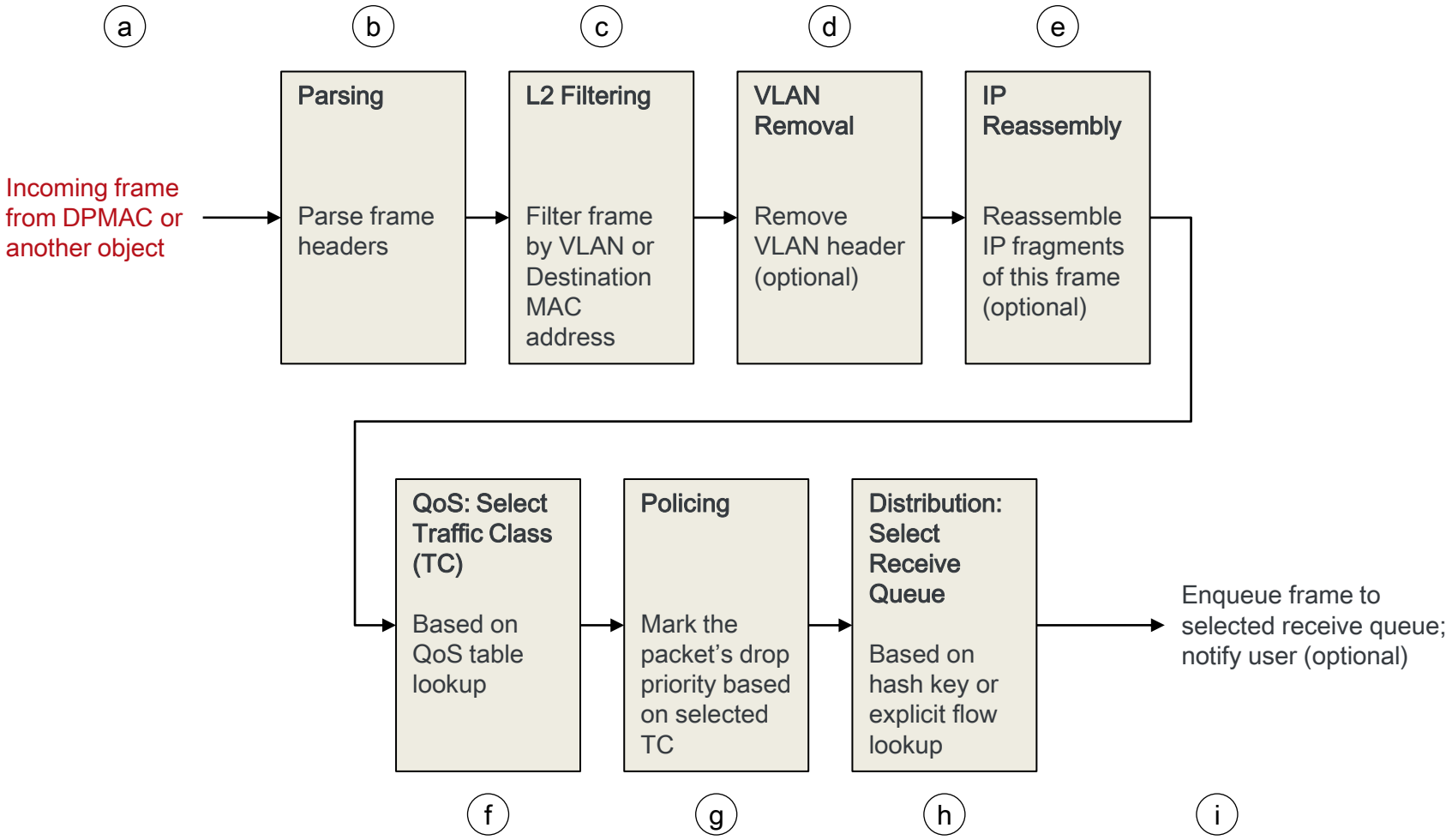
Data Path Network Interface (DPNI)

- **Wire-speed frame parsing**
  - Parsing results may be visible in frame annotation area
- **Filtering of received frames**
  - Exact-match filtering based on destination MAC address and/or VLAN IDs
  - Unicast promiscuous and Multicast promiscuous modes
- **QoS support**
  - Packet classification up to eight traffic classes, based on user-defined key
  - Policing based on classification result (tail-drop or WRED)
- **Distribution to receive queues**
  - Statistical distribution based on hash-generated key (RSS)
  - Explicit flow steering based on user-defined key
- **Up to eight different buffer pools**
- **Various scheduling options for received packets**

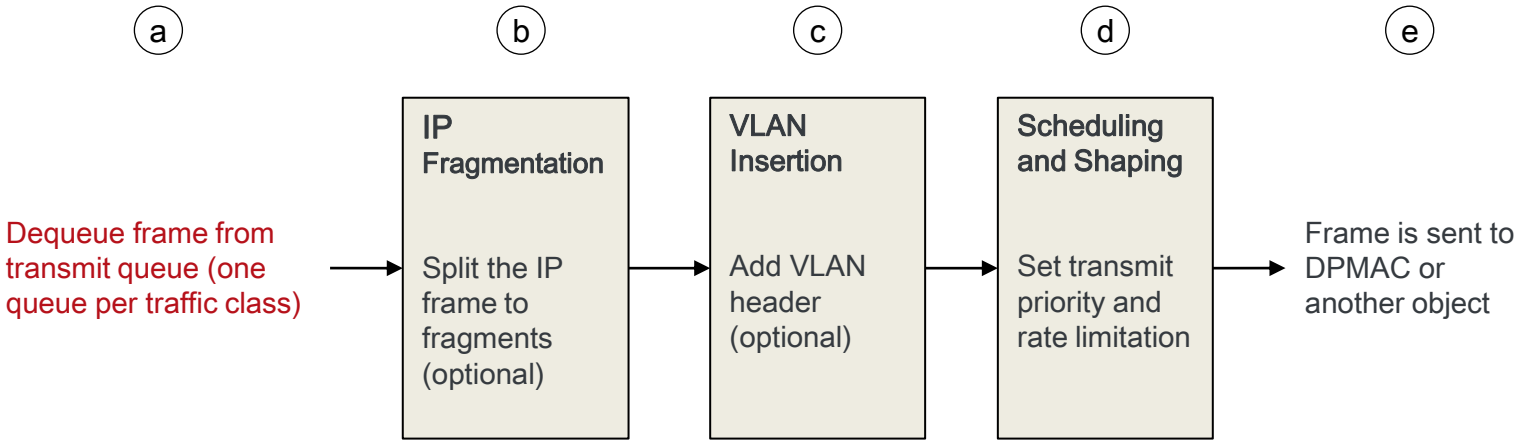
Data Path Network Interface (DPNI)

- **Traffic shaping of transmitted packets**
  - Up to eight transmit queues (traffic classes)
  - Rate limit
- **Various offload functions:**
  - L3 and L4 checksum generation and validation
  - VLAN add/remove
  - IP Reassembly and Fragmentation
  - GRO and GSO
  - IPSec transport
- **Priority-based Flow Control (PFC)**
  - Supporting queue congestion and/or buffer pool depletion
- **PTP (IEEE 1588) time-stamping**
- **Network interface statistics**
- **Full reset**

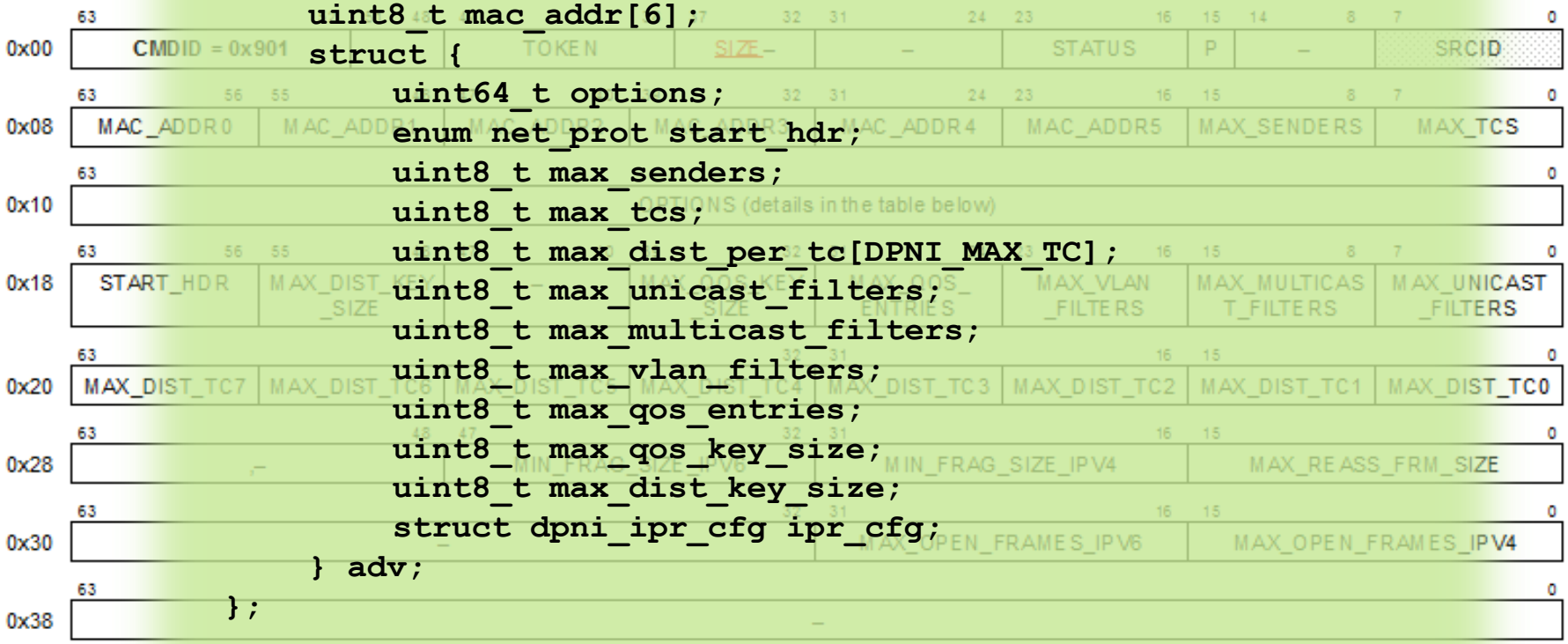
DPNI Ingress Frame Processing



DPNI Egress Frame Processing



Objects Configuration – Easy to Use API




```
struct dpni_cfg {
        uint8_t mac_addr[6];
        struct {
                uint64_t options;
                enum net_prot start_hdr;
                uint8_t max_senders;
                uint8_t max_tcs;
                uint8_t max_dist_per_tc[DPNI_MAX_TC];
                uint8_t max_unicast_filters;
                uint8_t max_multicast_filters;
                uint8_t max_vlan_filters;
                uint8_t max_qos_entries;
                uint8_t max_qos_key_size;
                uint8_t max_dist_key_size;
                struct dpni_ipr_cfg ipr_cfg;
        } adv;
};

int dpni_create(struct fsl_mc_io *mc_io,
                const struct dpni_cfg *cfg,
                uint16_t *token);
```

Example: NIC Creation Sequence

```
/* (1) DPIO creation */
dpio_cfg.channel_mode = DPIO_LOCAL_CHANNEL;
dpio_cfg.num_priorities = 4;
dpio_create(drv->mc_io, &dpio_cfg, &token);
dpio_enable(drv->mc_io, token);

/* (2) DPBP creation */
dpbp_create(drv->mc_io, &dpbp_cfg, &token);
dpbp_enable(drv->mc_io, token);
dpbp_get_attributes(drv->mc_io, token &dpbp_attr);
/* use dpbp_attr.bpid to fill buffers pool*/
```

```
/* (3) DPNI creation */
dpni_cfg.mac_addr = { ... };
dpni_cfg.adv.max_tcs = 4;
dpni_cfg.adv.max_unicast_filters = 32;
(+ other standard features / offload features)
dpni_create(drv->mc_io, &dpni_cfg, &token);

/* attach buffer pool */
pools_cfg.num_dpbp = 1;
pools_cfg.pools[0].dpbp_id = dpbp_attr.id;
pools_cfg.pools[0].buffer_size = 512;
dpni_set_pools(drv->mc_io, token, &pools_cfg);

dpni_enable(drv->dpni);

/* runtime control operations */
dpni_add_vlan_id(drv->mc_io, token, 0x0200);
```

DPSW: Data Path Switch

Data Path Switch (DPSW)

- Supports 802.1Q switching, based on (outer) VLAN and MAC address
- Supports **separate FDB** (MAC table) per VLAN, or **shared FDB** for multiple VLANs
- **Flooding** (configuration per VLAN)
- **Three address learning** modes, selected per FDB:
  - Automatic learning by the switch hardware
  - Secure learning by host GPP software
  - Non-secure learning by host GPP software
- Supports various VLAN handling options:
  - Supports port-based VLAN – definition of default VLAN per interface
  - Untagged frames transmission
  - Untagged frames admittance: admit tagged and untagged, or only tagged frames
  - VLAN filtering – dropping frames with unregistered VLANs
  - Supports trunk interface – accepting all VLANs
  - Two custom TPIDs per switch
- **Interface mirroring**, with option to mirror only specific VLAN
- **STP/RSTP/MSTP marking** (Spanning Tree Protocol handled by GPP software)

DPSW Features

- Supports 802.1Q switching, based on (outer) VLAN and MAC address
- Supports **separate FDB** (MAC table) per VLAN, or **shared FDB** for multiple VLANs
- **Flooding** (configuration per VLAN)
- **Three address learning** modes, selected per FDB:
  - Automatic learning by the switch hardware
  - Secure learning by host GPP software
  - Non-secure learning by host GPP software
- Supports various VLAN handling options:
  - Supports **port-based VLAN** – definition of default VLAN per interface
  - **Untagged frames transmission** (configuration per VLAN/interface)
  - **Untagged frames admittance**: admit tagged and untagged, or only tagged frames
  - **VLAN filtering** – dropping frames with unregistered VLANs
  - Supports **trunk interface** – accepting all VLANs (configuration per interface)
  - **Two custom TPIDs** per switch
- **Interface mirroring**, with option to mirror only specific VLAN
- **STP/RSTP/MSTP marking** (Spanning Tree Protocol handled by host GPP software)

DPSW Features (cont.)

- Supports **QoS capabilities**:
  - Traffic class selection based on DSCP or 802.1P
  - Transmission bandwidth allocation per traffic class
  - Transmission rate configuration per interface
  - WRED on ingress (configuration per traffic class)
- Supports **policy-based forwarding on ingress** (TCAM lookup on L2-L4 fields)
- Supports **forwarding of selective protocols to a control interface**, for example:
  - Ethernet monitoring, Multicast management (IGMP/MLD), Spanning Tree (BPDU), etc.
- **PTP** (IEEE 1588) time-stamping
- **Priority-based Flow Control** (PFC) based on queue congestion and/or buffer depletion
- **Statistics** counters per interface
- **Link state indication** per interface (with interrupts on changes)
- Supports **interface enable and disable** operations
- Supports **switch enable and disable** operations
- Supports **switch reset** operation

MC Makes it Easy

- Presents hardware as logical objects
- Virtualizes and isolates objects
- Hides complex sequences
- Manages resources
- Manages the Network-on-Chip
- Extends capabilities through sharing

www.Freescale.com