DPDK
DATA PLANE DEVELOPMENT KIT

Dynamic Device Management

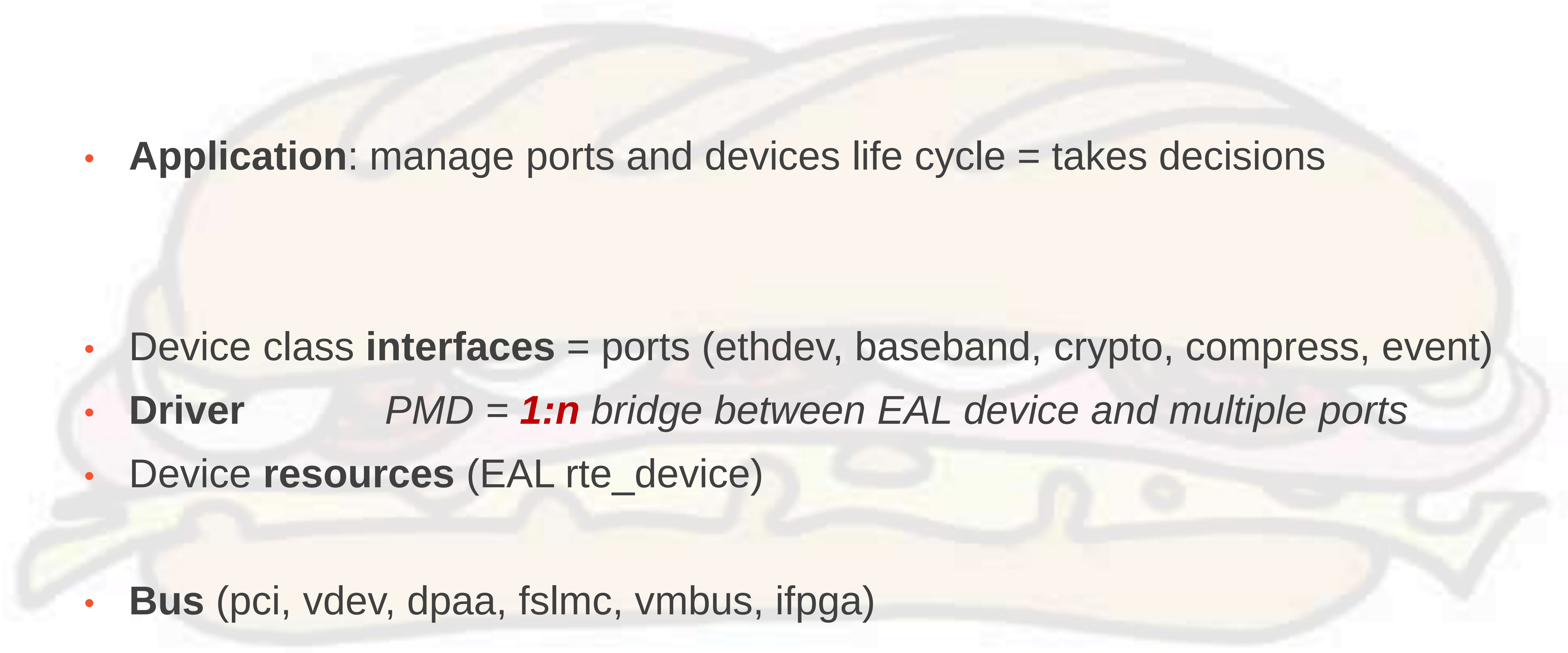
THOMAS MONJALON - MELLANOX
JEFF GUO - INTEL
QI ZHANG – INTEL

SEPTEMBER 2018, DUBLIN

DPDK origin = statically allocated resources

- CPU
  - No hotplug yet
- Memory
  - Dynamic since 18.05
- Devices
  - Work in progress

Layers



- **Application**: manage ports and devices life cycle = takes decisions

- Device class **interfaces** = ports (ethdev, baseband, crypto, compress, event)
- **Driver** *PMD = **1:n** bridge between EAL device and multiple ports*
- Device **resources** (EAL rte_device)

- **Bus** (pci, vdev, dpaa, fslmc, vmbus, ifpga)

Buses

- **PCI** – *historical one* – *best supported*
- VDEV
- NXP DPAA (17.11) / fslmc DPAA2 (17.05)
- Windows Hyper-V VMBus (18.08)
- iFPGA (18.05)

Devargs syntax

- Legacy syntax
  - Assume PCI id (BDF)
  - PMD-specific options

- New proposed syntax
  - More explicit: `bus=pci,id=BDF/class=eth,…/driver=virtio,…`
  - 18.08: introduce new parser
  - 18.11: implement syntax properties

- New proposed option
  - `--vdev` replaced by `--dev` generic option
  - `--whitelist` replaced by `--dev` option

Blacklist / Whitelist

- Static lists defined at initialization
- Control bus probing

- Should allow **dynamic** policy
- **Future** API to design
  - Application callback during probing?
  - DPDK lists updated via API?

Probe on demand

- from **ethdev** (legacy vdev use case)
  - rte_eth_dev_attach(const char *devargs, uint16_t *port_id) *deprecated in 18.08*
  - mixing EAL devargs and ethdev port

- from **EAL** (legacy failsafe case)
  - rte_eal_dev_attach(const char *name, const char *devargs) *deprecated in 18.08*
  - supports only PCI and VDEV buses

- from **EAL**
  - **rte_eal_hotplug_add**(const char *busname, const char *devname, const char *devargs)
  - Should be simplified: only one parameter for new devargs syntax
  - Multiple match requires option to skip already probed devices

Remove on demand

- from **ethdev**
  - **rte_eth_dev_close**(uint16_t port_id)
  - Should call rte_eth_dev_release_port
  - Should trigger resource freeing at EAL level

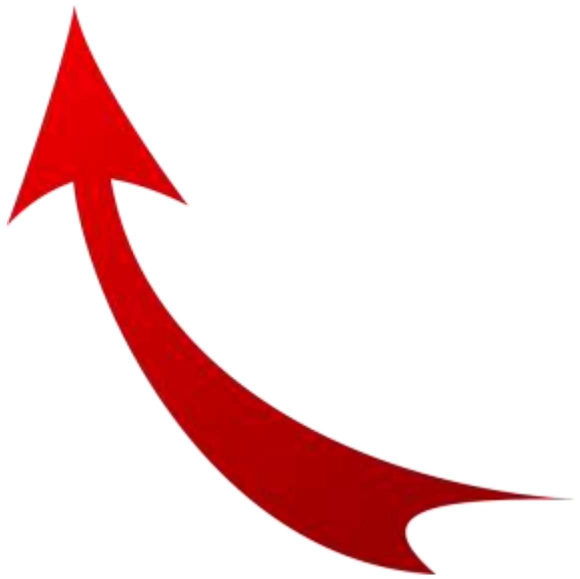
- from **EAL**
  - **rte_eal_hotplug_remove**(const char *busname, const char *devname)
  - Should be simplified: only one parameter (devargs? rte_device?)

Notifications jungle

- ethdev port events
  - RTE_ETH_EVENT_NEW
  - RTE_ETH_EVENT_DESTROY
  - *introduced in 18.02 / fixed in 18.05*

  - RTE_ETH_EVENT_INTR_RMV
  - *introduced for failsafe in 17.05*

- hardware events from kernel
  - Linux support: uevent
  - upper layer notification
    - RTE_DEV_EVENT_ADD
    - RTE_DEV_EVENT_REMOVE
  - *introduced in 18.05*



Hotplug sequences

- It’s a mess currently in PMDs

- Hotplug should be:
  - uevent
  - RTE_DEV_EVENT_ADD
  - **application** calls rte_eal_hotplug_add
  - PMD probe ports
  - PMD calls rte_eth_dev_probing_finish
  - RTE_ETH_EVENT_NEW
  - **application** get new ports

- Unplug should be:
  - uevent
  - RTE_DEV_EVENT_REMOVE
  - **application** calls rte_eal_hotplug_remove
  - PMD calls rte_eth_dev_removing ?
  - RTE_ETH_EVENT_DESTROY
  - **application** calls rte_eth_dev_close

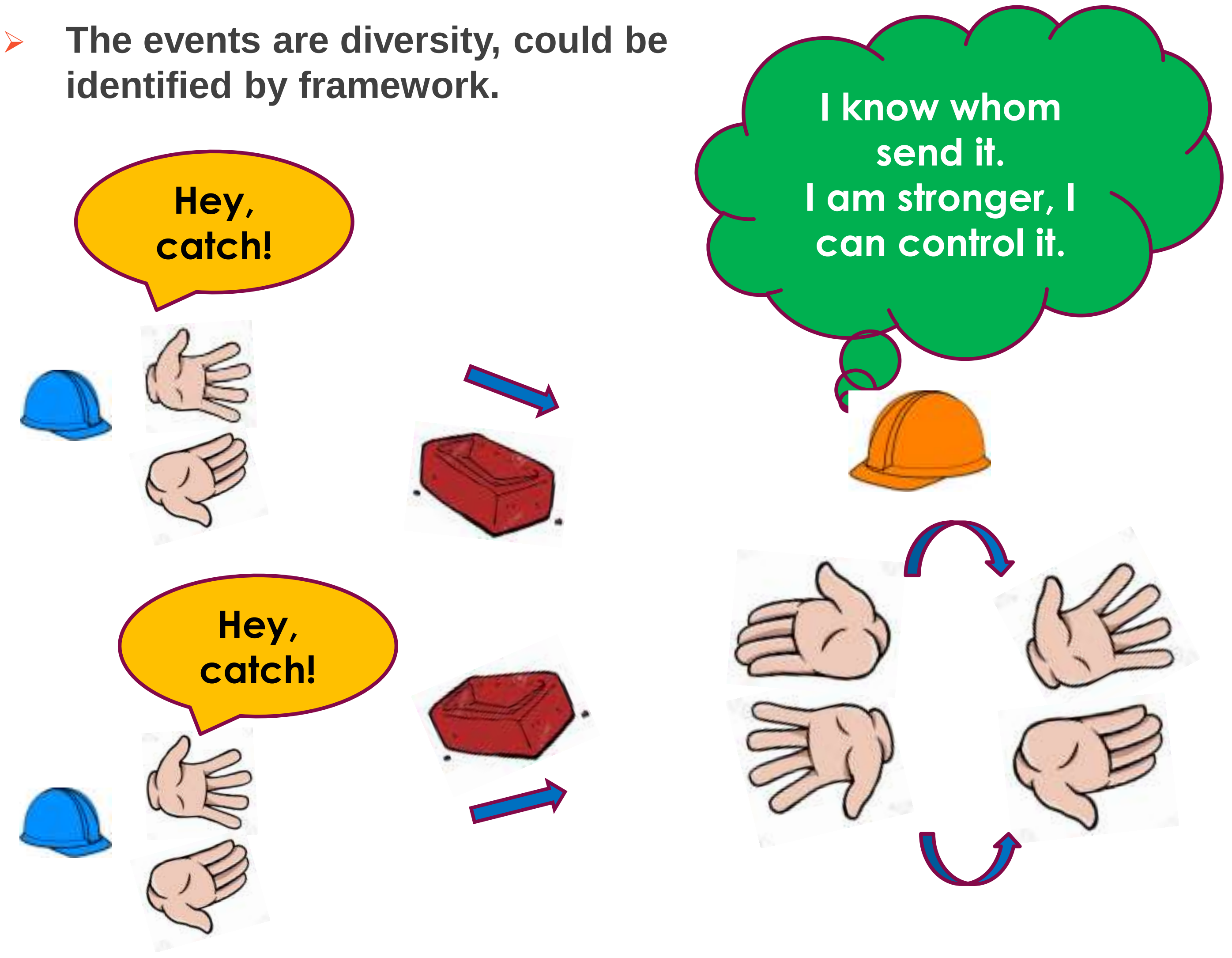
- And/Or
  - RTE_ETH_EVENT_INTR_RMV if supported
  - application calls rte_eth_dev_close

Rx/Tx during unplug

- PMD can stop any request if aware of the event
  - mlx case
- generic SIGBUS handler on device address ranges
  - *pending for 18.11*

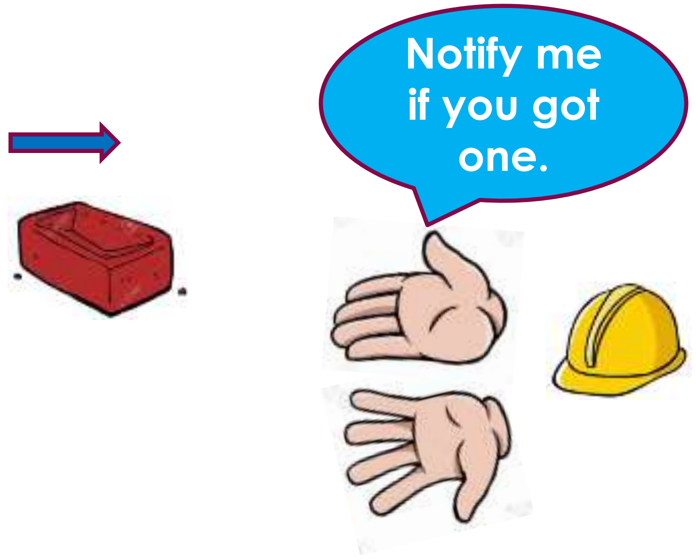
Hardware hotplug handle’s proposal

- The events are diversity, could be identified by framework.



- Kernel handle and user space handle are independent. Framework help to decoupling the segment tasks.

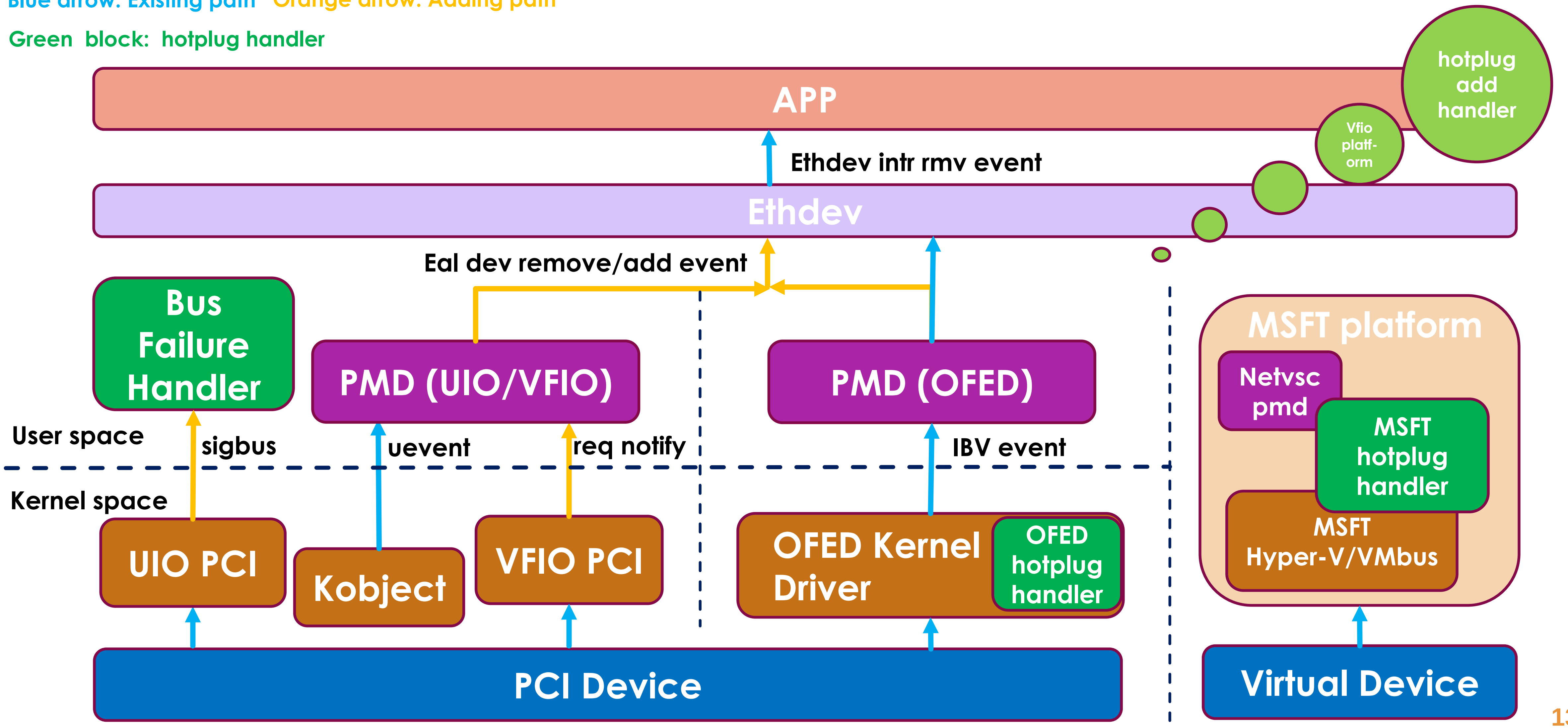
- Framework provide service for taking over the control at some break point or handle some tough task.



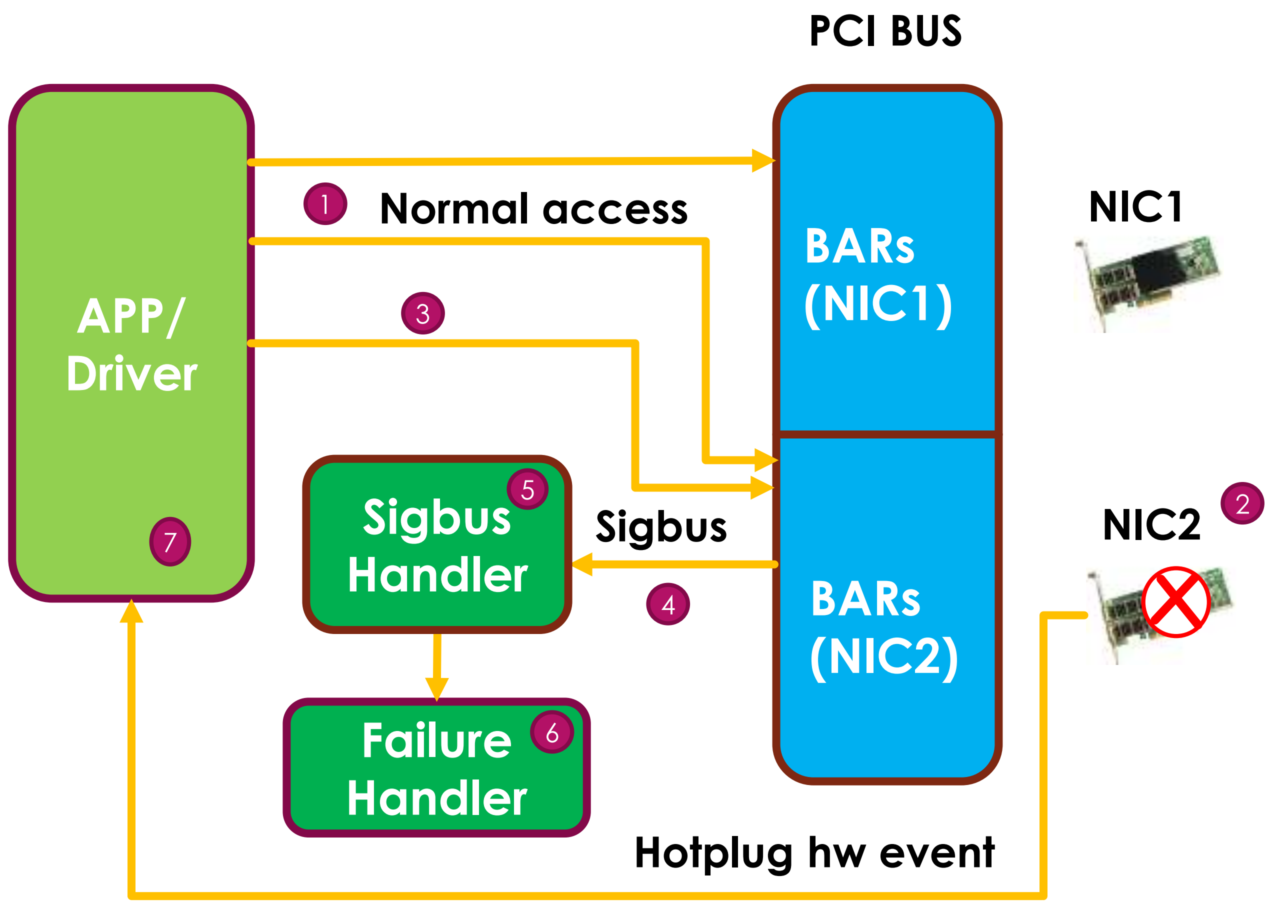
Hardware event and handler

Blue arrow: Existing path   Orange arrow: Adding path

Green block: hotplug handler



UIO/VFIO PCI hotplug failure handler



1) -> 2) Nic2 is suddenly broken when working.

3) Irresistible access the BARs of Nic2.

4)Kernel issue sigbus error.

5)Signal handler identifies the faulting BARs.

6)Failure handler guaranty the rest memory access, by remap a new fake one.

7)Handle hotplug hw event, stop process and detach device.

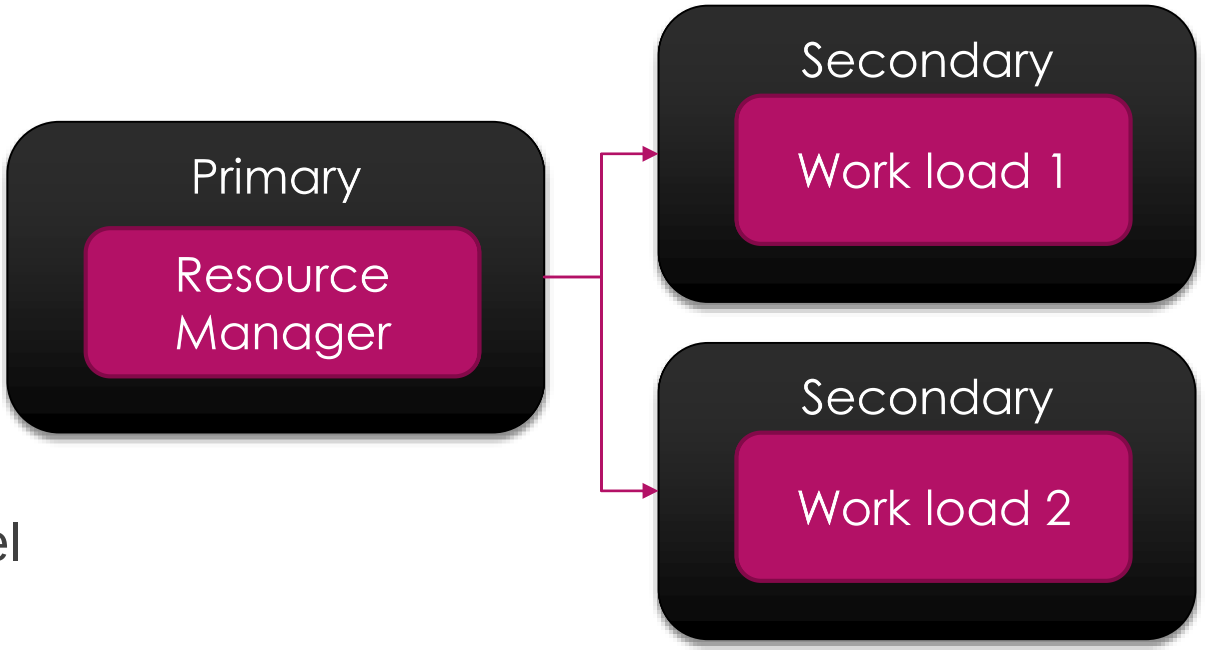
UIO hotplug: 1)-2)-3)-4)-5)-6)-7)

VFIO special hotplug: VFIO kernel specially send release request and monitor status, it will not delete device until user space release device resource.

It is 1)-2)-3)-7).

Multi-process

- Gap: hotplug does not support multi-process
- Patches pending for 18.11

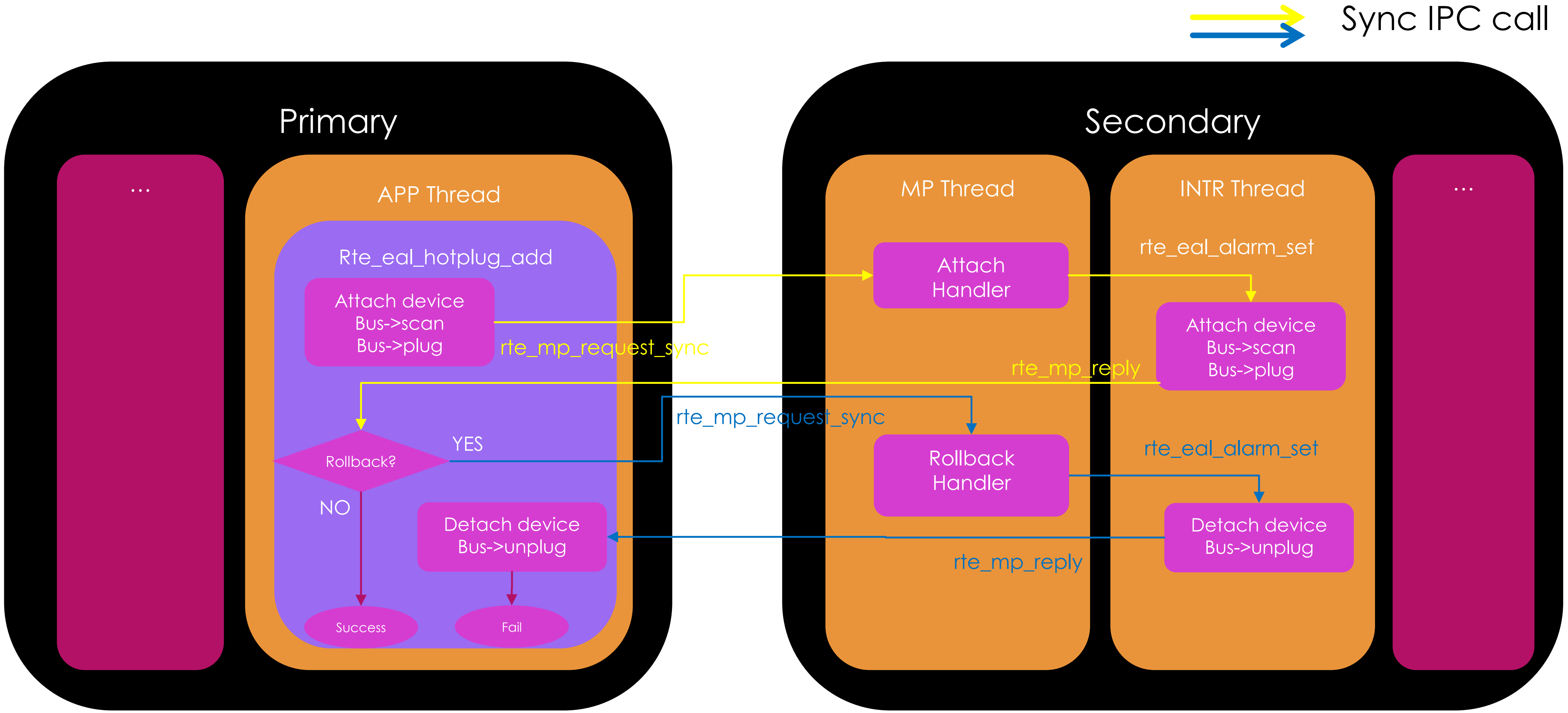


- Broadcast hotplug messages via IPC channel
- Rollback for any failure
- NO need for private vdev

- On unplug
  - Port detached locally by each process
  - Last process will destroy the port

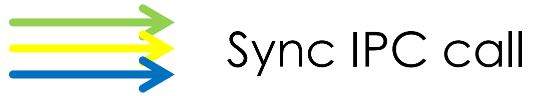
Multi-process implementation (1)

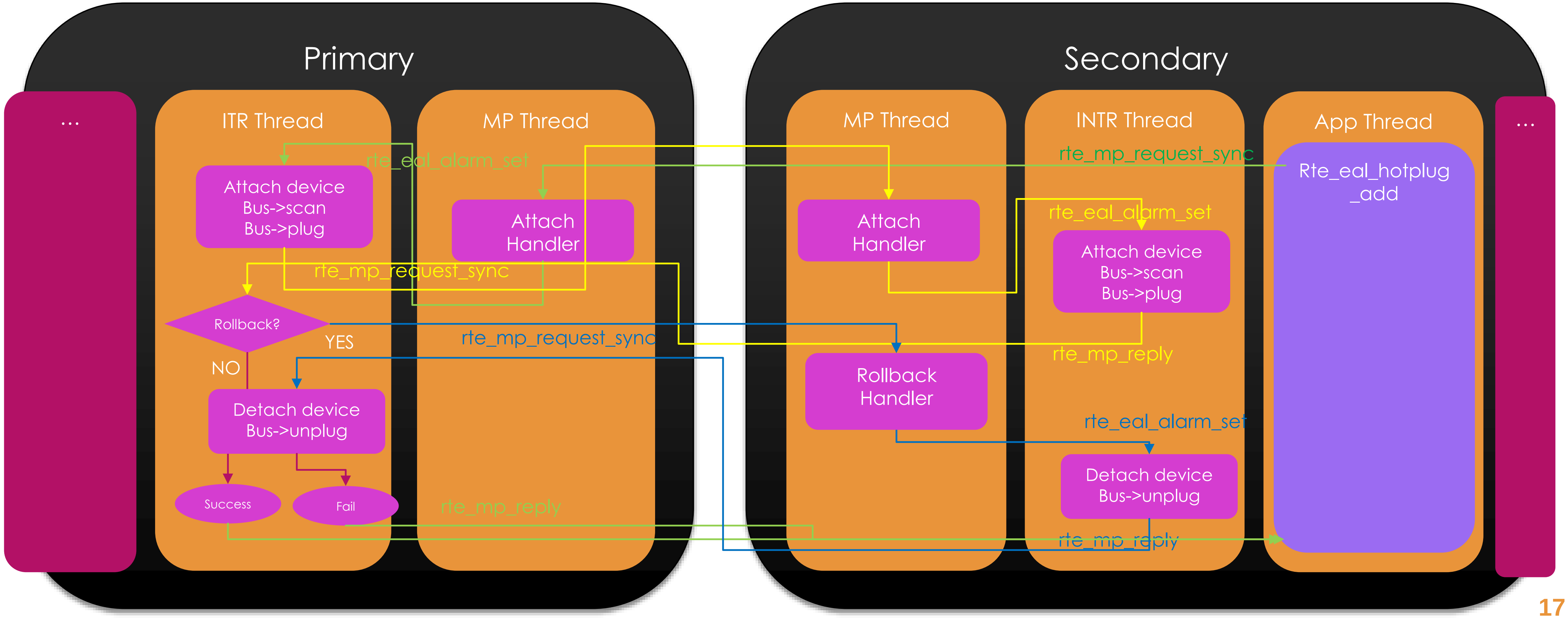
- **Attach a device from Primary**



Multi-process implementation (2)

- **Attach a device from Secondary**





Multi-process hotplug - Future works

- Reliable implementation: reserve shared memory spaces
  - http://patchwork.dpdk.org/patch/40537/
- Support replying sync IPC request from a separate thread
  - remove existing hacky code
- Expose driver capability
- Safe device detaching
  - handshake? ownership?

Device migration

- **at ethdev level**

- **Bonding**
  - Slave devices are configured separately
  - Master and slaves are all seen by the appplication

- **Failsafe**
  - Sub-devices get the same configuration
  - Migration transparent to the application
  - Only failsafe port is seen by the (good) applications (RTE_ETH_DEV_NO_OWNER)
    - Ownership introduced in 18.02

Ownership

- Application (or upper layer like failsafe) can own an ethdev port.
- Recommended to get ownership on new port event.

- Only one entity can own a port (locks).
- By convention, only the owner should manage a port.
- The port iterator can list own ports.

- Usages:
  - Failsafe sub-devices are owned (i.e. managed) by failsafe
  - Multi-process can protect itself

ethdev iterator

- Legacy iteration of ports was
  - `for(int port; port < rte_eth_dev_count(); port++)`
  - not hotplug proof
  - not ownership proof

- Applications encouraged to fix port iteration
  when deprecating rte_eth_dev_count (in 18.05) and should be removed in 18.11.

- Iterators are
  - RTE_ETH_FOREACH_DEV
  - RTE_ETH_FOREACH_DEV_OWNED_BY

Device classes

- ethdev
  - Ownership
  - Iterator
  - Events
  - Failover vdev

- baseband
- crypto
- compress
- event

  - **TODO**

“

Thank you

”

Questions?