Improving security and flexibility within Windows DPDK networking stacks

RANJIT MENON – INTEL

OMAR CARDONA – MICROSOFT

Agenda

DPDK
DATA PLANE DEVELOPMENT KIT

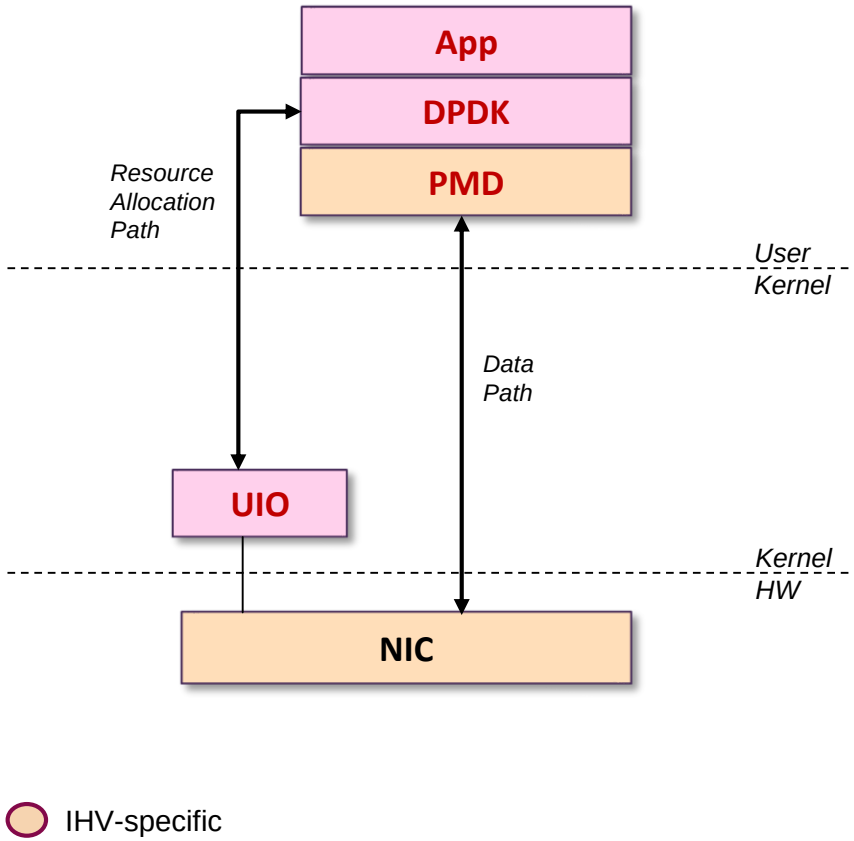
- The story so far…
- Windows DPDK Architecture
- Proposing a change to the architecture
- Benefits of new architecture
- “Secure” API Interface
- Multi-process/multi-user security
- Multi-tenancy security
- Availability
- Further areas of investigation

The story so far…

- Support for DPDK on Windows announced a year ago at this summit
- Code available in a draft repo (*dpdk-draft-windows*)
  - dpdk.org – compatible with release 18.08
- Many of the core libraries available on Windows
  - *librte_eal, librte_ethdev, librte_mbuf, librte_mempool etc.*
- Seeing increasing interest with some key industry partners
  - video / media processing

Windows DPDK architecture

- Similar to the architecture on Linux and other OS
- Uses UIO driver to allow user-space access to networking hardware
- UIO driver required to allocate physically contiguous memory

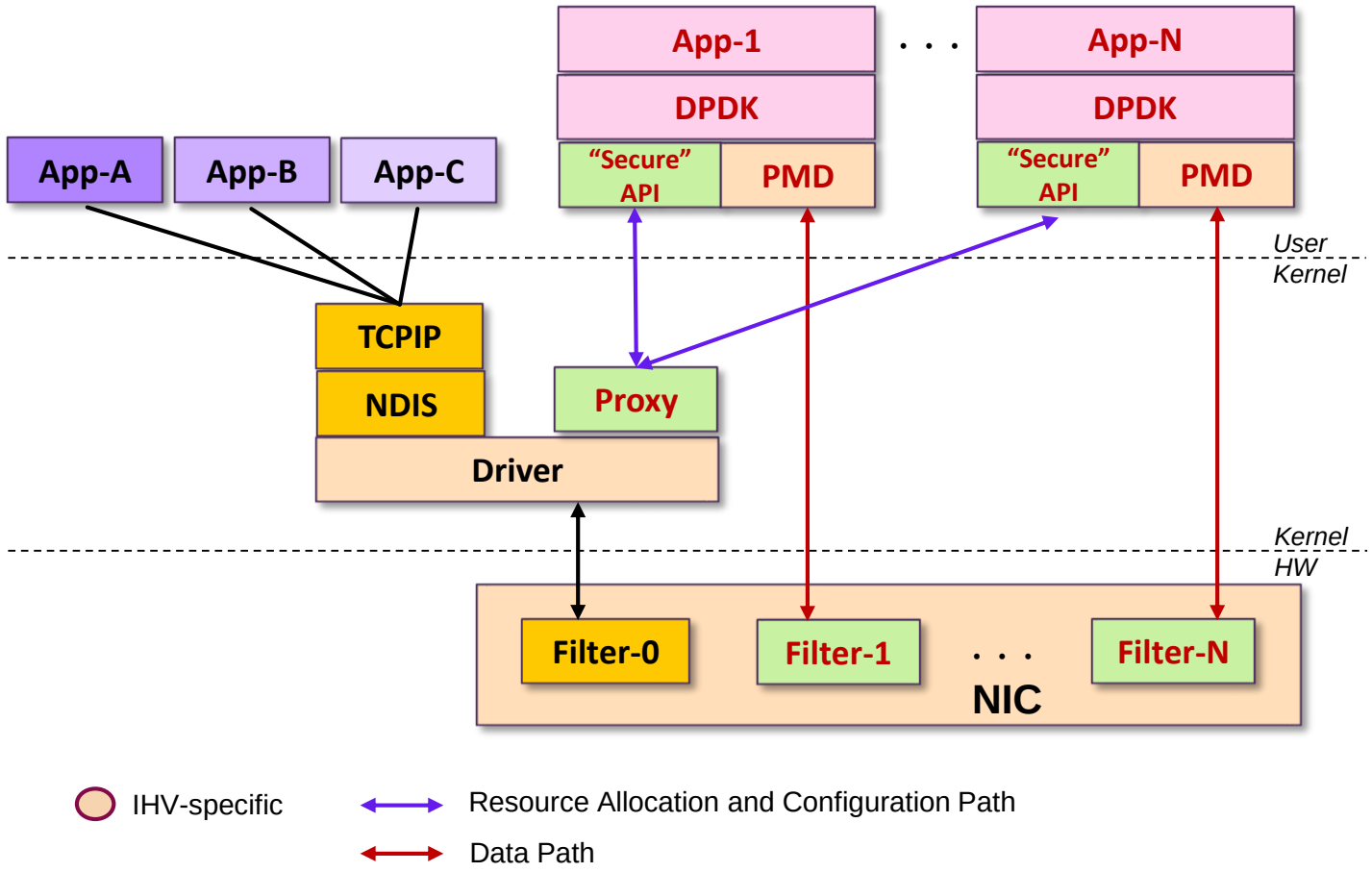


Not ideal…

- UIO driver takes over the whole networking device – inefficient use of network resources
  - Will not work with Live Migration when using a single device
- Not multi-user/multi-process secure
- Networking device cannot be shared with kernel Ethernet driver
- UIO driver needs to be certified and signed independently by DPDK consumers on Windows leading to complicated ecosystem deployment
- **Need a solution that provides the ability to “share” NIC with multiple DPDK VNFs and hypervisor/host in a secure manner**

Proposing a change to the architecture

- Extend kernel Ethernet (NDIS) driver to provide a secure, multi-consumer interface to networking device
- “Secure API” interface would be used to initialize networking resources for DPDK
- Network device can be shared with host and other DPDK consumers

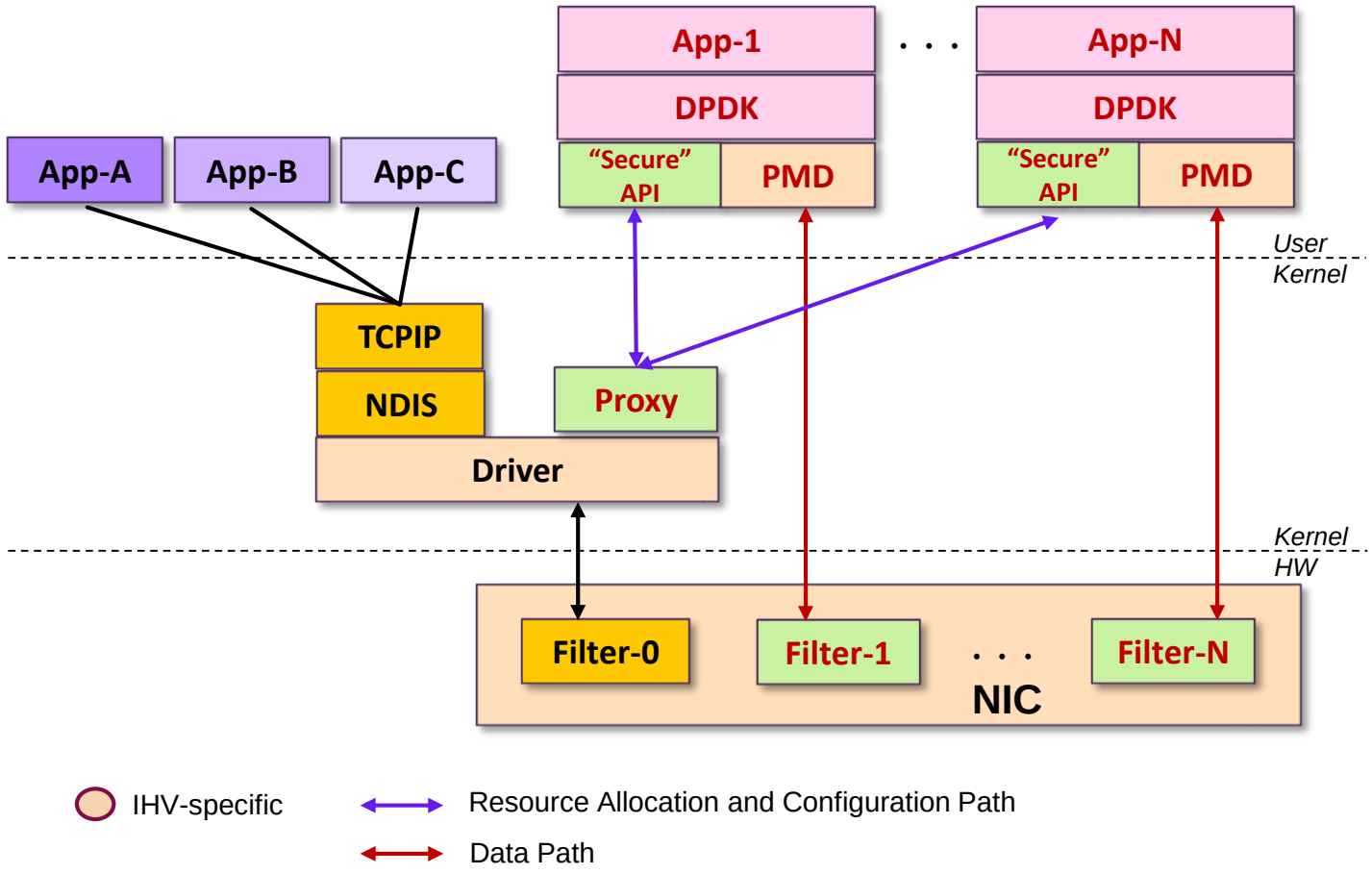


Benefits of new architecture

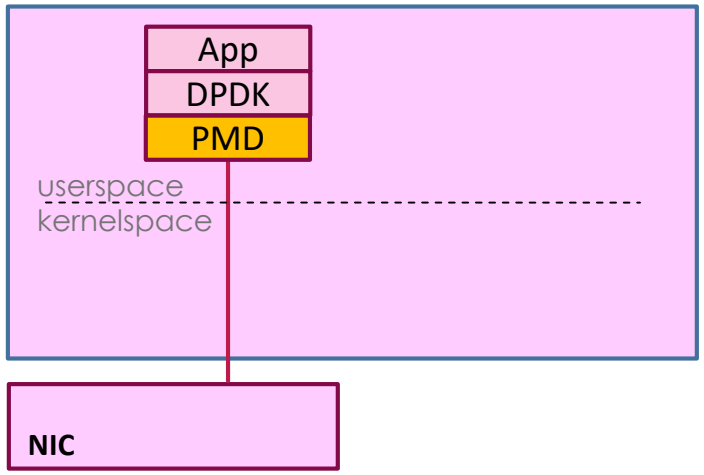
- Memory/resource allocation in Kernel driver
- Security enforced with proxy in the kernel driver
- Can filter flows to a particular filter through existing mechanisms – mac, VLAN, mac-VLAN, IP filtering etc.
- Kernel driver can be fully certified as it is done today
- No UIO driver required

“Secure” API interface

- Device-agnostic interface
- OS-agnostic interface
- Per user/process configuration
- Compartmentalize resources

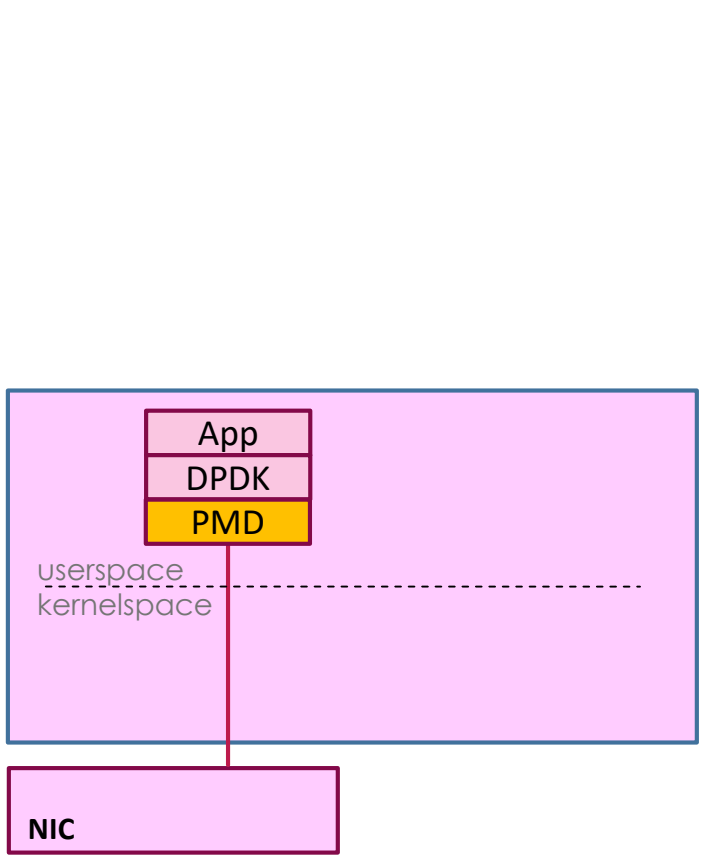


Scope of Trust

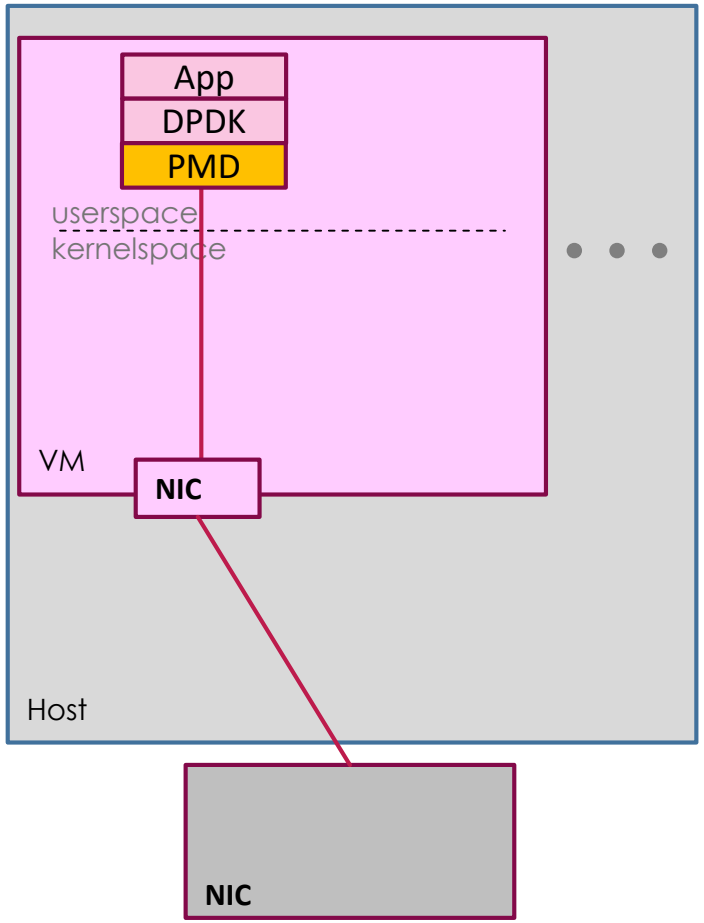


Physical Machine Scope

Scope of Trust

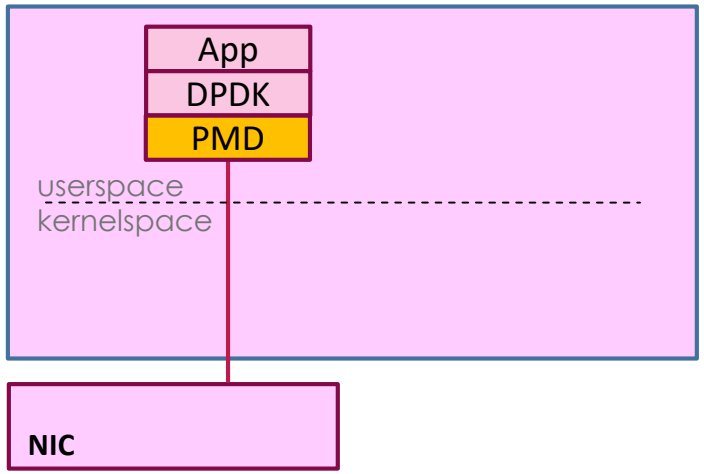


Physical Machine Scope

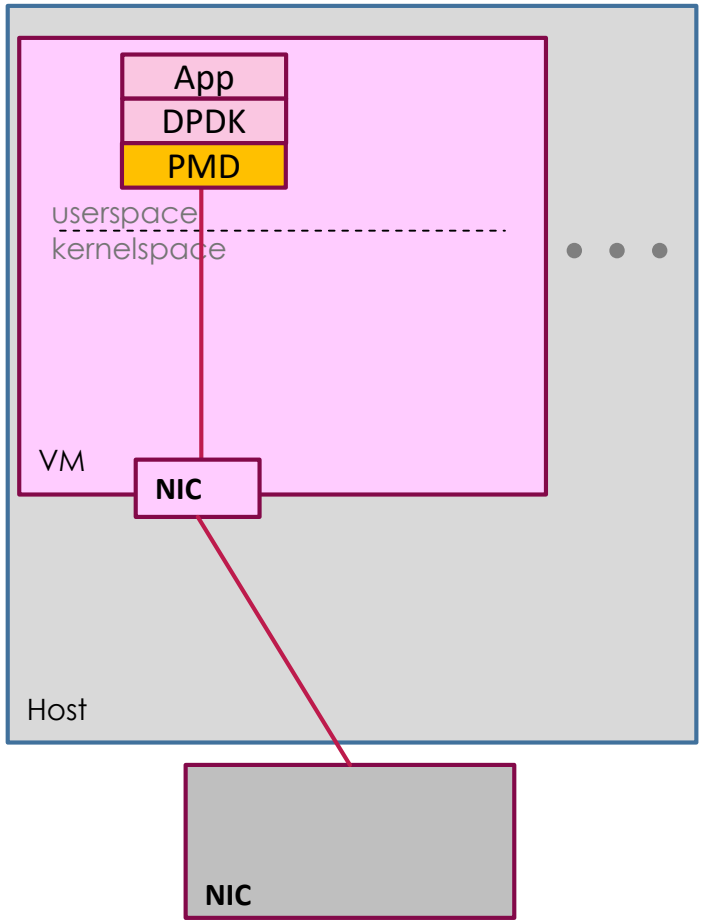


Virtual Machine Scope

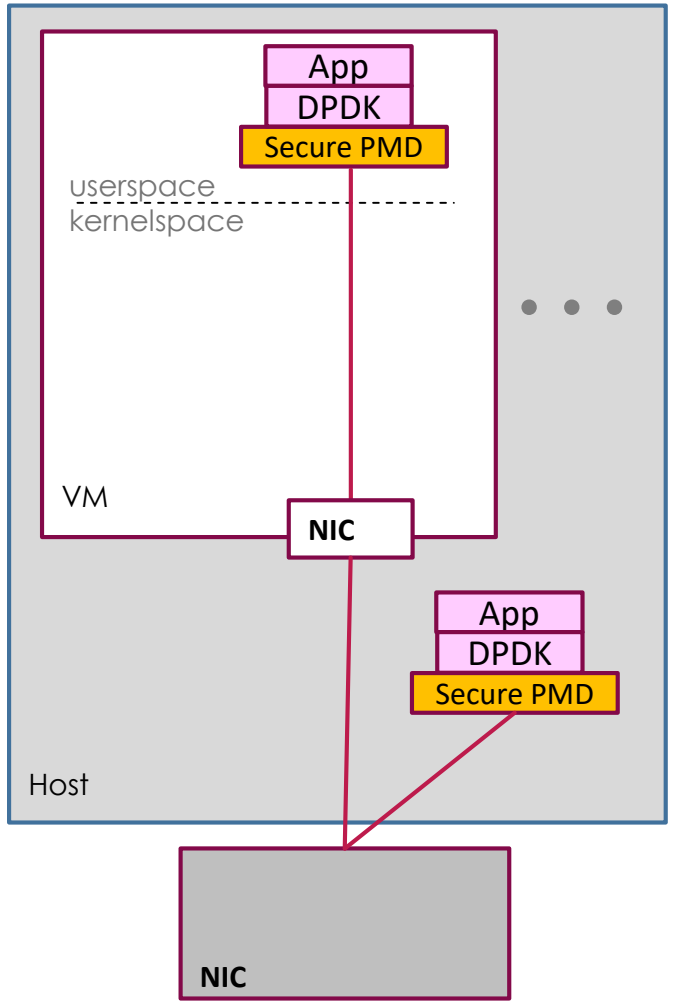
Scope of Trust



Physical Machine Scope



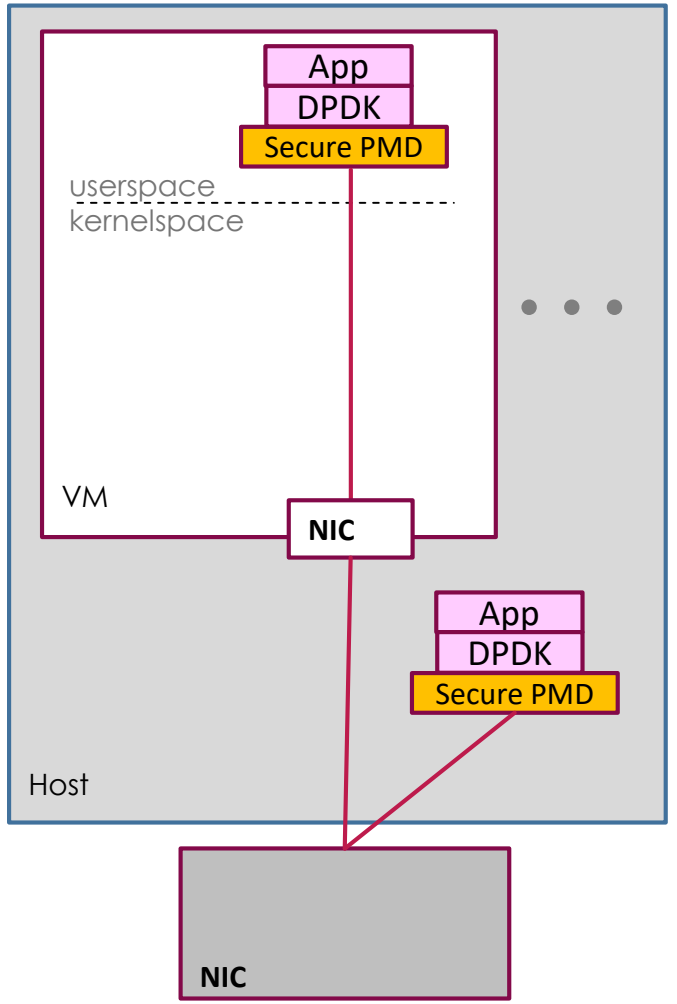
Virtual Machine Scope



Application Instance Scope

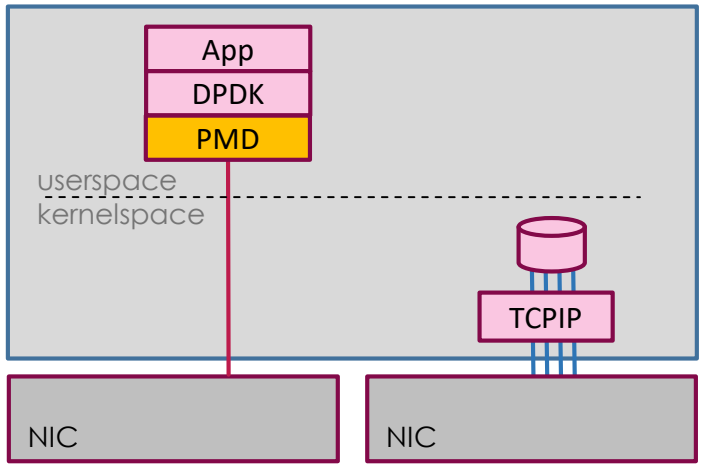
Multi-process / Multi-user security

- User space registered memory
  - Address, Length, Key - *MMU enforced
- HW Agnostic Kernel space Control Path visibility
  - Challenges with low-end vs high-end device and capabilities
  - IOT vs Server
- Per user/process resource caps and reservations
  - Shape and control QP, CQ, MR, and associated HW resource consumption
- Kernel space Network Diagnostics and Monitoring
  - Operationalize!
  - Target First Failure Data Capture



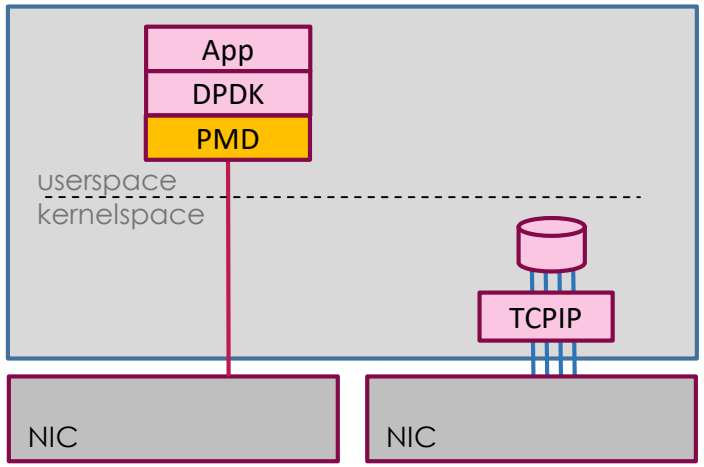
Application Instance Scope

Multi-tenancy security

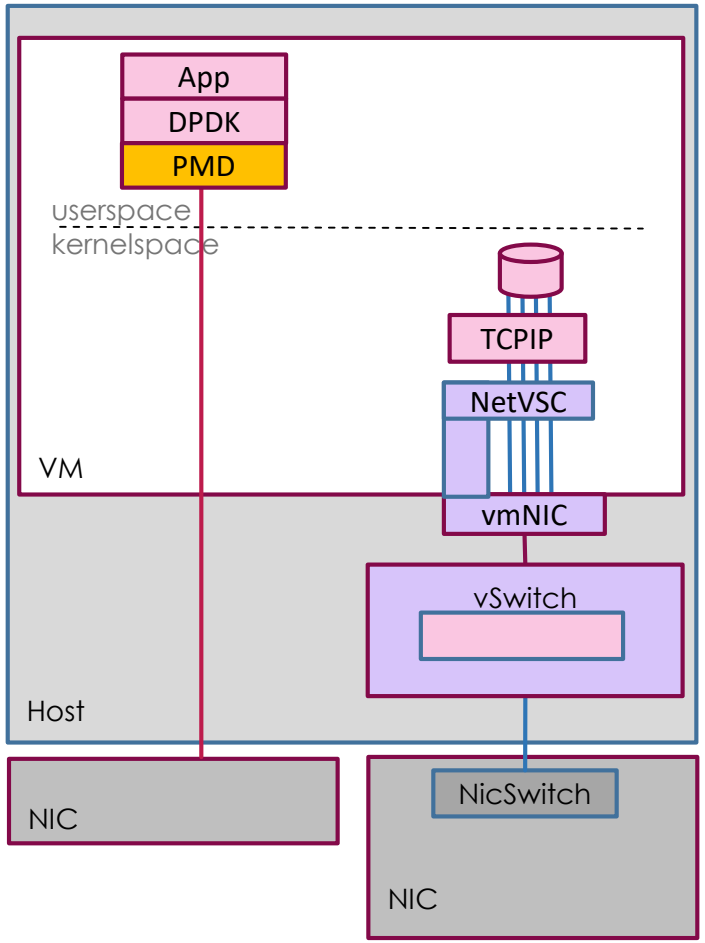


Native

Multi-tenancy security

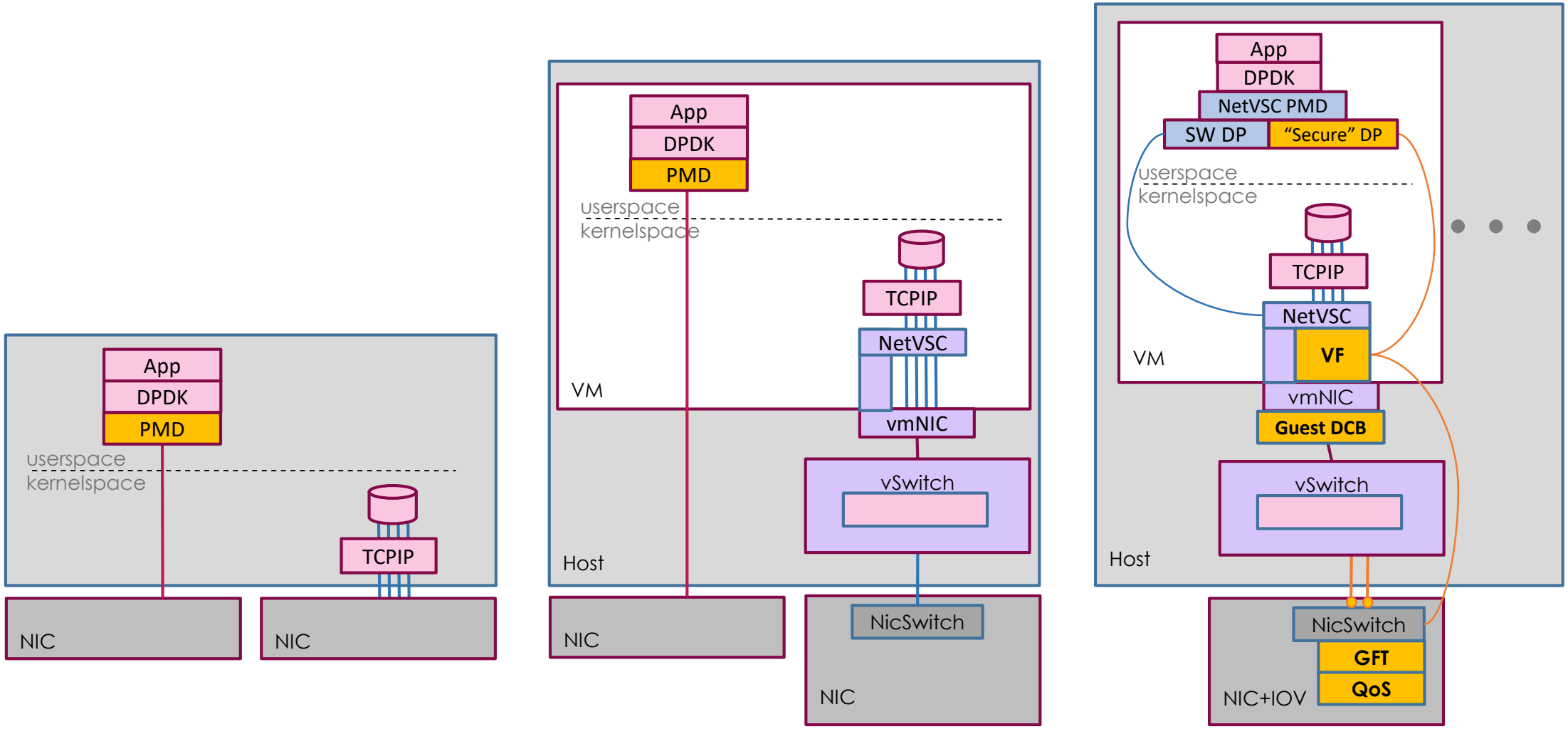


Native



DDA – Direct Device Assignment

Multi-tenancy security



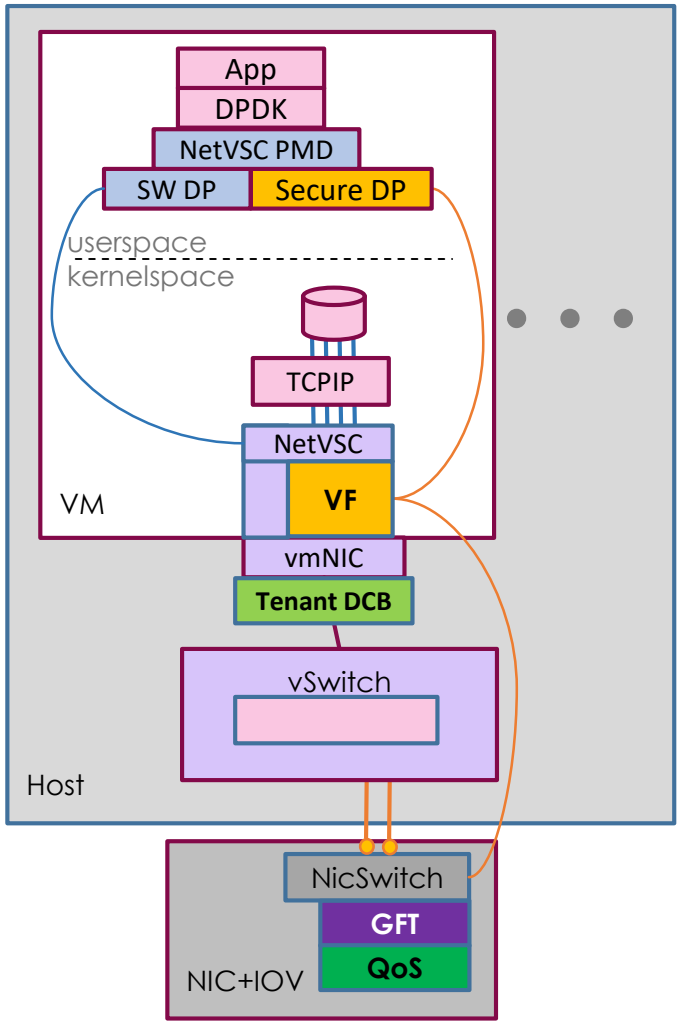
Native

DDA – Direct Device Assignment

Multi-Tenancy

Multi-tenancy security

- Performance and Security conflict
  - VFs bypass security… Fabric compromised…
  - Acceptable for trusted Guests

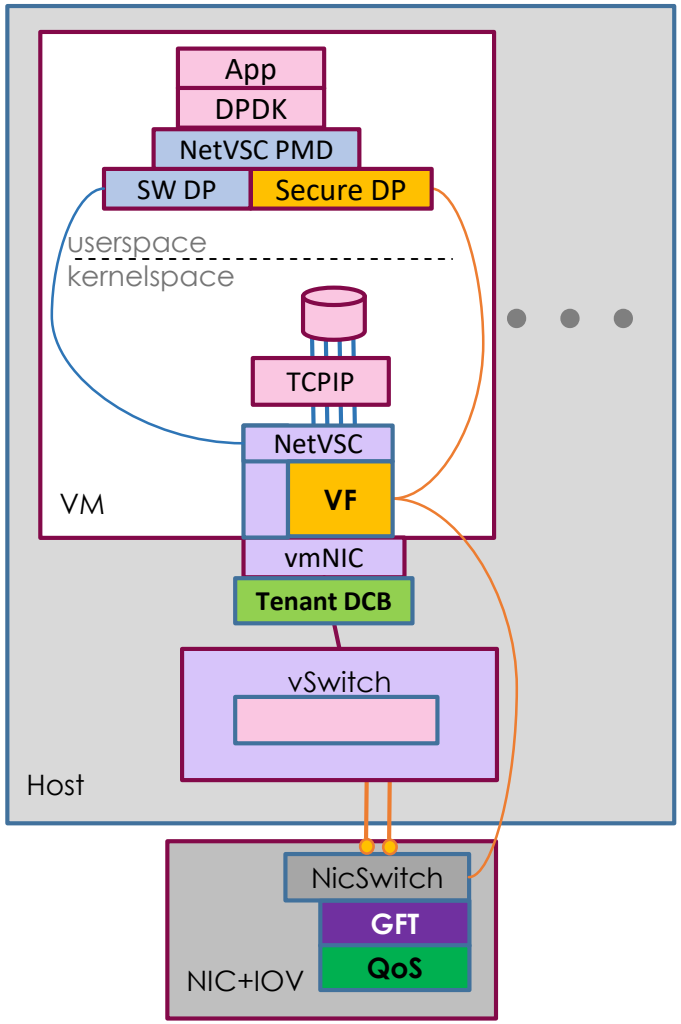


Multi-Tenancy

Multi-tenancy security

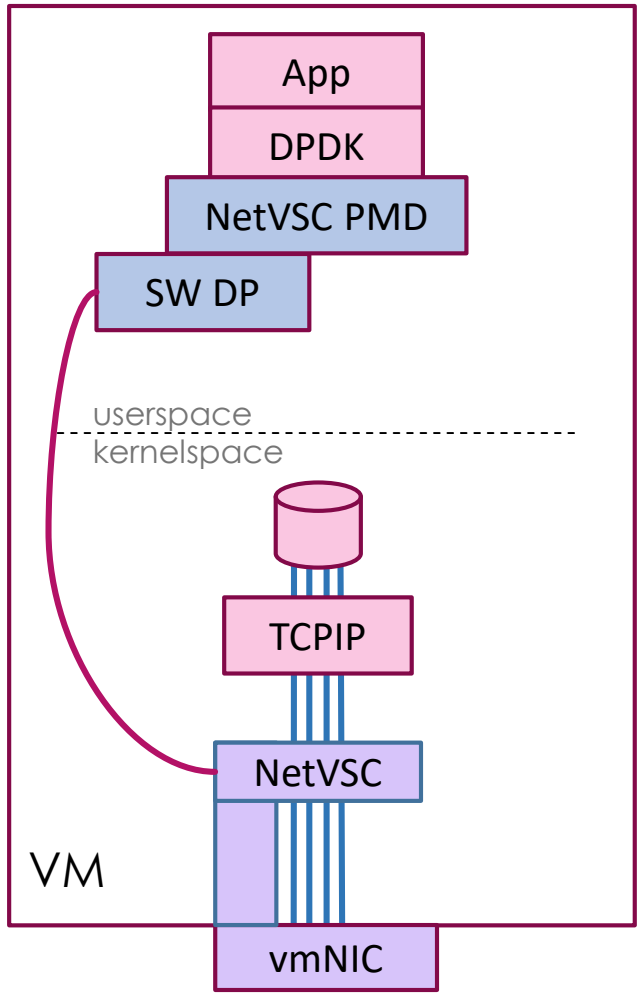
- Performance and Security conflict
  - VFs bypass security… Fabric compromised…
  - Acceptable for trusted Guests

- How can we secure tenants?
  - (1) Control what tenant places on the fabric
    - GFT – Generic Flow Tables
      - Parse, Push/Pop, Transpose…
    - Tenant DCB
      - VF level conversion
      - Automatic DCB correction
  - (2) Control how much tenant places on the fabric
    - Per-TC HW QoS
    - Send: Caps/Reservations. Recv: Caps
  - (3) Control what HW resources tenant consumes
    - VF Resource Caps (QP, CQ, PD, MR, etc.)



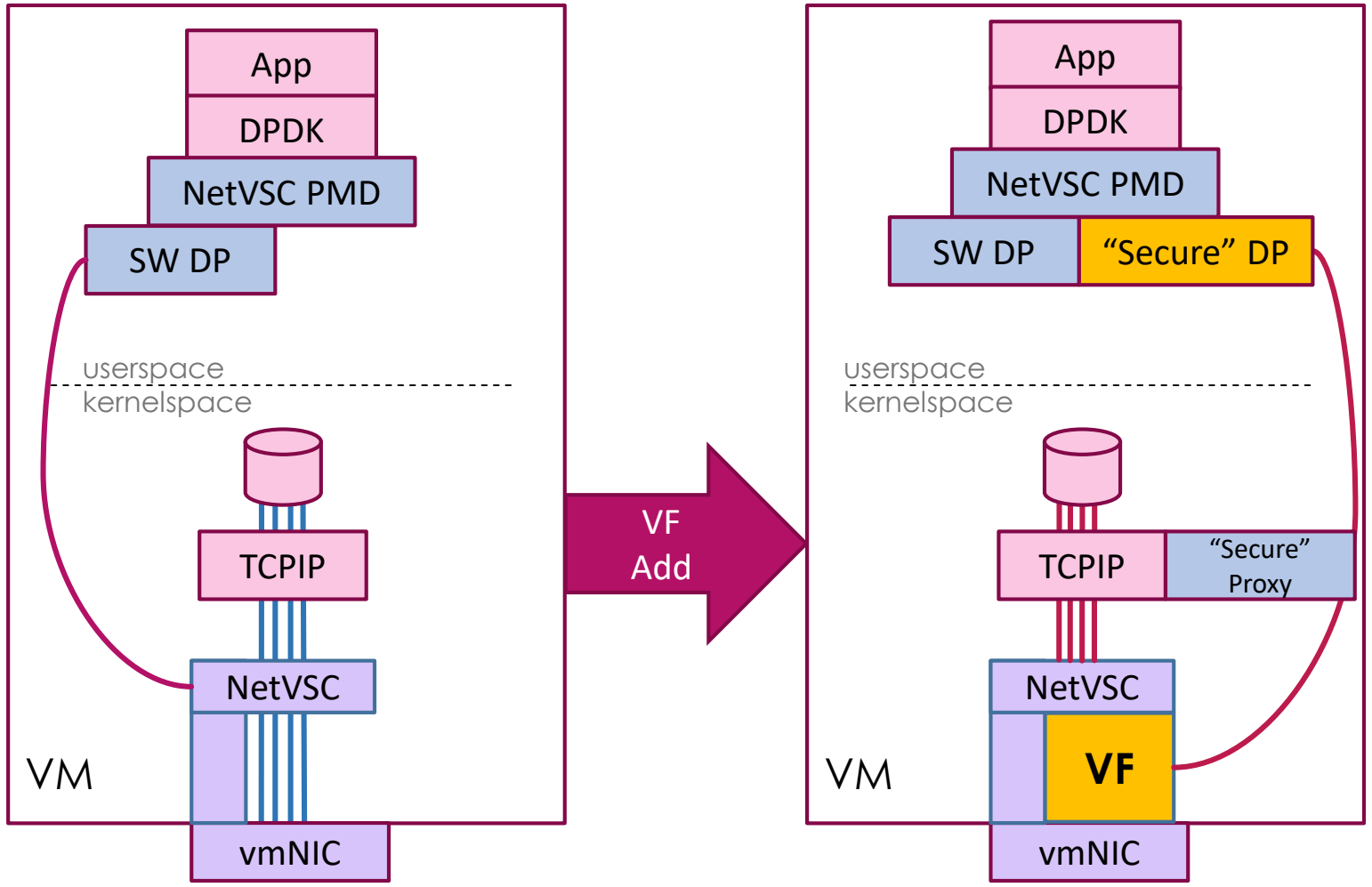
Multi-Tenancy

Availability



No VF

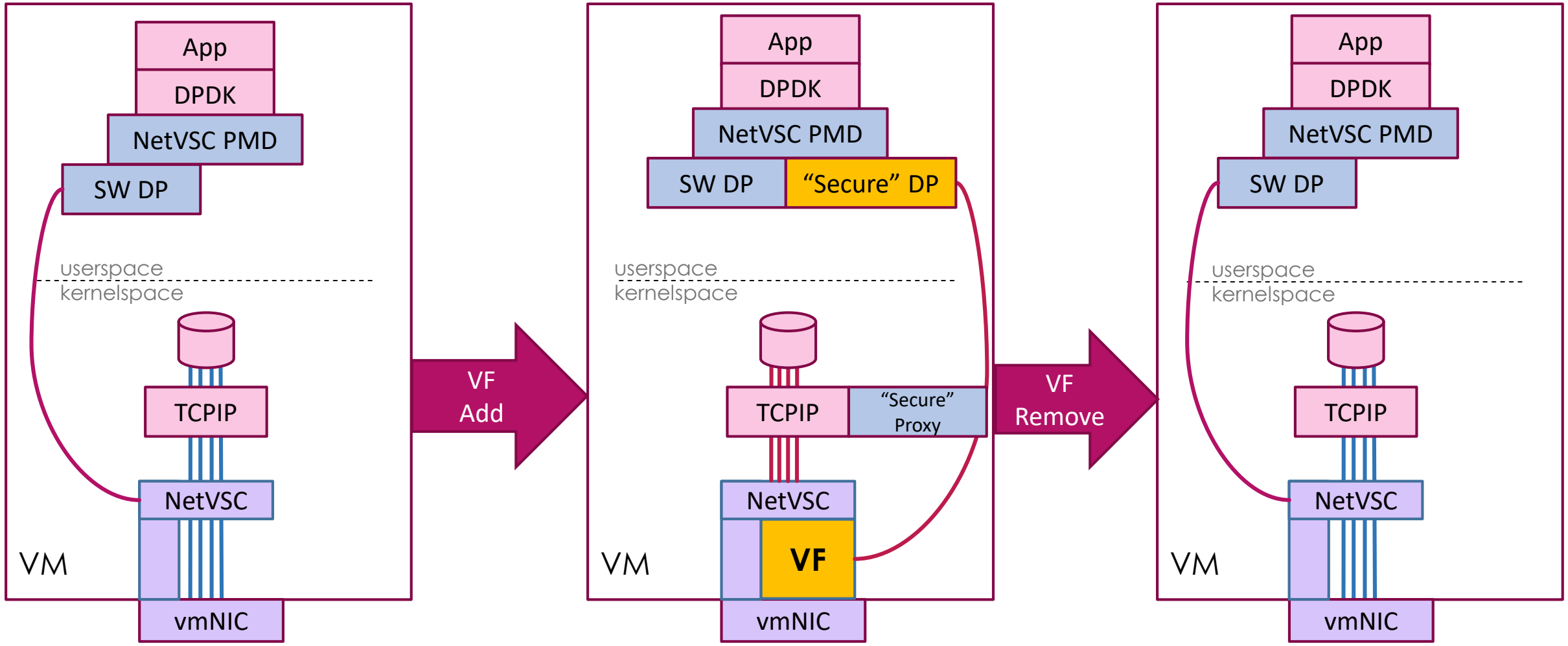
Availability



No VF

Dynamically Add VF

Availability



No VF

Dynamically Add VF

Dynamically Remove VF

Areas of investigation

- AF_XDP
  - Interesting approach for flexible SW -> HW flow steering and user space DMA
  - Potential simplification to synthetic slow path at Socket vs Device
- eBPF
  - Required to control ***what*** is placed on wire
  - Can potentially be used to offload GFT rules/transpositions (Secure IOV)
- Virtual IOMMU
  - Implementation feasibility vs leveraging the existing/supported ND security model

Call to Action

- Provide feedback on new model
- Download and use existing Windows support code from draft repo
- How to contribute:
  - https://core.dpdk.org/contribute/
  - Reference “*dpdk-draft-windows*” in contribution

- Help us make it better!