[Public]

vDPA-net Live Migration with Shadow VirtQueue

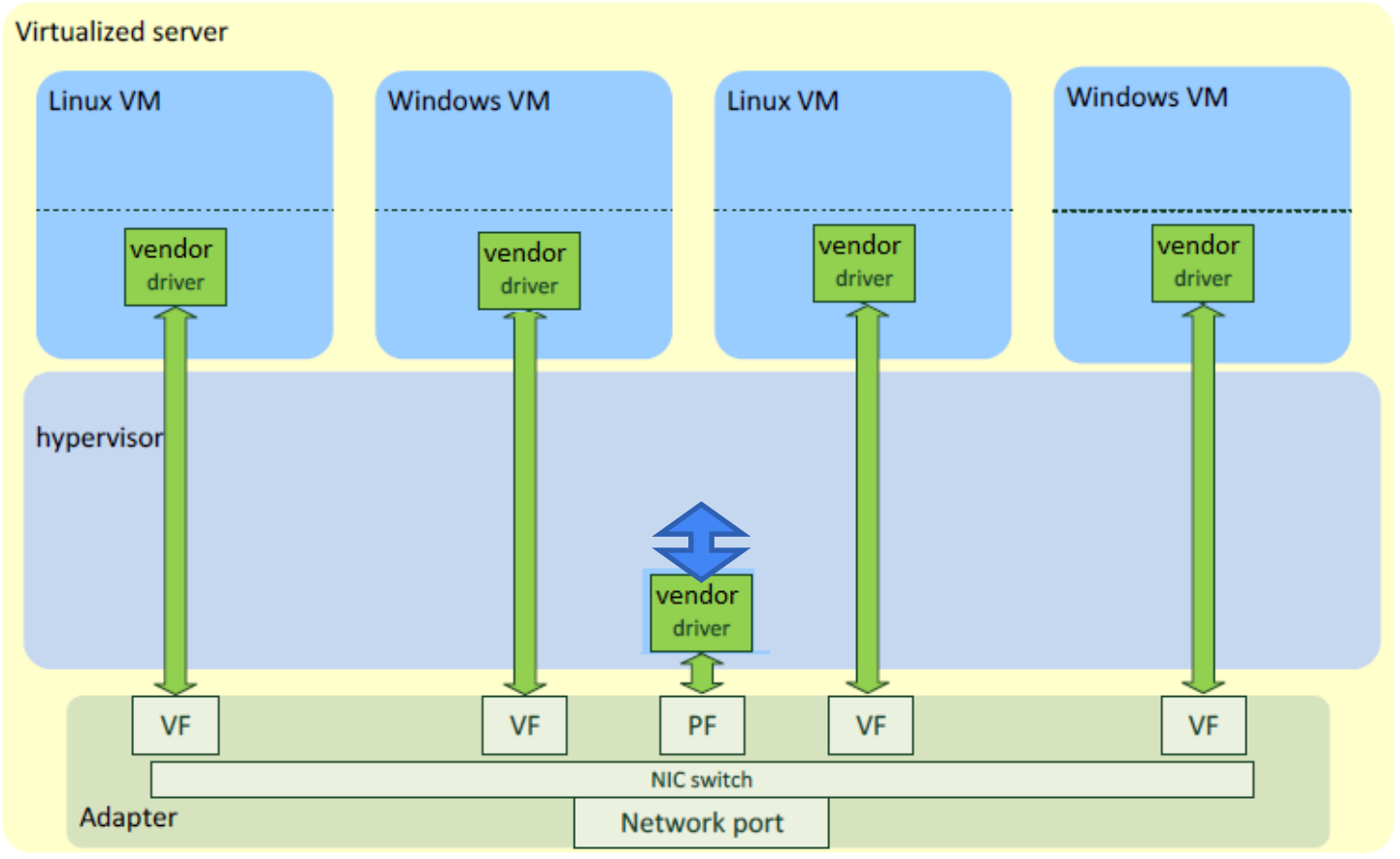
Eugenio Perez Martin
Sr. Software Engineer <eperezma@redhat.com>

Gautam Dawar
Sr. Manager Software Development <gdawar@amd.com>

Agenda

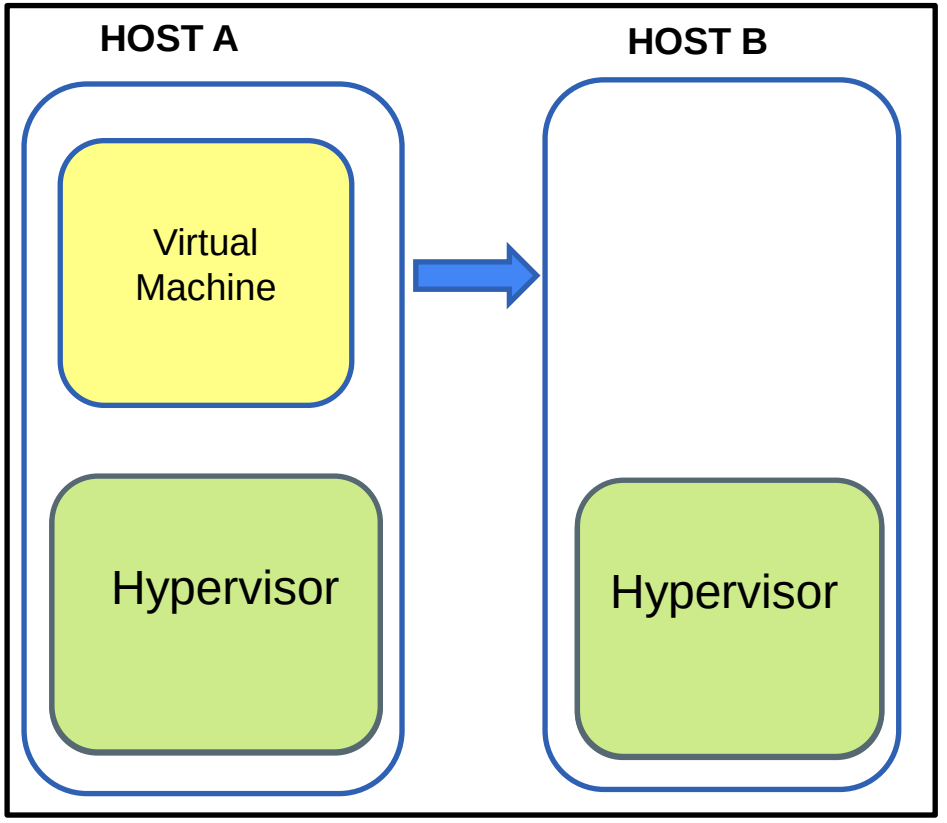
- SR-IOV
- Live Migration
- Problem: LM with passthrough VF
- Solution: vDPA
- Cross-vendor VM Live Migration Demo
- Shadow virtqueue operation
- Q&A

SR-IOV



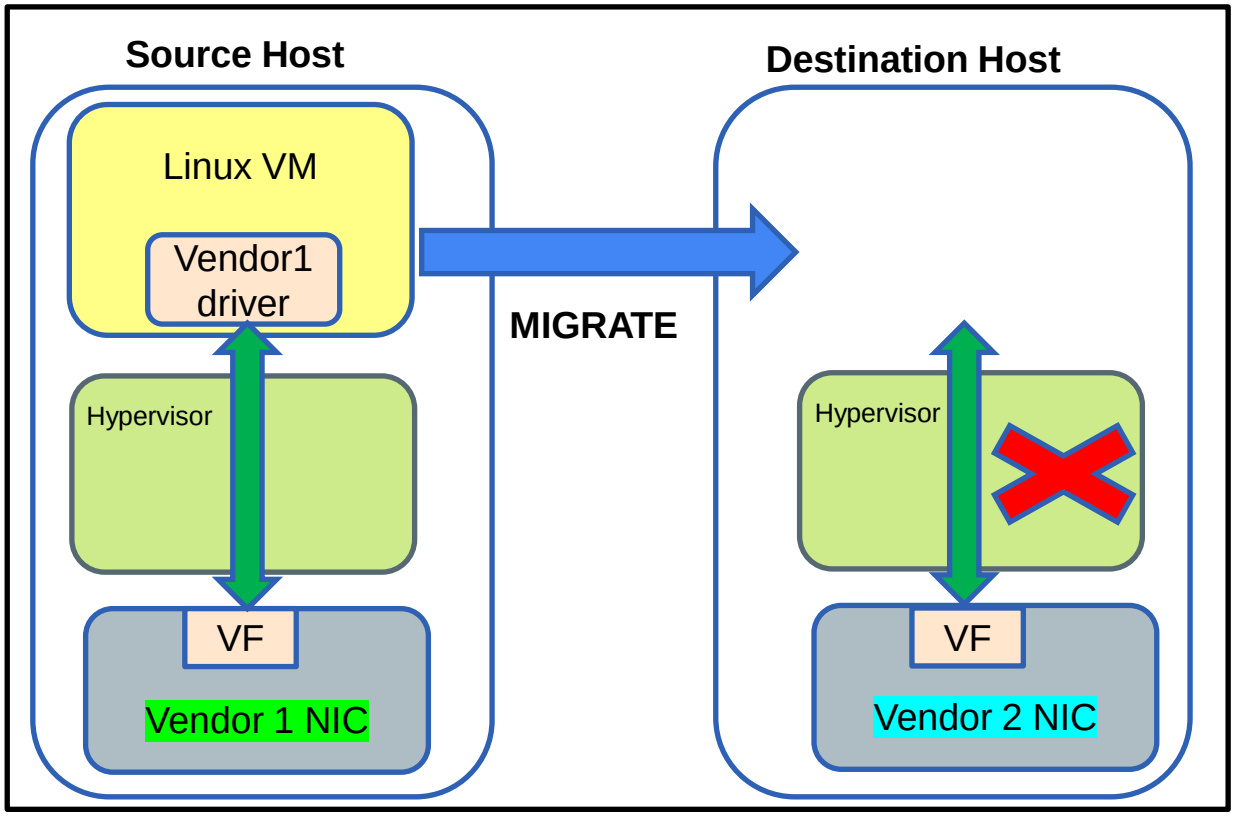
VM Live Migration

- What is “Live” Migration?
  - Process of moving a VM running on one physical host to another while the guest OS is **running**
  - Useful for load balancing, hardware/software maintenance, etc.
- How does it happen?
  - Marking modified RAM pages as “**dirty**”
  - **Sending** these dirty RAM pages to the destination until a *threshold* is reached
  - Stop guest, **transfer remaining** dirty RAM, device state
  - **Resume execution** on destination



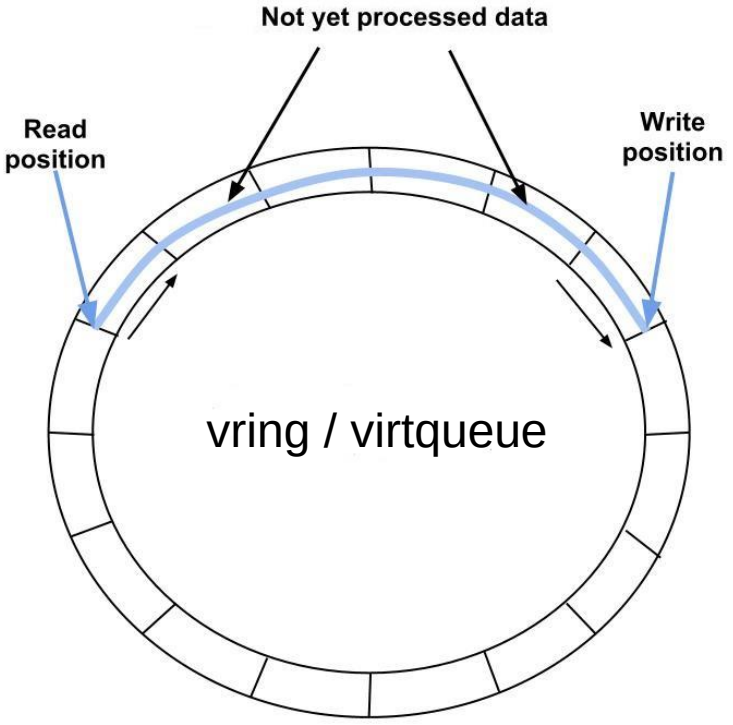
Live Migration: SR-IOV VF Passthrough

- Requires identical NIC HW on both source and destination host
  - Tight coupling between the Guest SW and Host HW
  - Vendor's VF driver required in the Guest OS

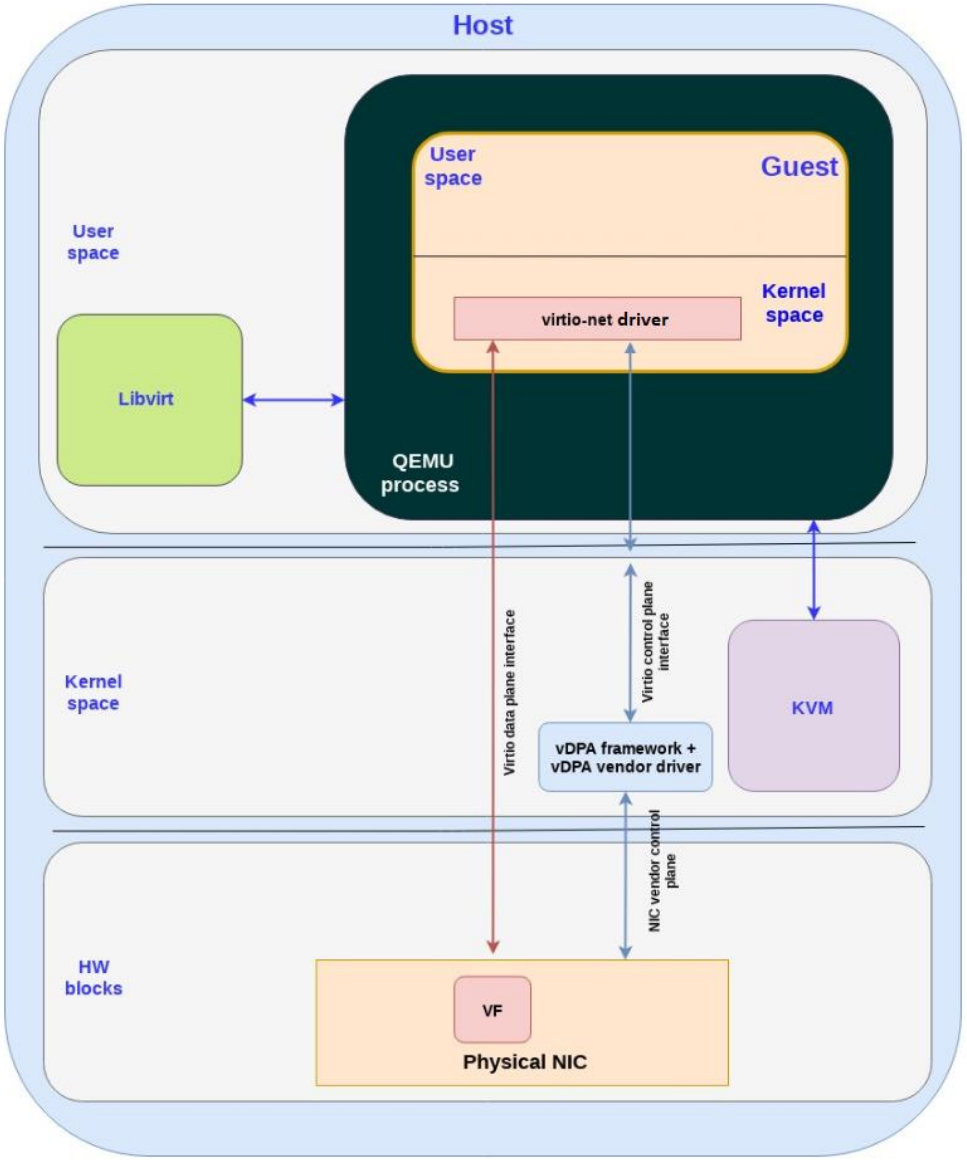


Virtual I/O Device (VIRTIO)

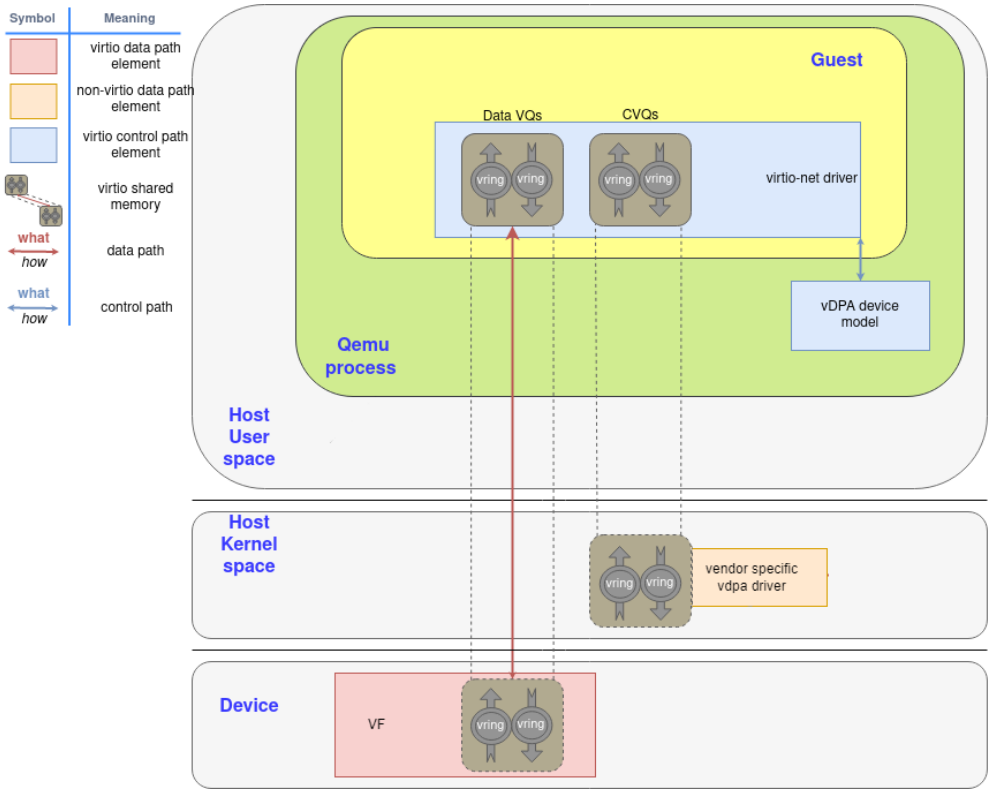
- Virtio is a virtualization specification that describes virtual devices, drivers and how they interact.
- Virtio spec defines how to create a control plane and the data plane between the guest and host.
- **Data plane**
  - Composed of buffers and rings layouts
  - Used for transferring the actual (bulk) data (packets) between host and guest
- **Control plane**
  - For establishing and terminating the data plane.
  - Feature negotiation, vring configuration, etc.



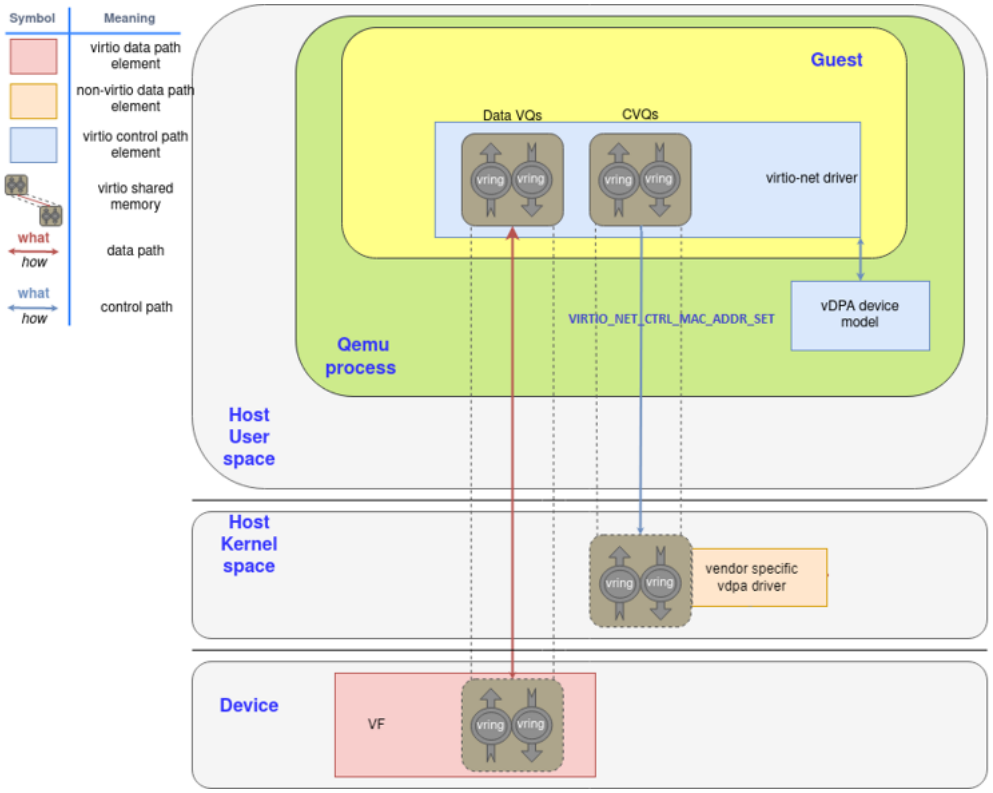
vDPA


Host
User space
Guest
User space
Kernel space
virtio-net driver
Libvirt
QEMU process
Kernel space
Virtio data plane interface
Virtio control plane interface
KVM
vDPA framework + vDPA vendor driver
NIC vendor control plane
HW blocks
VF
Physical NIC

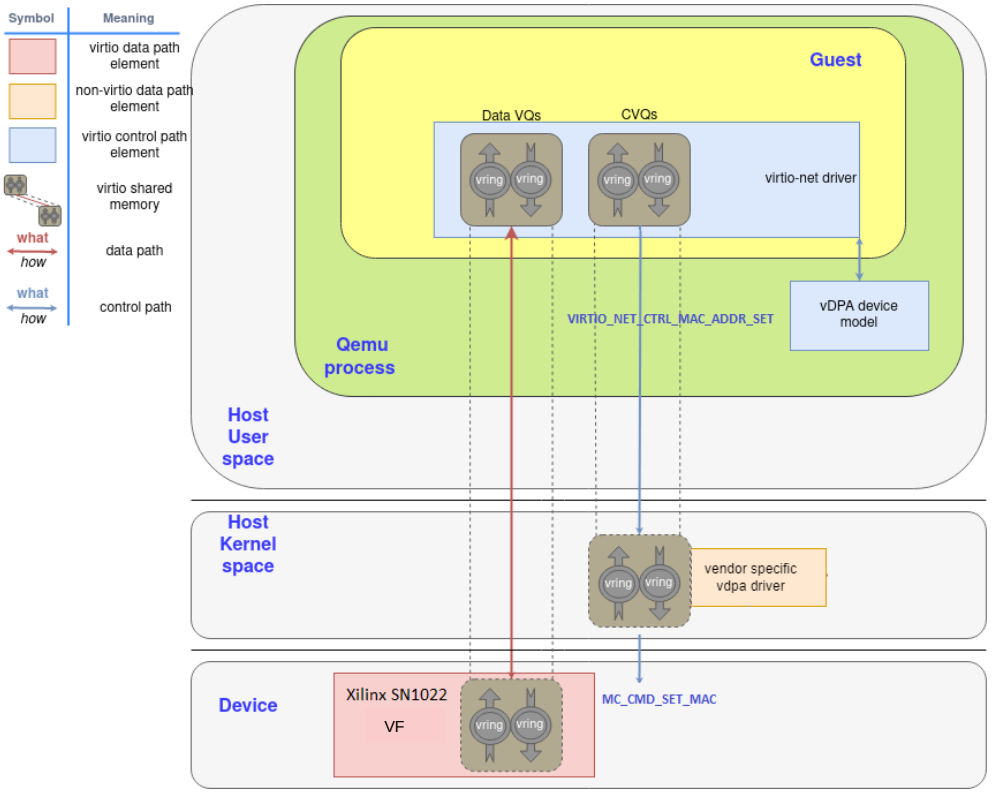
vDPA dataplane & control plane



vDPA dataplane & control plane

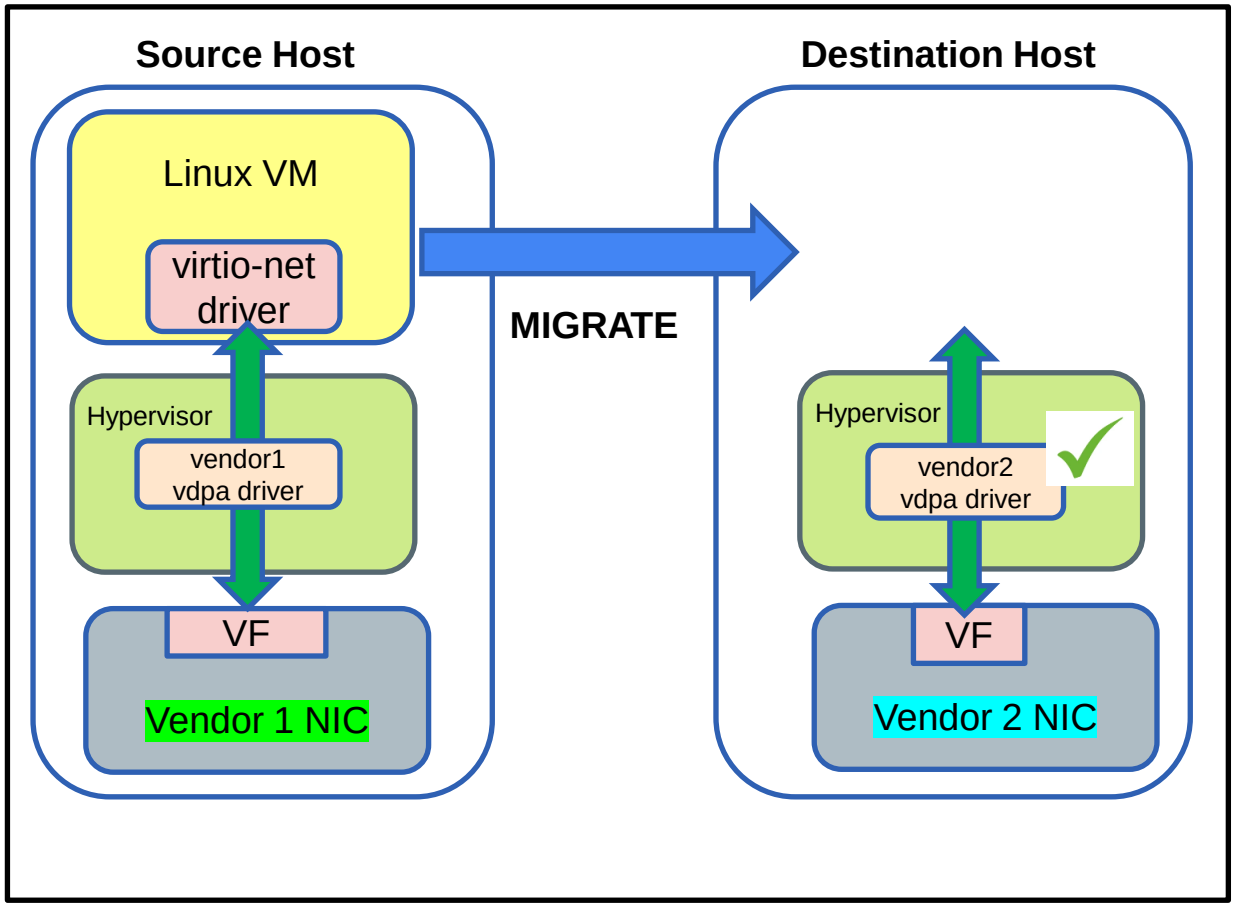


vDPA dataplane & control plane



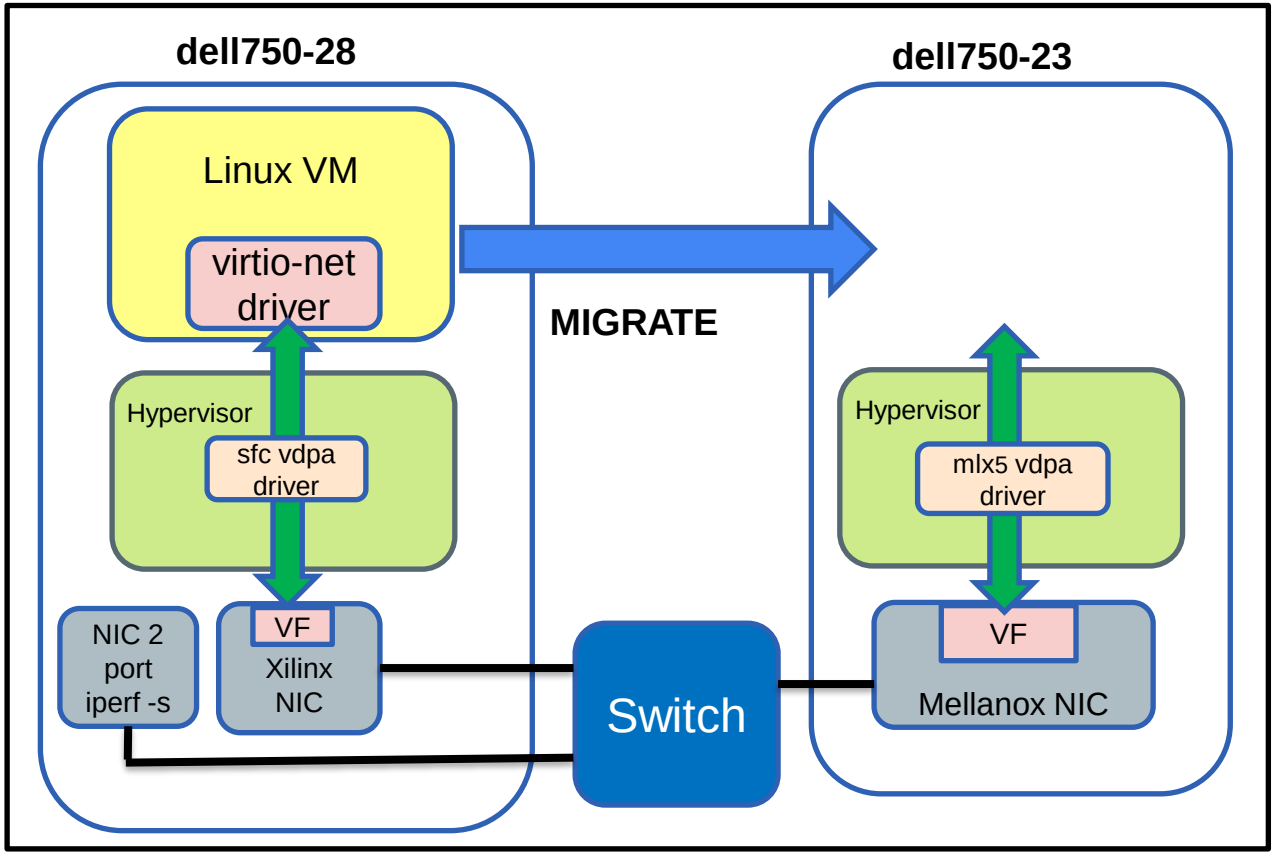
Live Migration with vDPA

- Live migration should be transparent to the guest
  - it only sees a virtio-net device, irrespective of actual vendor HW
  - Hypervisor doesn't require guest's collaboration



Live Migration - Setup

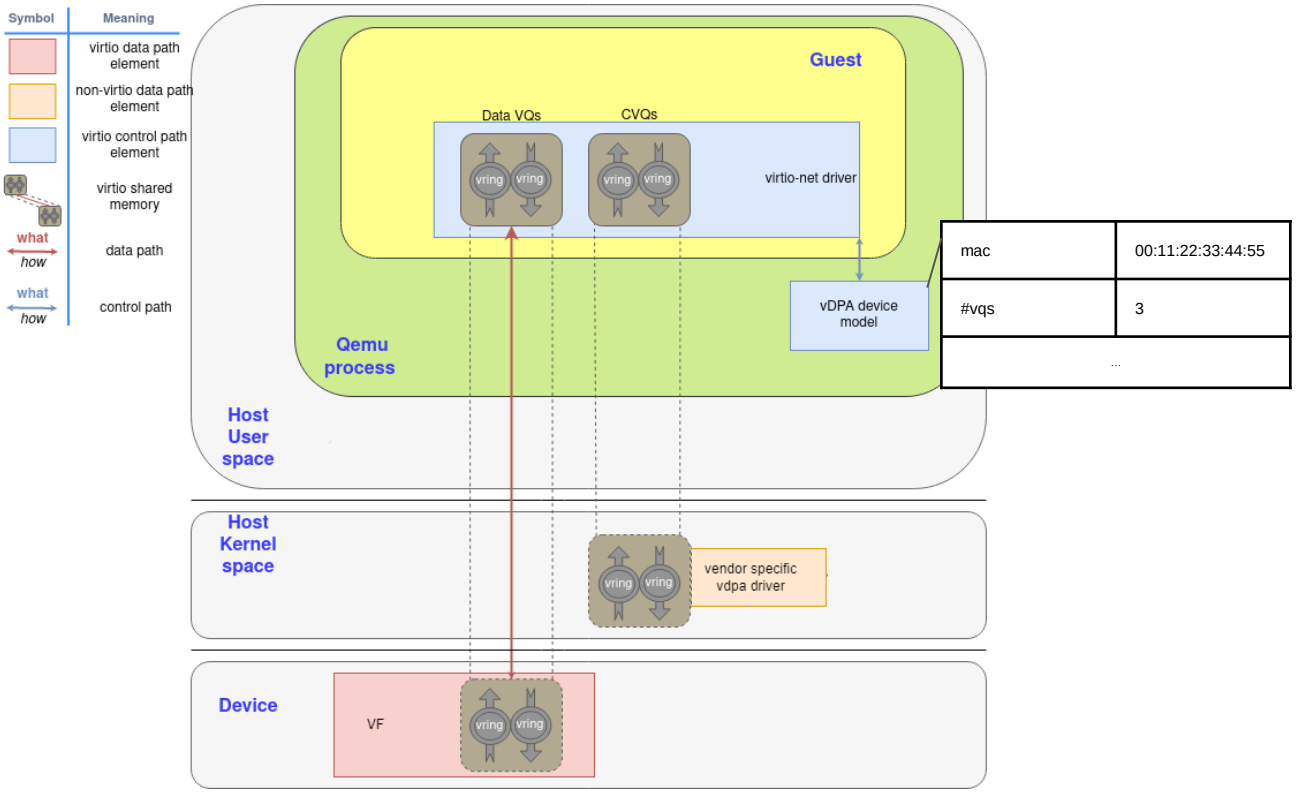
- Source host (dell750-28) has two NICs
  - **AMD Xilinx SN1022**
  - Physical port running iperf server
- Destination host (dell750-23) has single NIC
  - **Mellanox ConnectX 6**
- These NIC ports are connected via Switch and configured with *same VLAN ID*



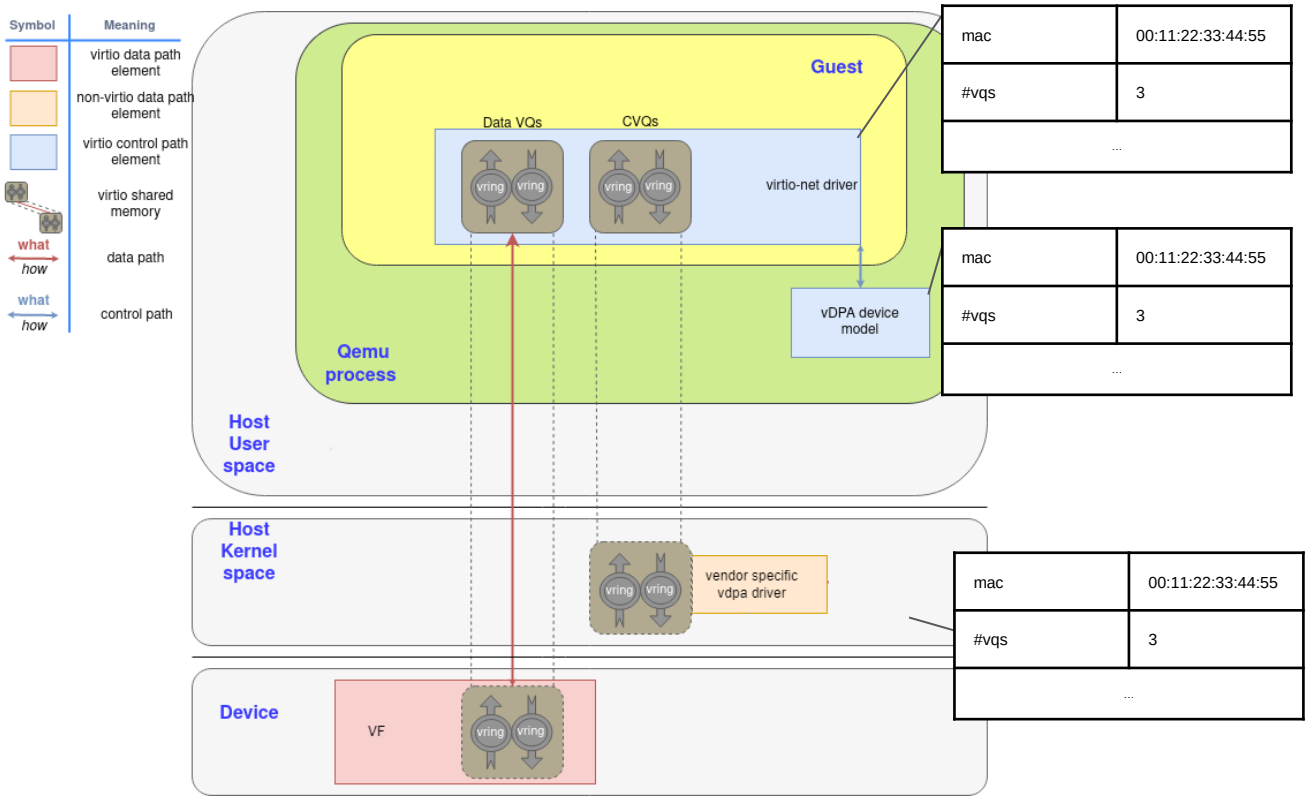
Demo

Watch it online here

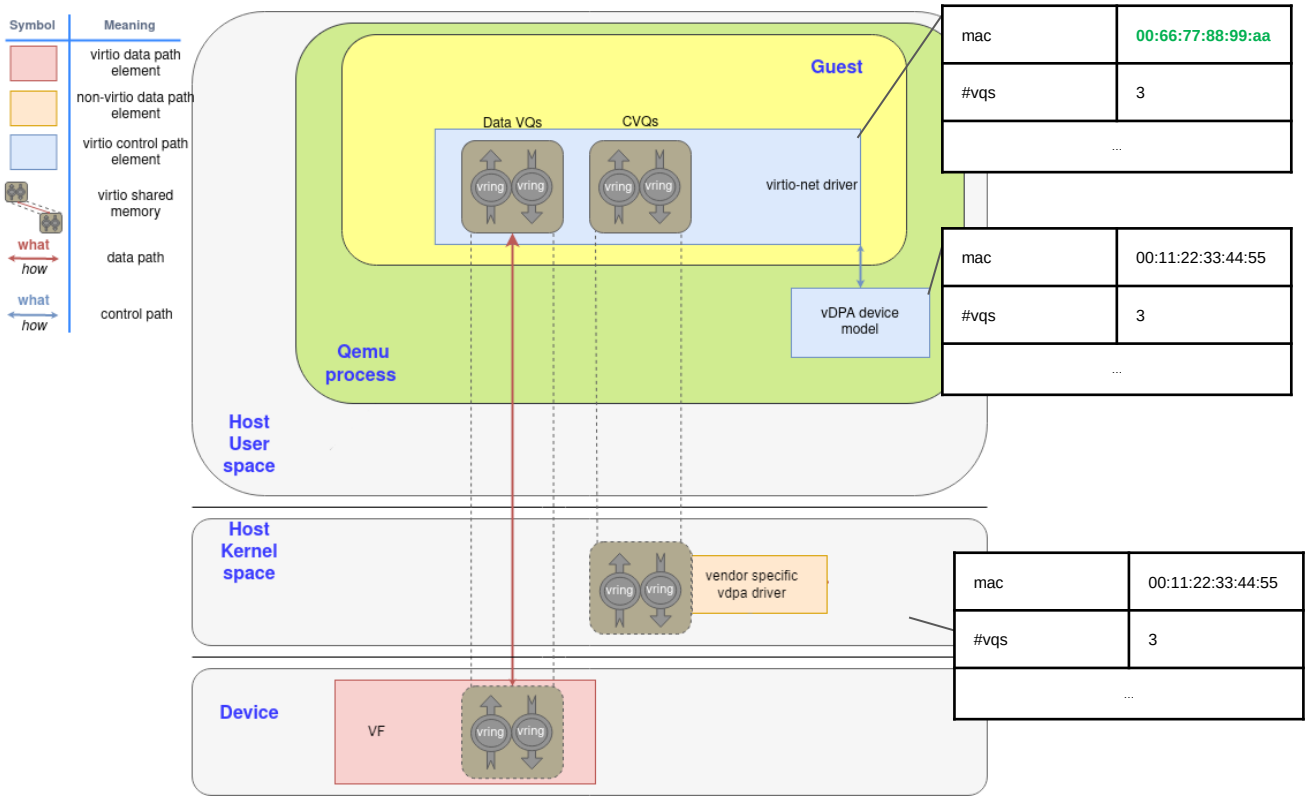
vDPA: CVQ bypassing QEMU



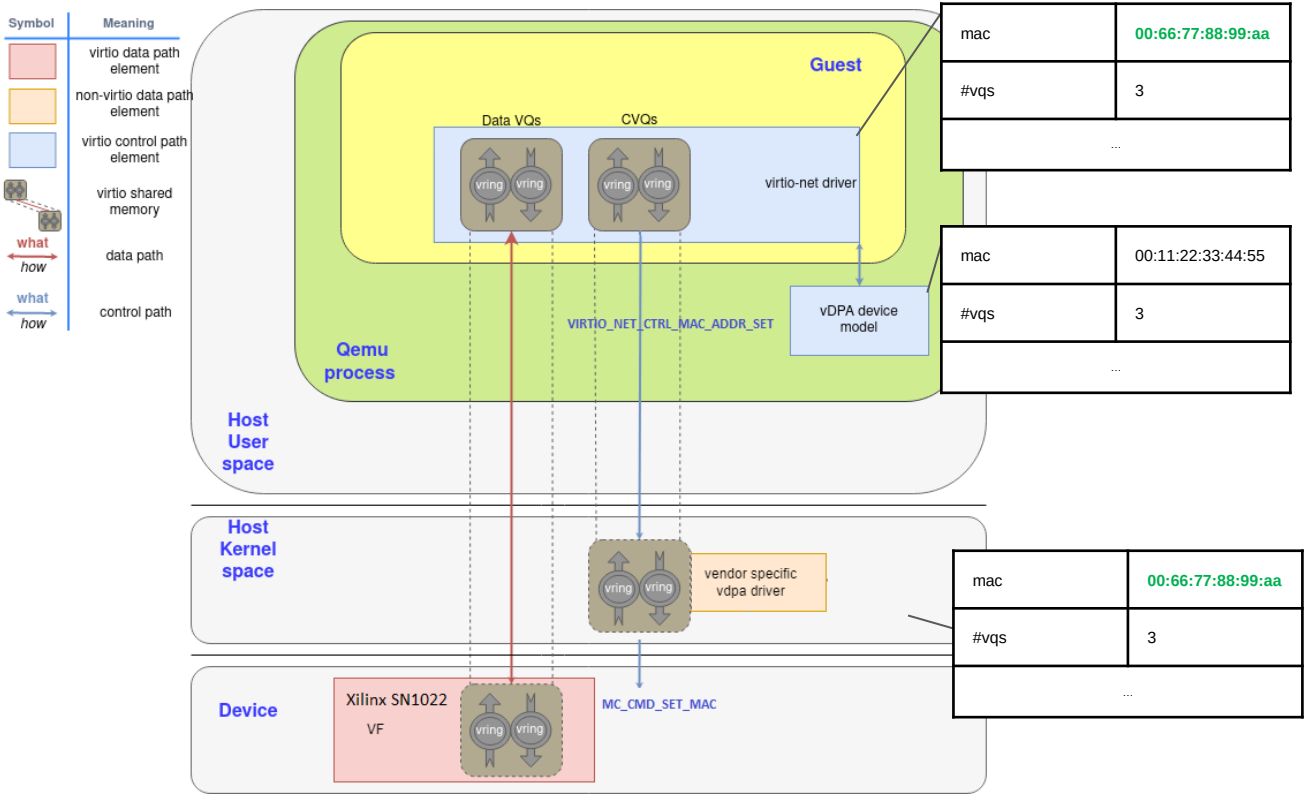
vDPA: CVQ bypassing QEMU



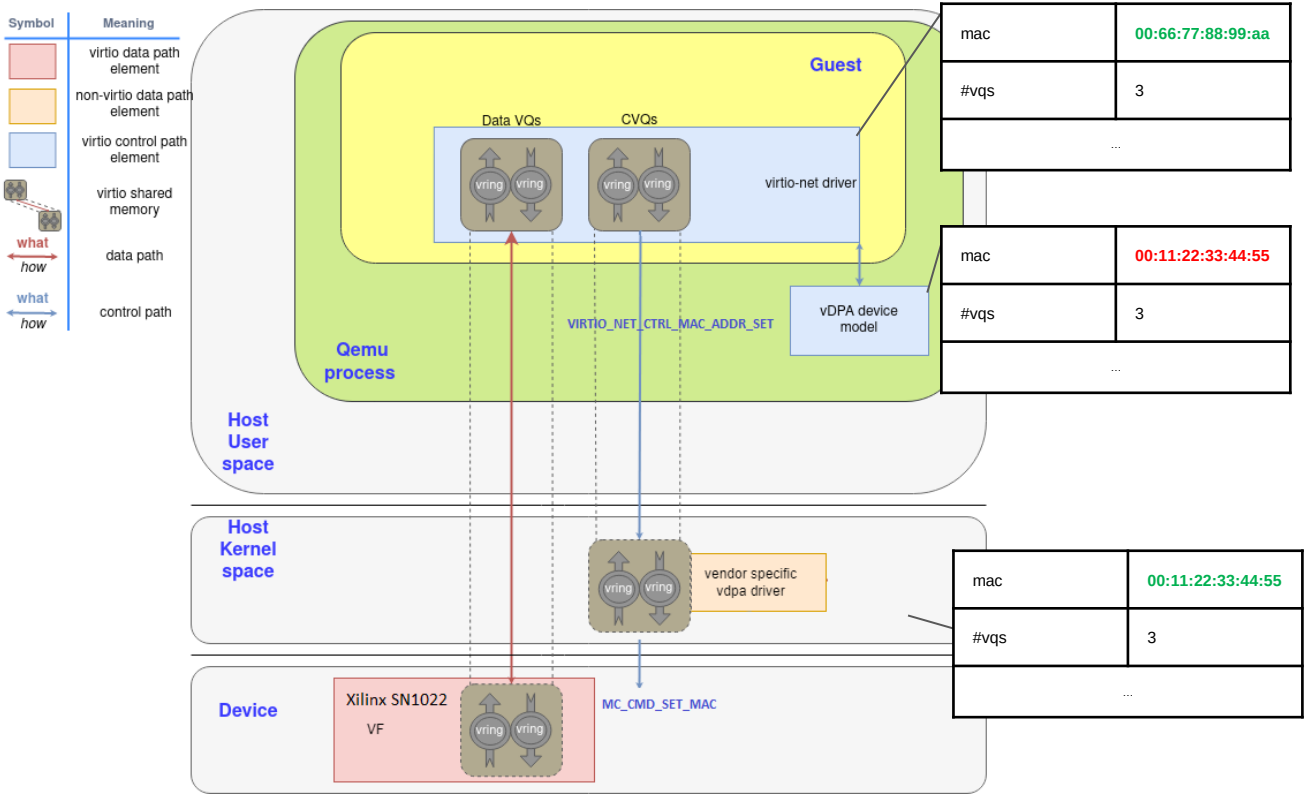
vDPA: CVQ bypassing QEMU



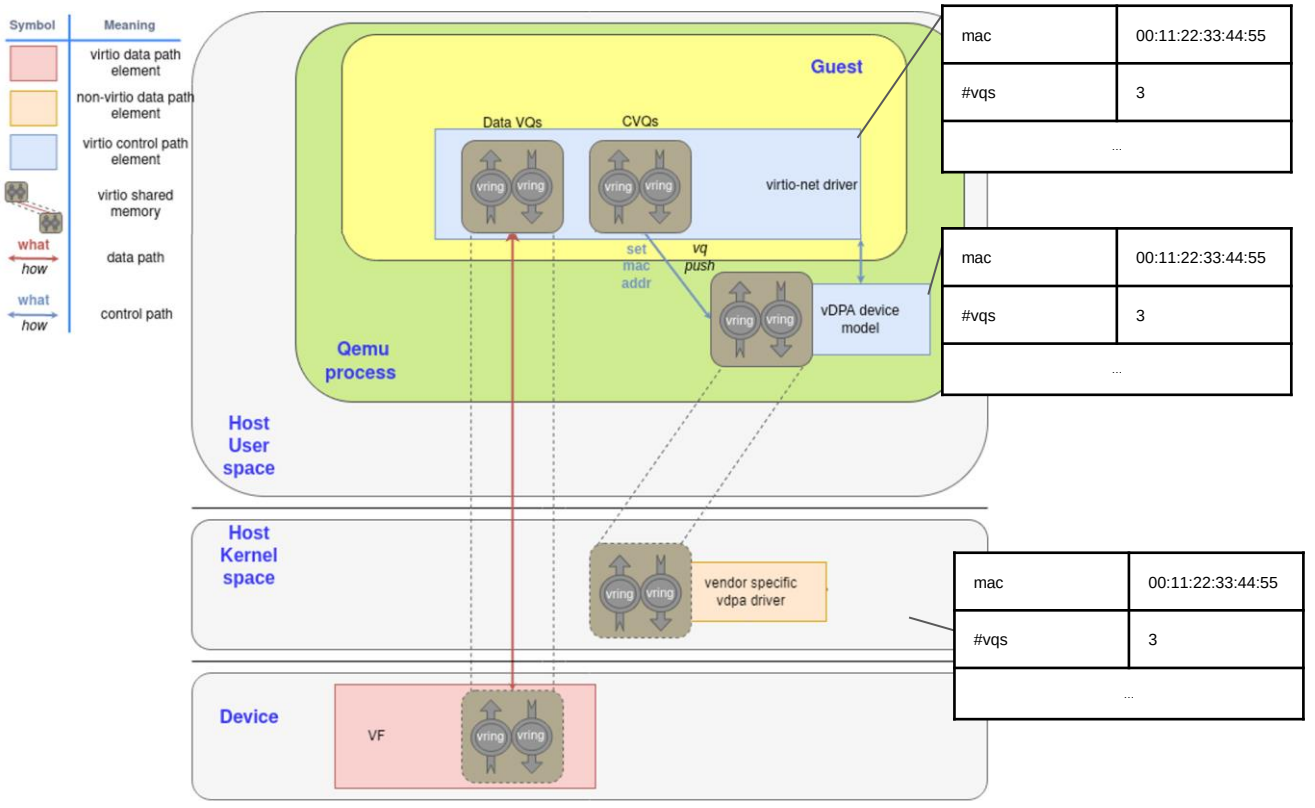
vDPA: CVQ bypassing QEMU



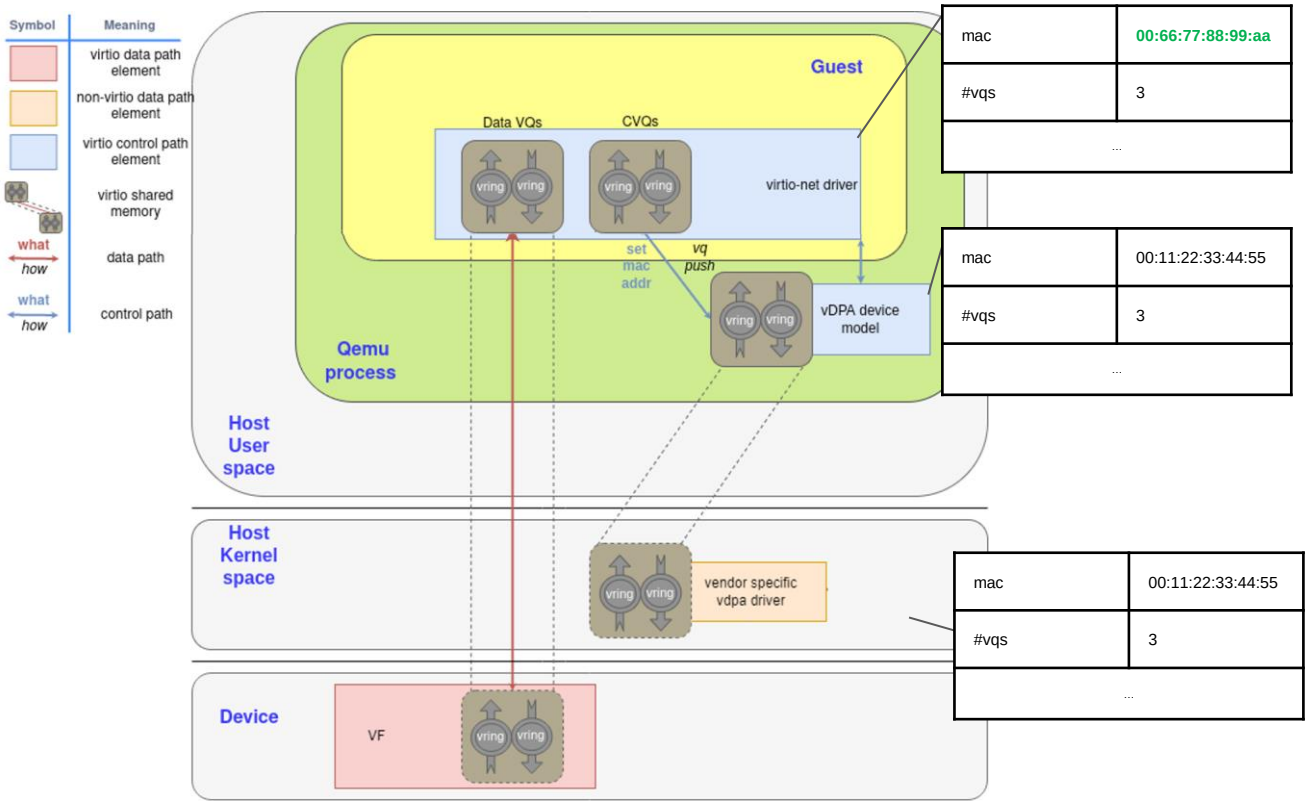
vDPA: CVQ bypassing QEMU



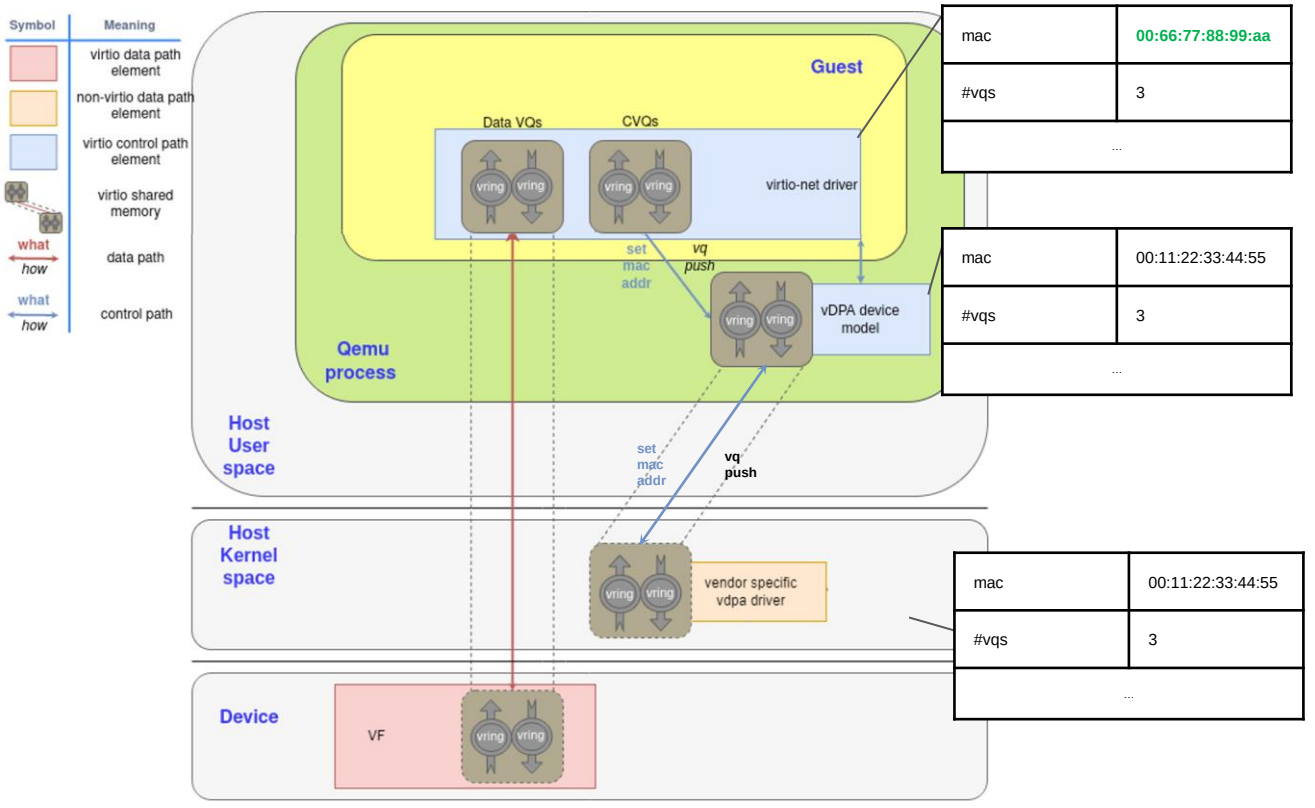
vDPA: shadowed CVQ



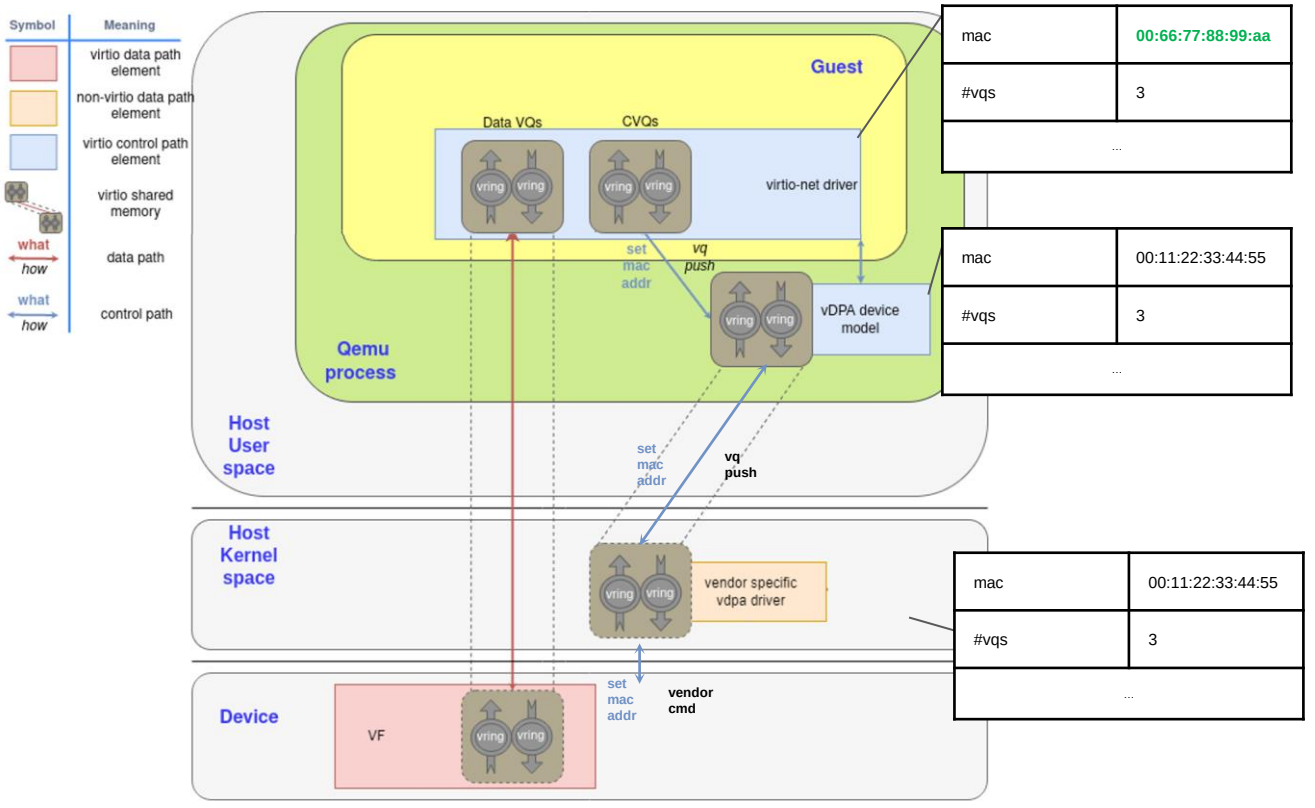
vDPA: shadowed CVQ



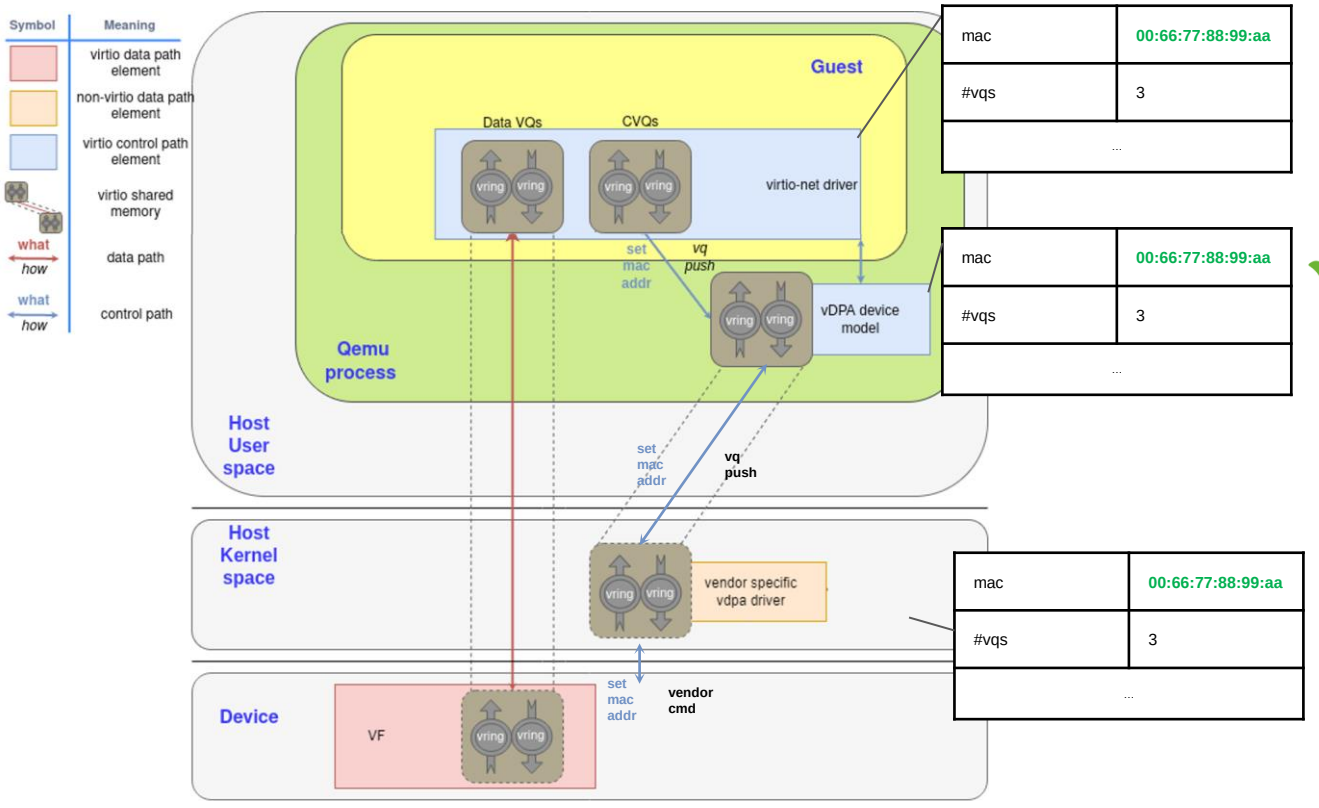
vDPA: shadowed CVQ



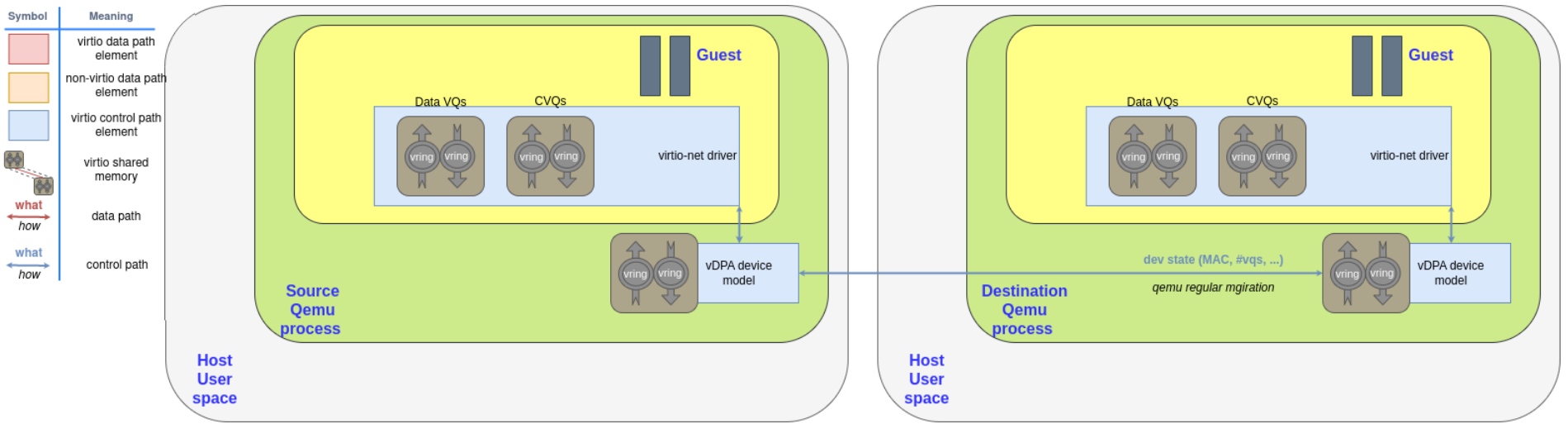
vDPA: shadowed CVQ



vDPA: shadowed CVQ

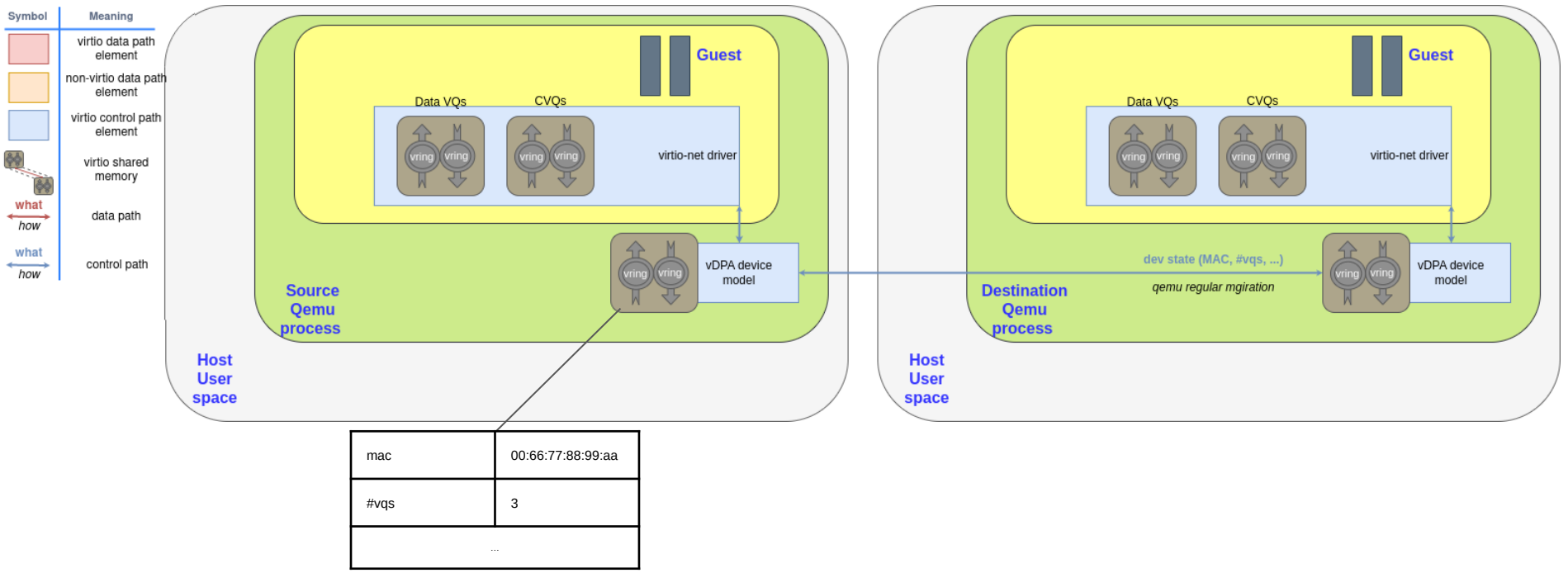


Shadow virtqueue: migrate device state

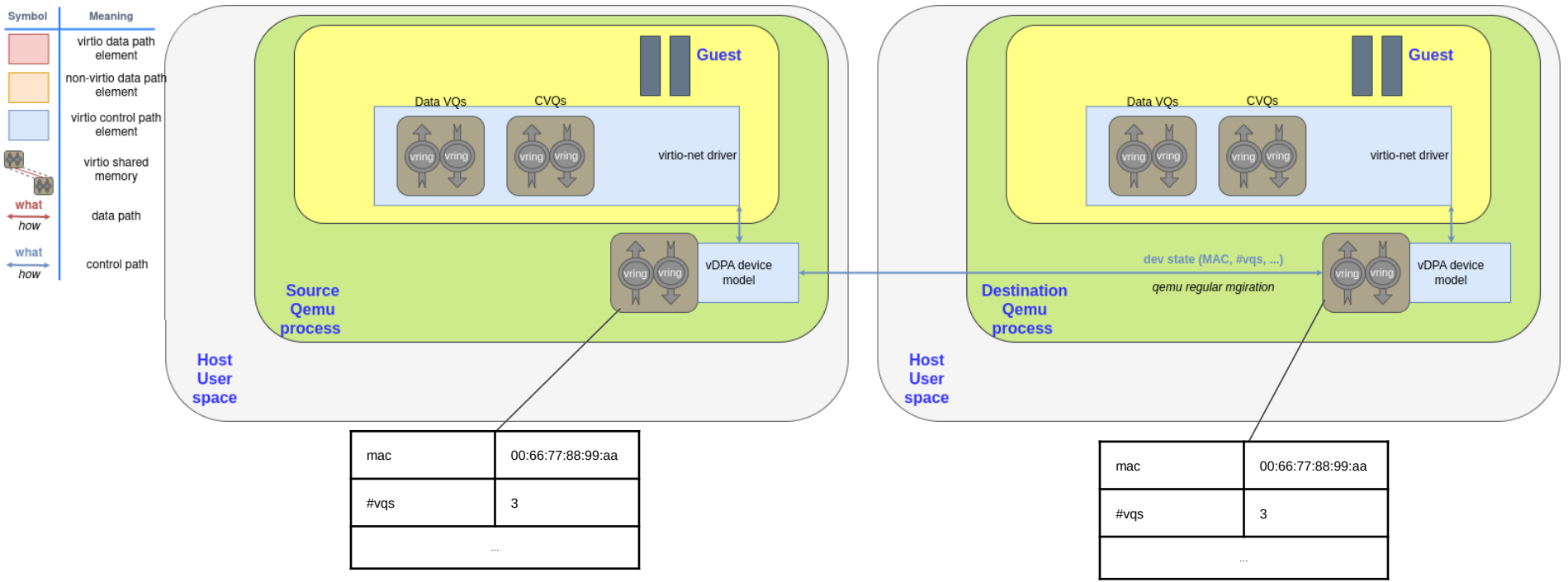


[Public]

Shadow virtqueue: migrate device state

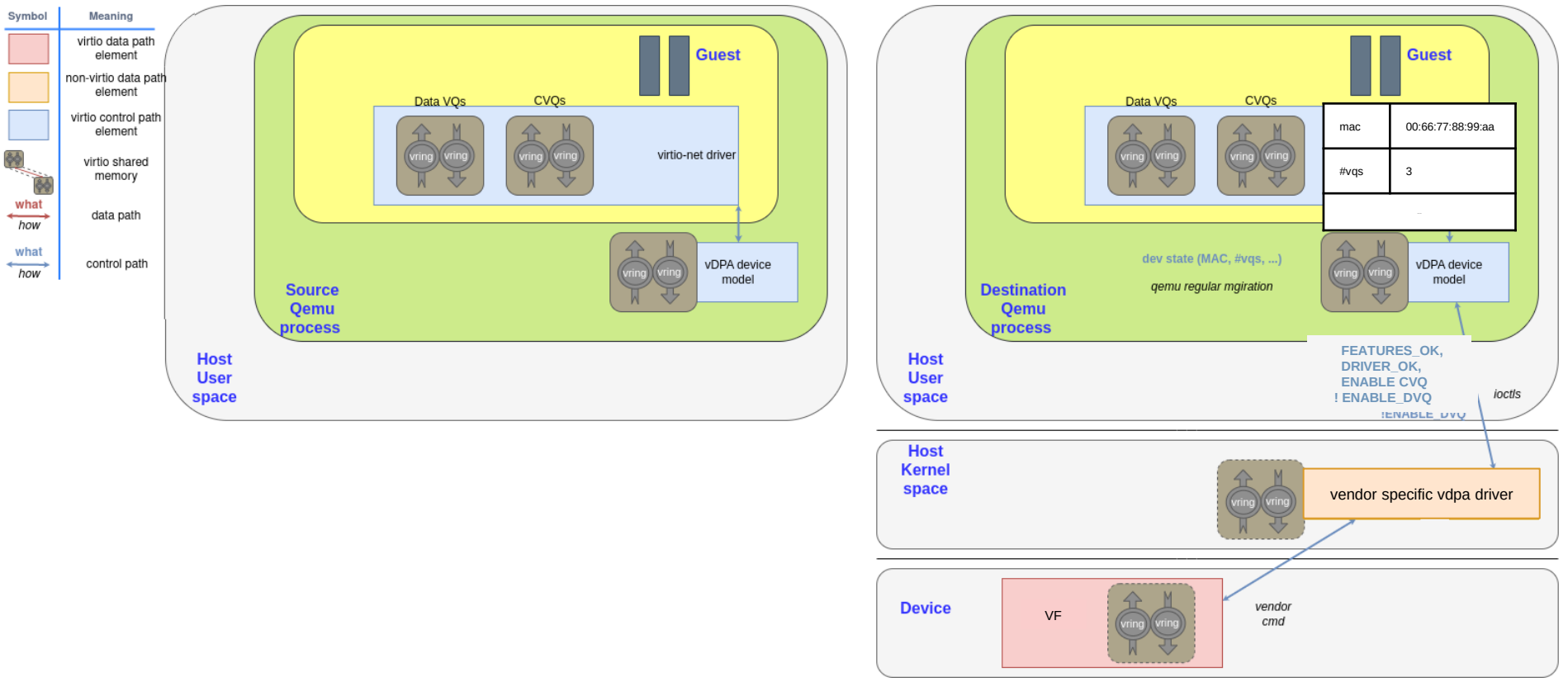


Shadow virtqueue: migrate device state

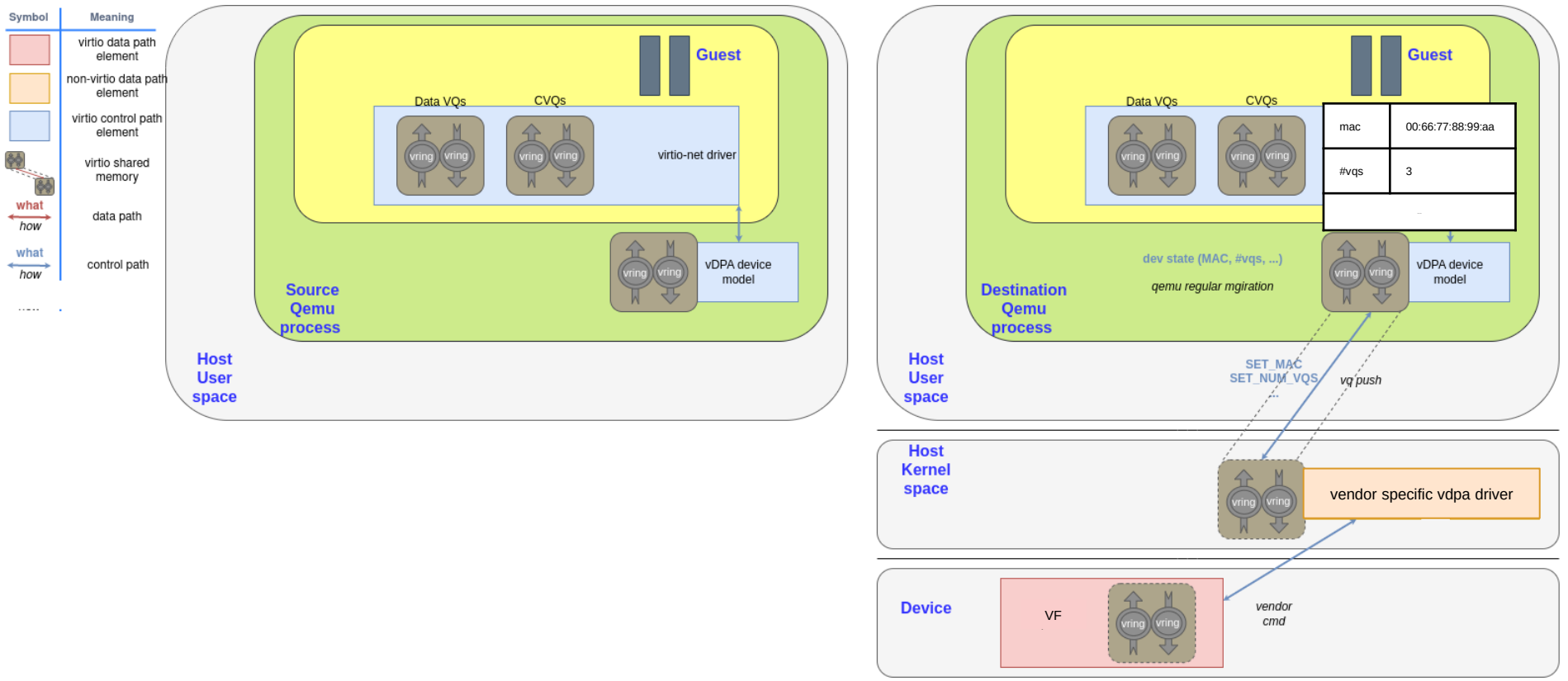


[Public]

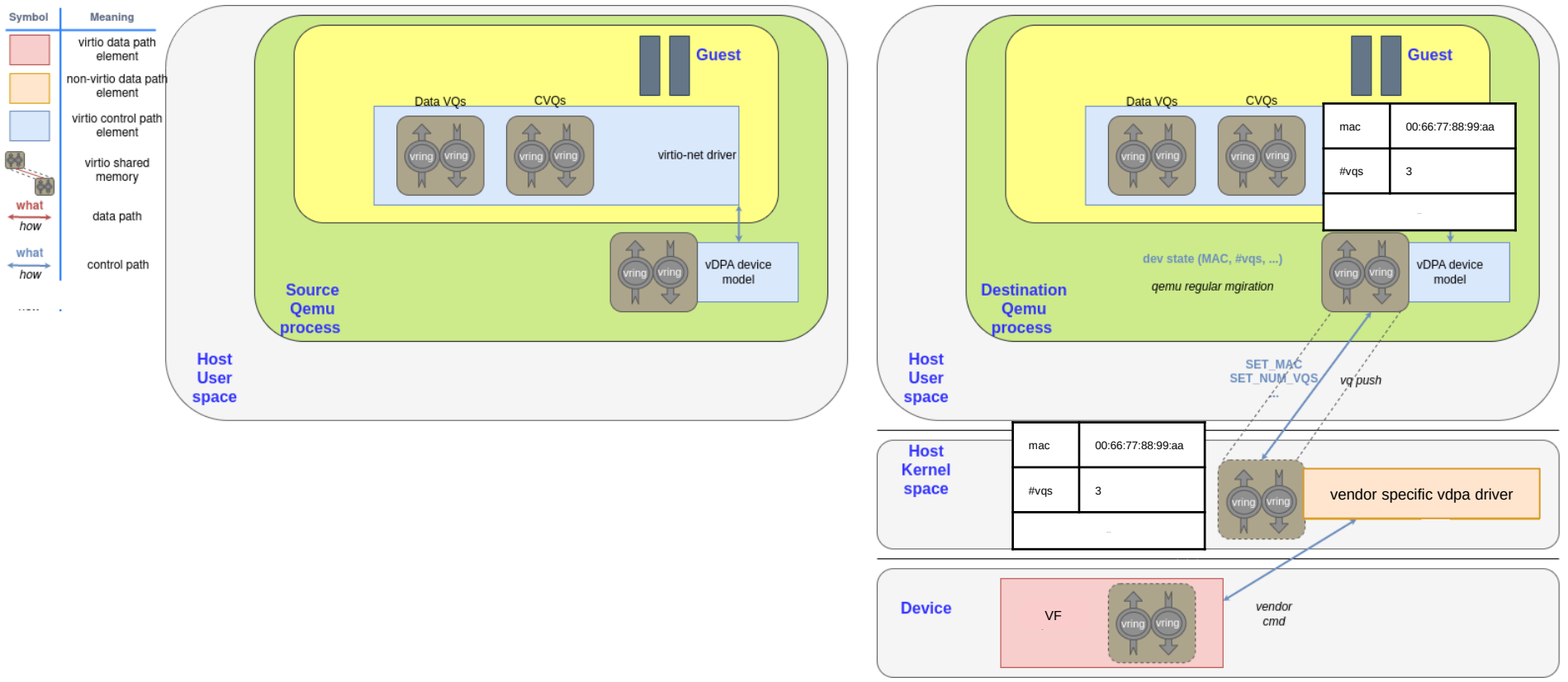
Shadow virtqueue: migrate device state



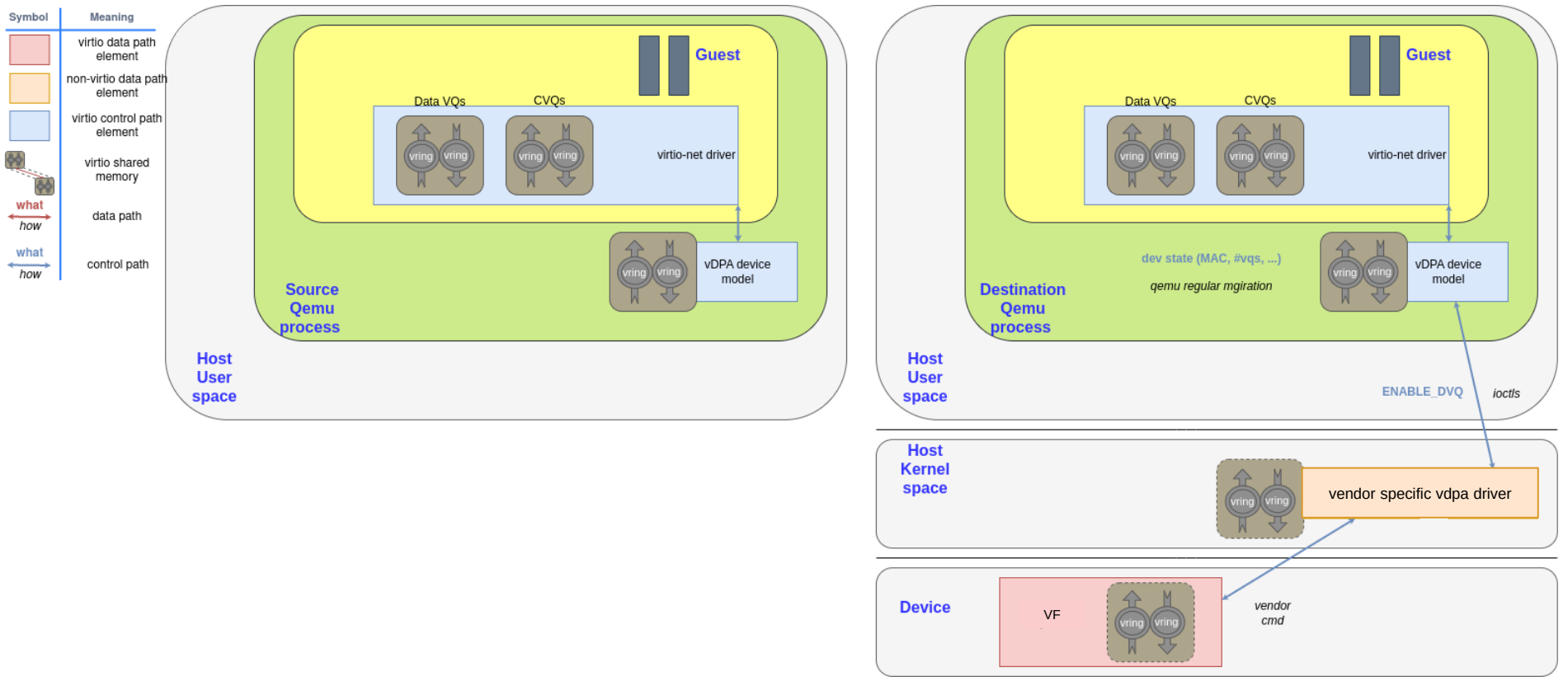
Shadow virtqueue: migrate device state



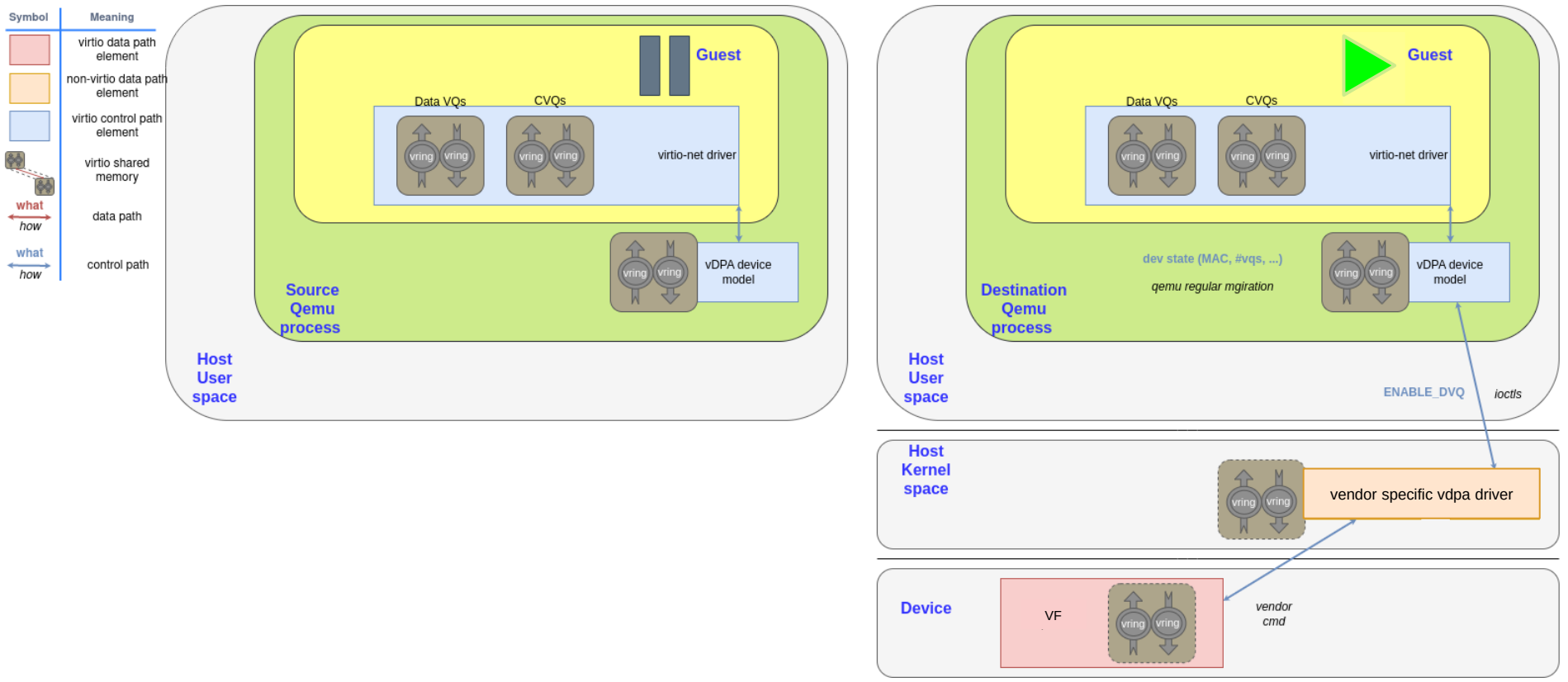
Shadow virtqueue: migrate device state



Shadow virtqueue: migrate device state



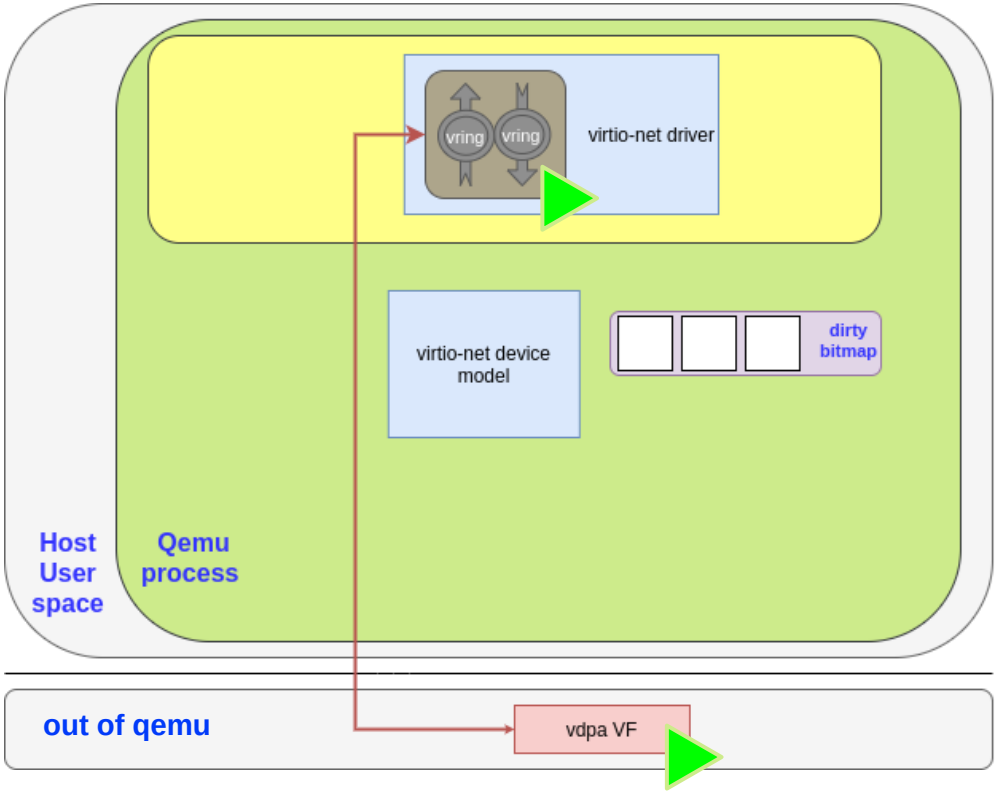
Shadow virtqueue: migrate device state



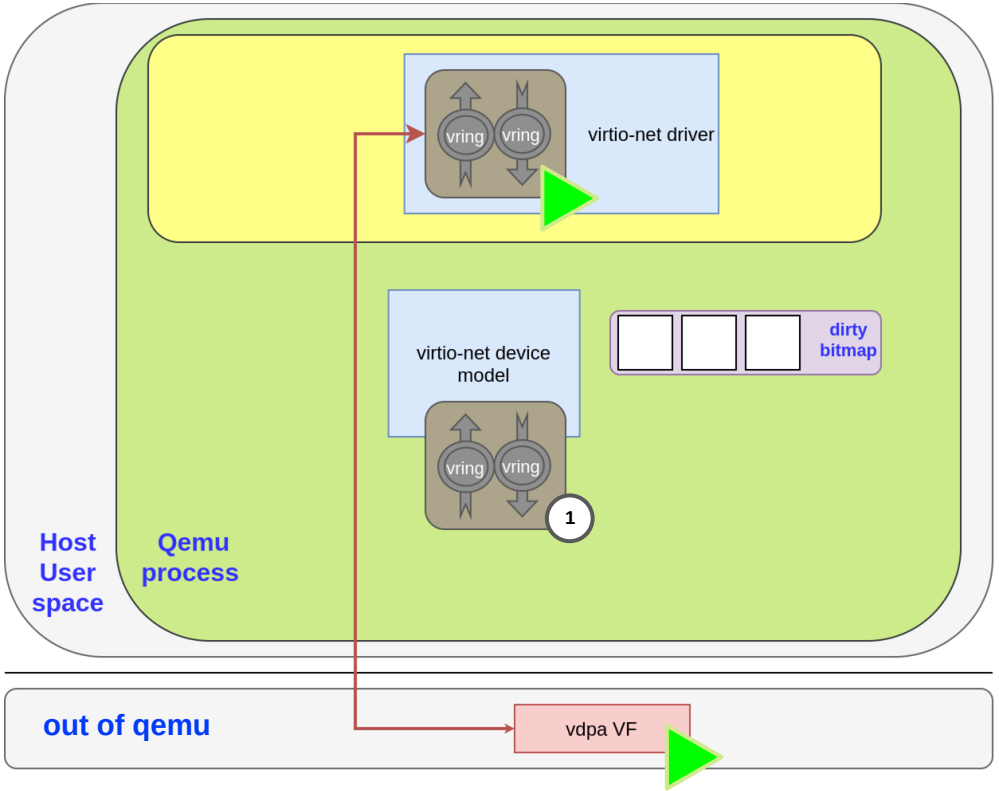
Shadow virtqueue: Dirty memory tracking

- SVQ already uses the QEMU emulated device infrastructure
  - So, dirty memory tracking is “**for free**” in terms of code changes
  - Improvements to dirty memory tracking are applied to SVQ if they apply to emulated device’s virtio

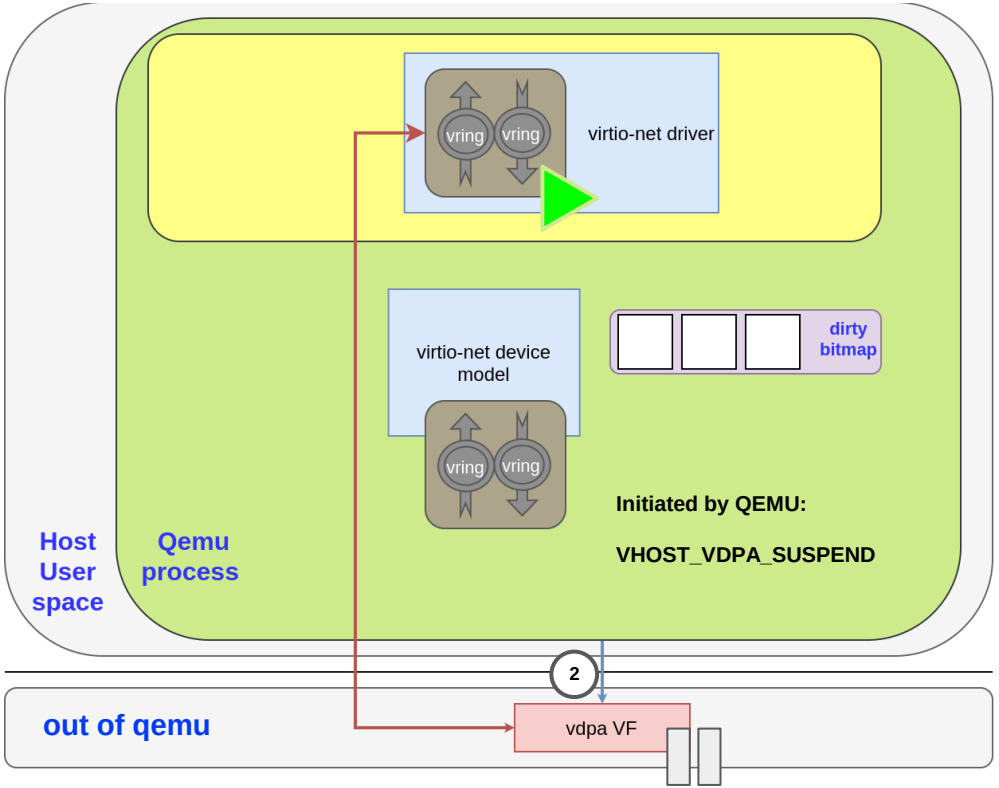
Shadow virtqueue: Regular operation



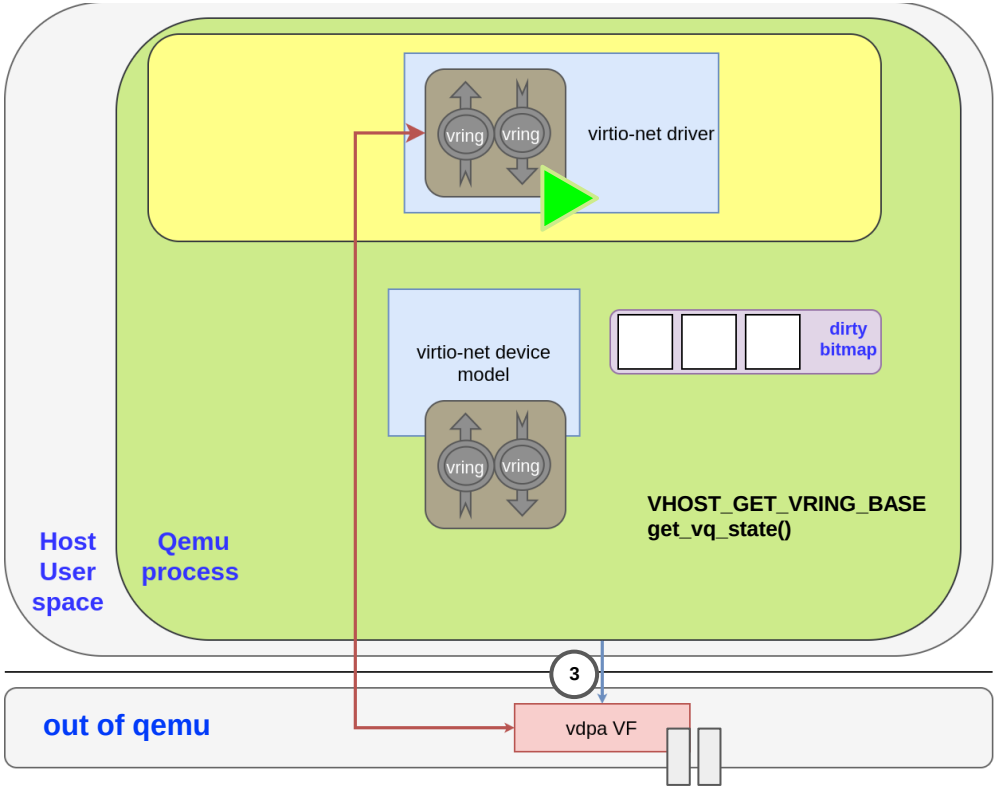
Shadow virtqueue: allocating SVQ vring



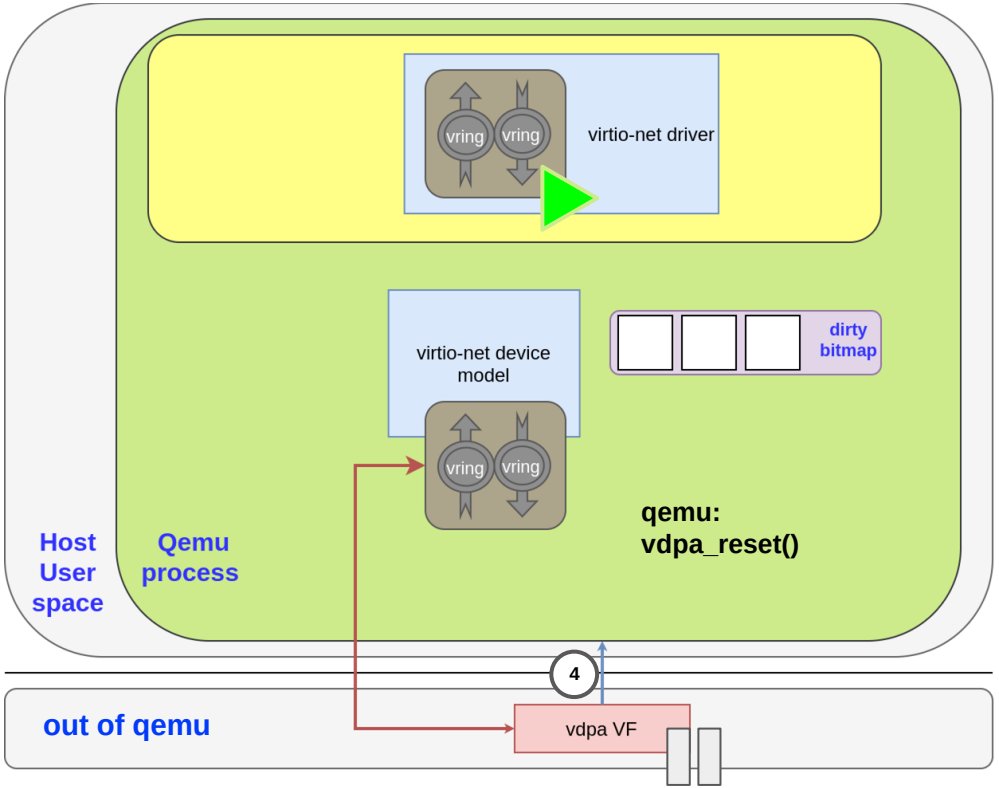
Shadow virtqueue: filling dirty bitmap



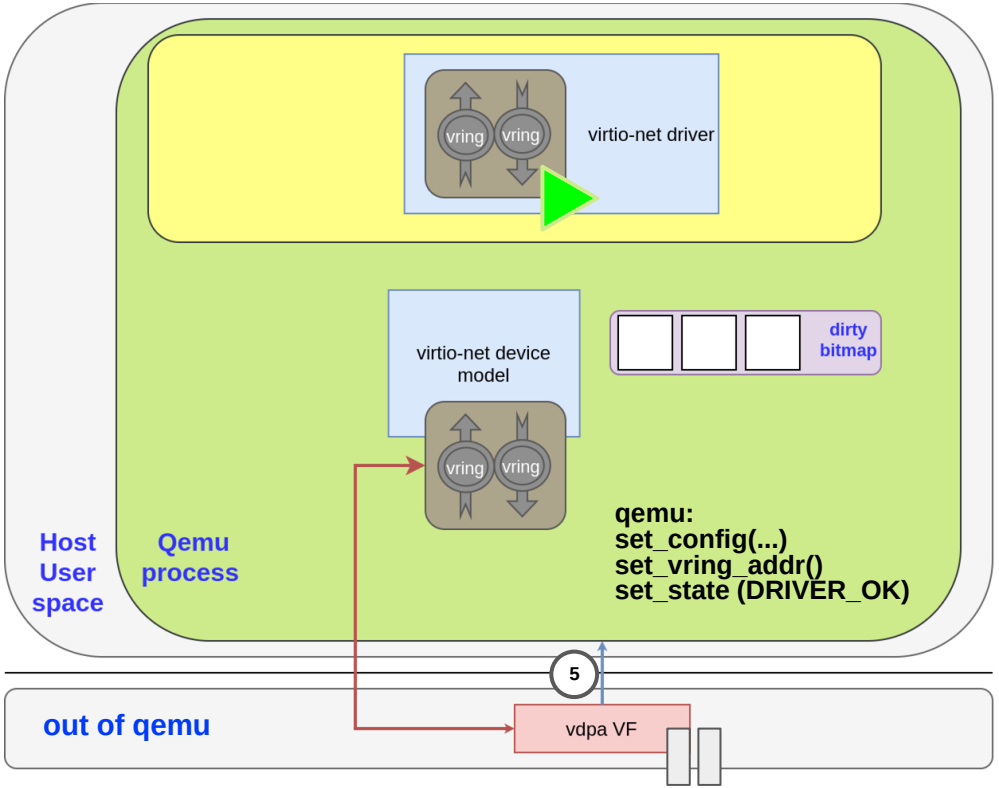
Shadow virtqueue: filling dirty bitmap



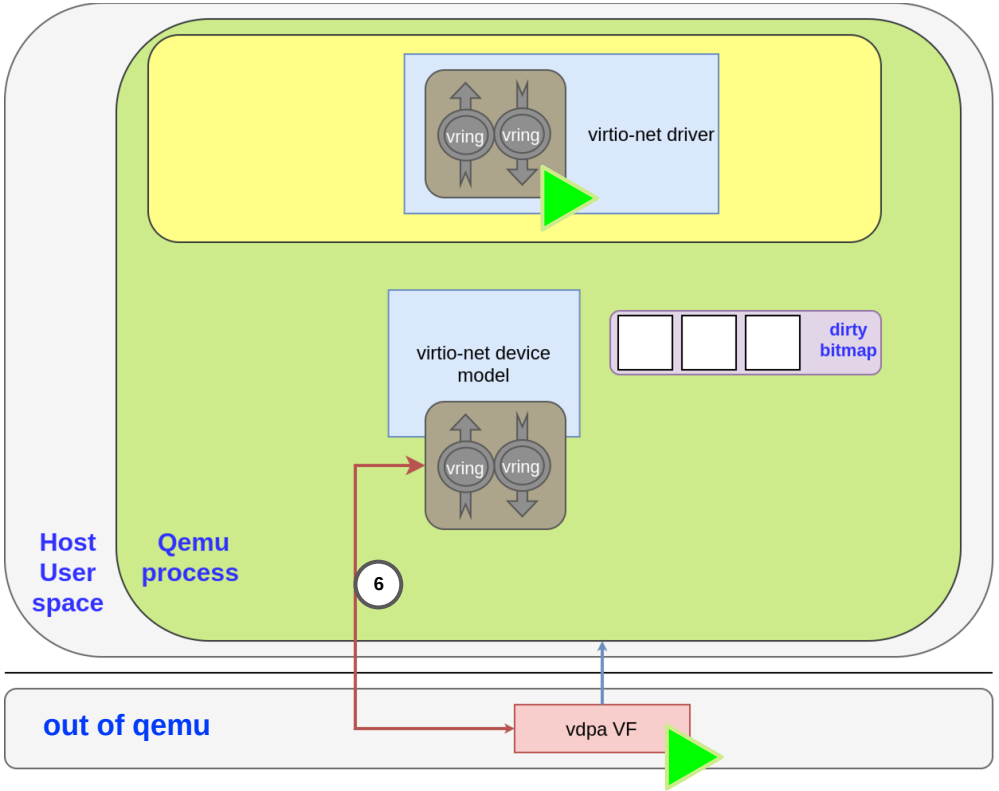
Shadow virtqueue: filling dirty bitmap



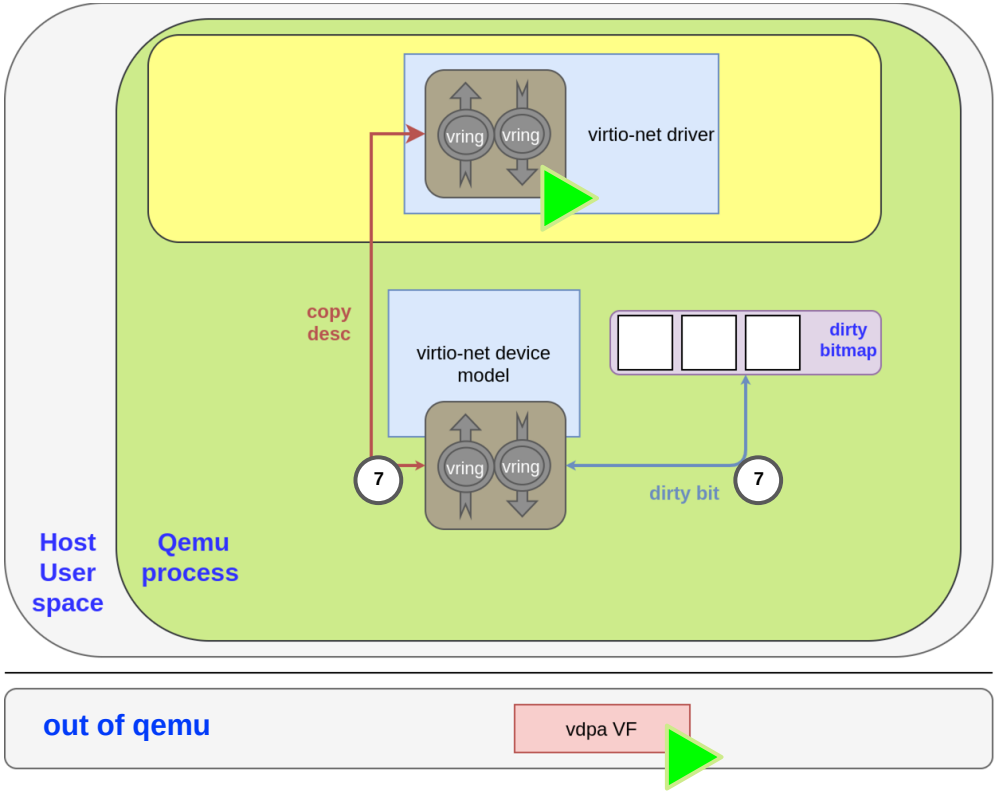
Shadow virtqueue: configure new vring



Shadow virtqueue: filling dirty bitmap



Shadow virtqueue: filling dirty bitmap



Shadow virtqueue: Recap

- No changes in the device or the guest to
  - Track device state
  - Track dirty memory
  - Restore device state

- Changes for new features are
  - about the same as adding it to qemu
  - plus, code to send them through CVQ

Vendor support for vDPA

Vendor	Device / Card
AMD-Xilinx	Alveo SN1022
AMD-Pensando	Pensando DSC-200
Nvidia	Mellanox MT2892 [ConnectX-6 Dx]
Intel	N3000, C5000X-PL, F2000X-PL
Solid-run	SolidNET LX2162A DPU
Alibaba	Alibaba ENI (Elastic Network Interface)

Alveo SN1000 SmartNIC Accelerator Card

- **Industry's *first*** SmartNIC offering software-defined hardware acceleration for all function offloads in a single platform
- Supports **custom offloads at line rate**, including customer-built and third-party offloads
  - **Network**: Open vSwitch and virtualization acceleration (Virtio.net)
  - **Security**: IPsec, kTLS and SSL/TLS
  - **Storage**: Virtio.blk, NVMe™ over TCP, Ceph, and compression & crypto services

- Based on the AMD **16nm UltraScale+™ architecture**
- Powered by the **low-latency** XCU26 **FPGA**
- **16-core Arm®** processor.

- **P4 Programmability**: Vitis Networking, P4 toolkit from AMD, enables customers to compose custom offloads and tweak existing offloads

vDPA: More information

- https://www.redhat.com/en/blog/learn-about-virtio-networking
- https://www.redhat.com/en/blog/introduction-vdpa-kernel-framework
- https://www.redhat.com/en/blog/virio-networking-series-advanced
- https://www.redhat.com/en/blog/vdpa-kernel-framework-part-3-usage-vms-and-containers
- https://www.redhat.com/en/blog/hands-vdpa-what-do-you-do-when-you-aint-got-hardware
- https://www.redhat.com/en/blog/hands-vdpa-what-do-you-do-when-you-aint-got-hardware-part-2

Thanks!

Questions?

vdpa: dirty page tracking alternatives

- Alternatives proposed
  - Device based
    - Dirty bytemap -> 8x times more memory, bad cache usage, …
    - Dirty ring -> https://lwn.net/Articles/833206/
  - IOMMU based
    - Page Request Interface PRI -> Not available at the moment?
  - Software based (QEMU)
    - failover
    - **SVQ**
      + Not related to guest's memory size but host's memory bandwidth. Automatic throttle for migration case.
      + Device does not need to learn new format (Virtio queue).
      + Re-uses emulated device -> Well tested and maintained.