ACRN Vhost Introduction

yu1.wang@intel.com Nov 2018

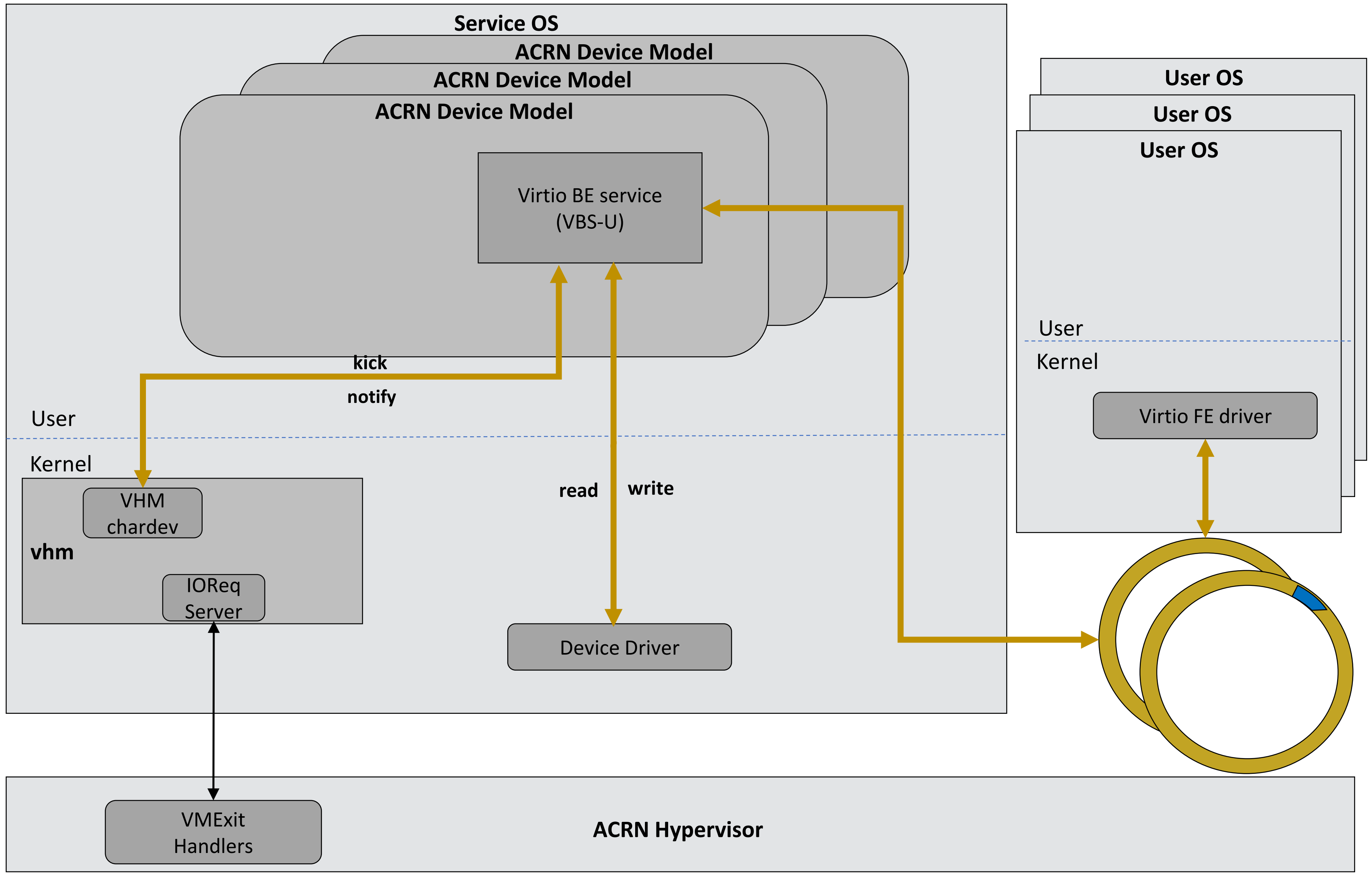
Agenda

- Vhost architecture overview
- vhm eventfd architecture
- Q/A

What is Virtio?

- Virtio is an industry para-virtualization I/O mediator interface standard.
- Virtio provides a straightforward, efficient, standard, and extensible mechanism, with which any boutique per-environment or per-OS mechanism is no longer needed.
- For example, rather than having a variety of device emulation mechanisms, virtio provides a common frontend driver framework which not only standardizes device interfaces, but also increases code reuse across different virtualization platforms.

VBS-U Architecture



- ACRN follows the virtio standard to implement I/O virtualization of the performance critical devices such as network, storage and so on.
- The virtio device be export as pci device which should be matched one pci virtio frontend driver.
- Virtio device adopts a frontend-backend architecture. Basically the FE and BE driver communicate with each other through a shared memory, i.e. the virtqueues.
- The FE driver talks to the BE driver in the same way as it is talking to a real PCIe device. And the BE driver handles requests from the FE driver by communicating native device drivers. Then trigger interrupt to notifies the FE driver if the request has been processed.

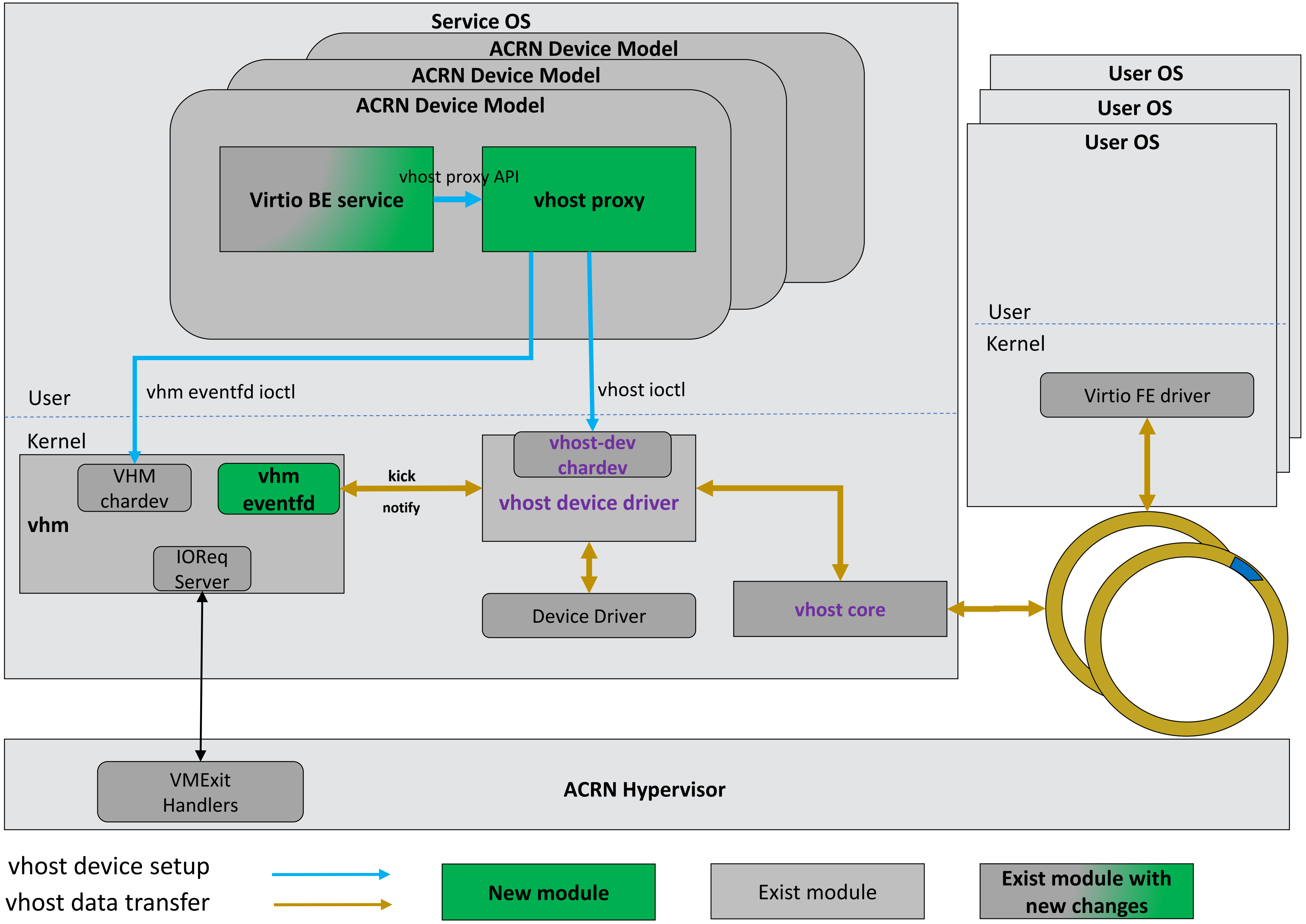
What is Vhost?

- The Vhost drivers in Linux provide in-kernel virtio device emulation.
- Normally the device model userspace process emulates I/O accesses from the guest. Vhost puts virtio emulation code into the kernel, taking device model userspace out of the picture.
- This allows device emulation code to directly call into kernel subsystems instead of performing system calls from userspace.

Pros of Vhost

- Can re-use the exist upstream Linux kernel mediators.
- Provide upstream-able kernel mediator solution.
- Smaller ioreq latency due to data path be cut down.

Vhost architecture



- vhost proxy creates 2 eventfds per virtqueue
  - One is for kick, which is a ioeventfd
  - One is for call, which is a irqfd
- vhost proxy registers the 2 eventfds to VHM through vhm char dev
  - Ioevenftd is bound to a PIO/MMIO. If it is a PIO, it is registered with (fd, port, len, value). If it is a MMIO, it is registered with (fd, addr, len)
  - Irqfd is registered with INTx/MSIX vector
- vhost proxy sets the 2 fds to vhost kernel through ioctl of vhost dev
- vhost starts polling the kick fd and is waked up when guest kicks a virtqueue which results a event_signal on kick fd by VHM ioeventfd
- vhost device in kernel signals on the irqfd to notify the guest.
- No changes are required in host kernel since it is decoupled from hypervisor.
- FE virtio drivers keep the same as those in VBS-U.

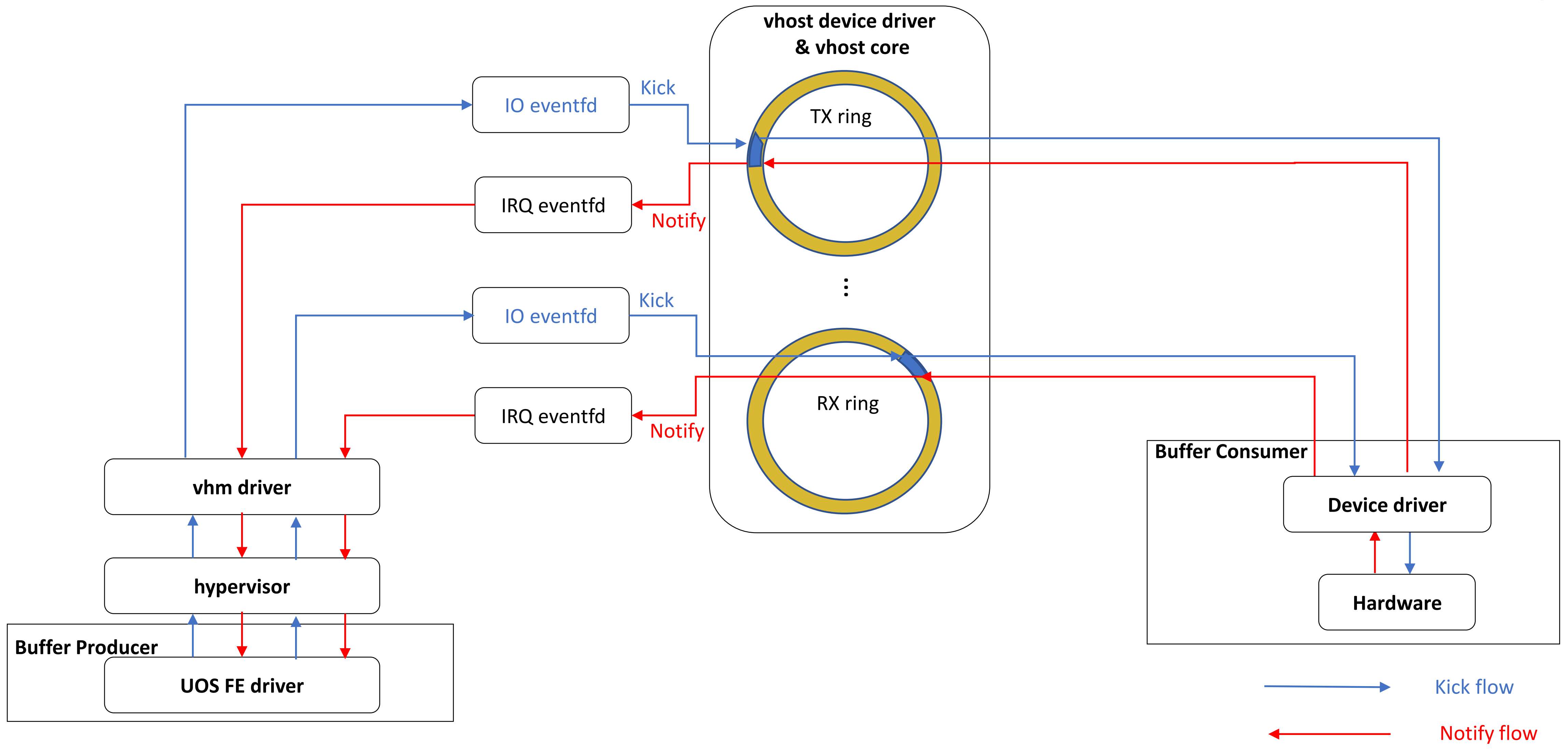
Control Plane & Data Plane

- Control Plane: Device Configurations
  - Feature negotiations: identify virtqueue info
  - Device status: ACKNOWLEDGE/FEATURES_OK/DRIVER_OK
  - PCI configuration space
- Data Plane: Device Operations
  - Batch data transfer over virtio ring, through SOS driver and UOS FE driver.
  - Notifications
    - Kick: FE -> BE, PIO/MMIO register access
    - Notify: BE -> FE, interrupt

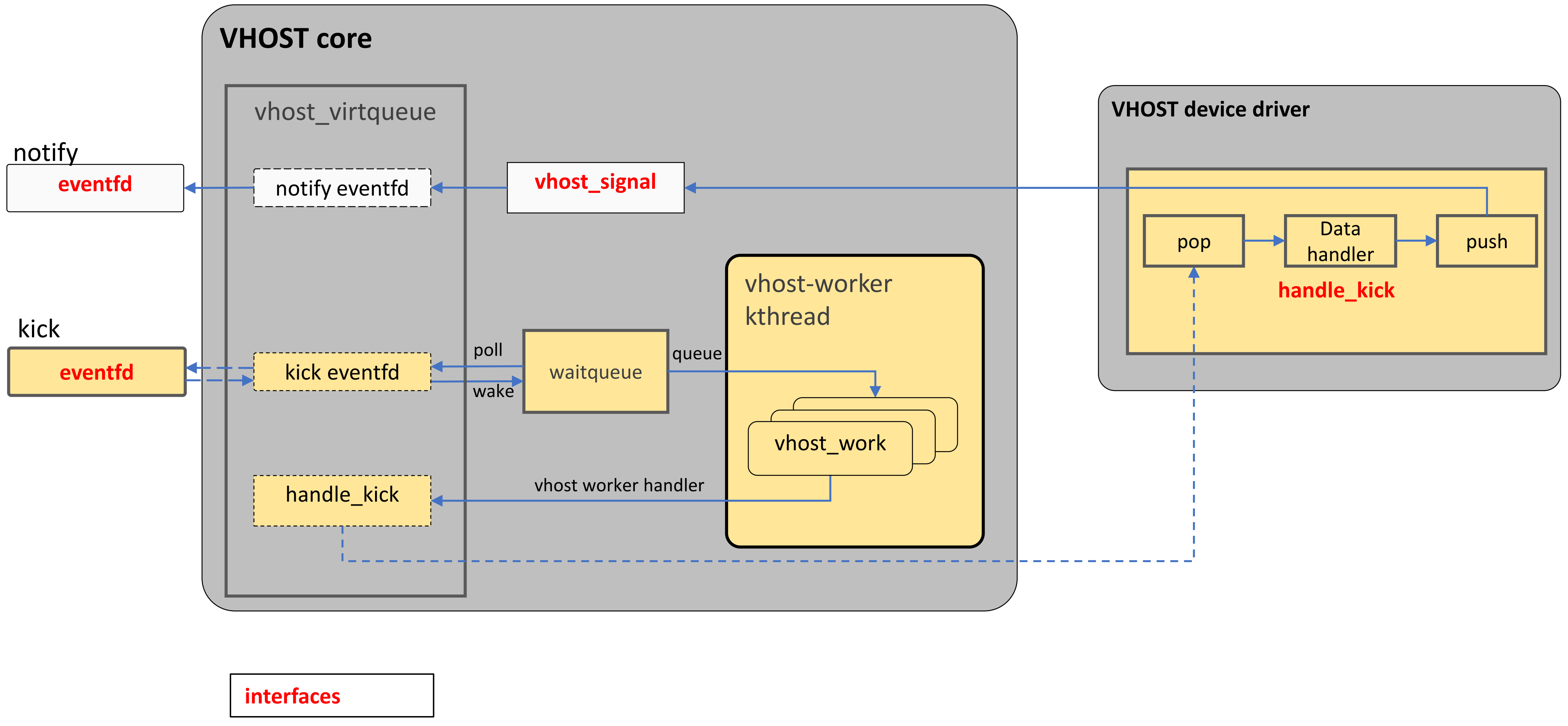
Agenda

- Vhost architecture overview
- vhm eventfd architecture
- Q/A

Core mechanism for virtio ring



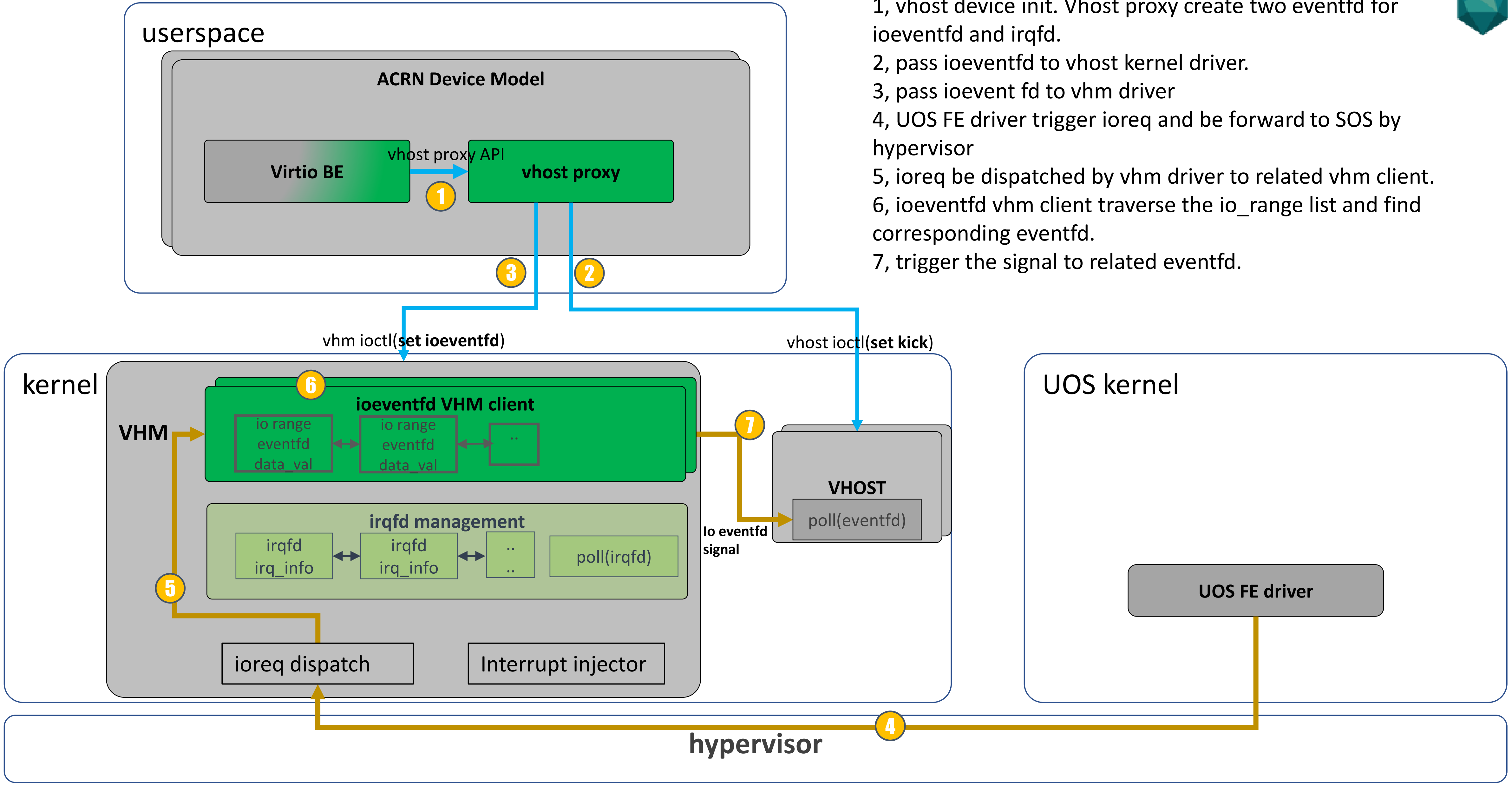
Vhost kernel framework interfaces



VHM eventfd

- Two eventfd:
  - ioeventfd and irqfd
- vhost proxy creates 2 eventfds per virtqueue
  - One is for kick, which is a ioeventfd
  - One is for call, which is a irqfd
- vhost proxy registers the 2 eventfds to VHM through ioctl
  - Ioevenftd is bound to a PIO/MMIO range. If it is a PIO, it is registered with (fd, port, len, value). If it is a MMIO, it is registered with (fd, addr, len)
  - Irqfd is registered with INTx/MSIX vector
- vhost proxy sets the 2 fds to vhost kernel through ioctl of vhost dev

ioeventfd

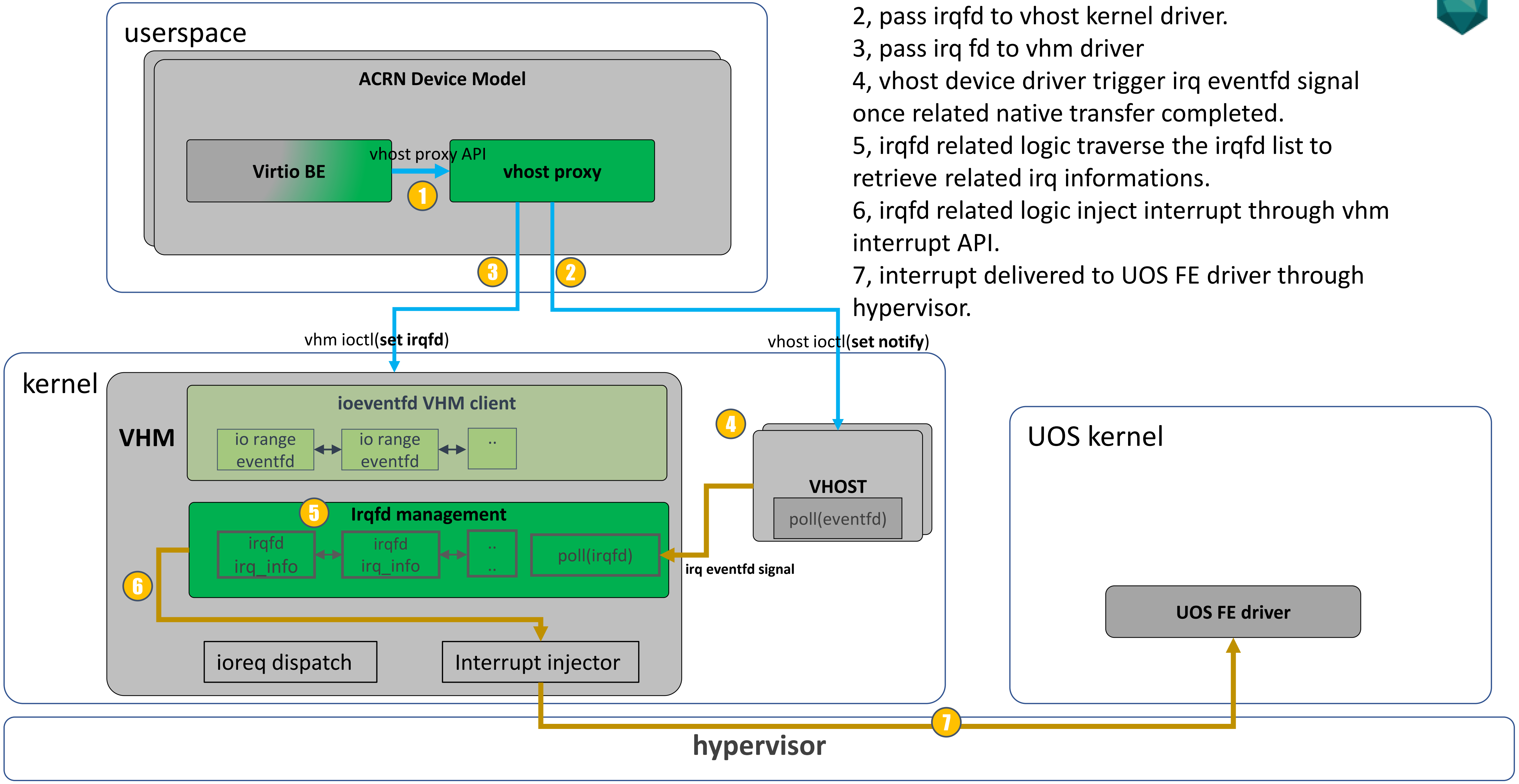


1, vhost device init. Vhost proxy create two eventfd for ioeventfd and irqfd.
2, pass ioeventfd to vhost kernel driver.
3, pass ioevent fd to vhm driver
4, UOS FE driver trigger ioreq and be forward to SOS by hypervisor
5, ioreq be dispatched by vhm driver to related vhm client.
6, ioeventfd vhm client traverse the io_range list and find corresponding eventfd.
7, trigger the signal to related eventfd.

ioeventfd

- It is used by vhost virtqueue kick mechanism
- DM registers the ioeventfd with an associated MMIO/PIO range into VHM. When guest access resources in this range, ioeventfd will be notified.
- VHM need has a dedicated client for multiple ioeventfd of each VM.

irqfd



1, vhost device init. Vhost proxy create two eventfd for ioeventfd and irqfd.
2, pass irqfd to vhost kernel driver.
3, pass irq fd to vhm driver
4, vhost device driver trigger irq eventfd signal once related native transfer completed.
5, irqfd related logic traverse the irqfd list to retrieve related irq informations.
6, irqfd related logic inject interrupt through vhm interrupt API.
7, interrupt delivered to UOS FE driver through hypervisor.

irqfd

- It is used by vhost virtqueue notify mechanism
- DM registers the irqfd with an associated intx or msi.
- Vhost write the fd to trigger the interrupt injection through VHM hypercall

Q/A?