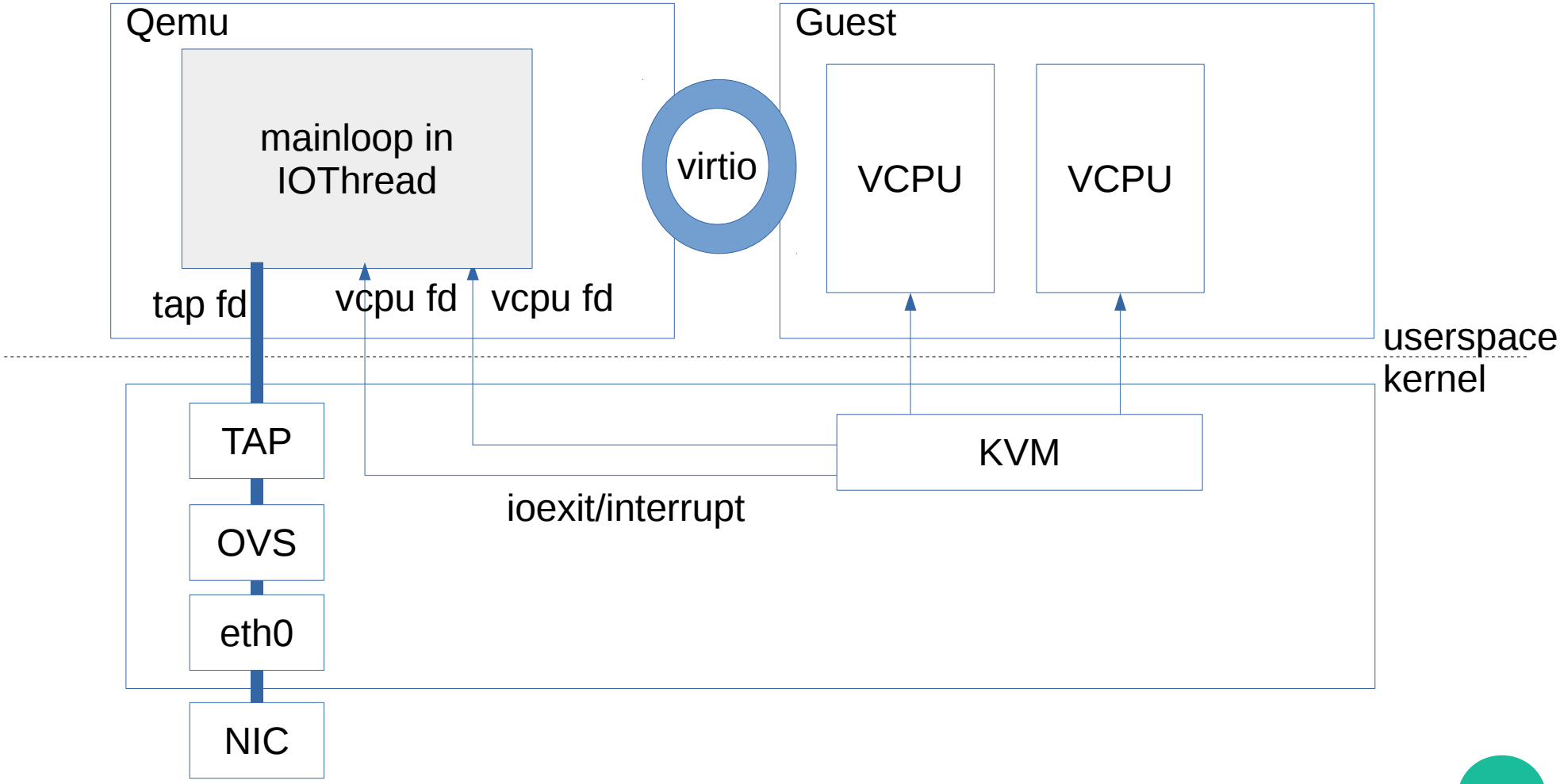
Vhost dataplane in Qemu

Jason Wang
Red Hat

Agenda

- History & Evolution of vhost
- Issues
- Vhost dataplane
- TODO

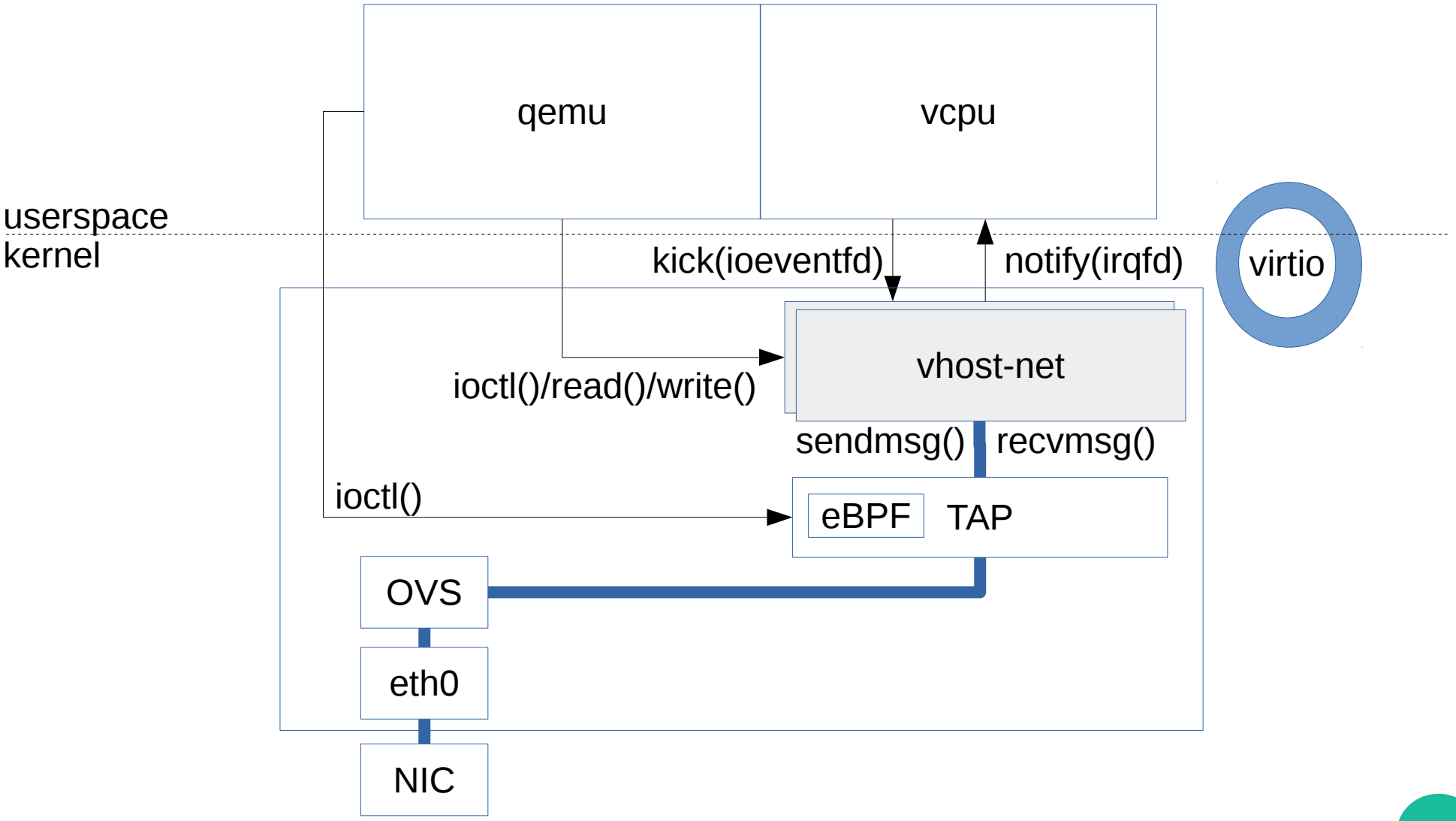
Userspace Qemu networking



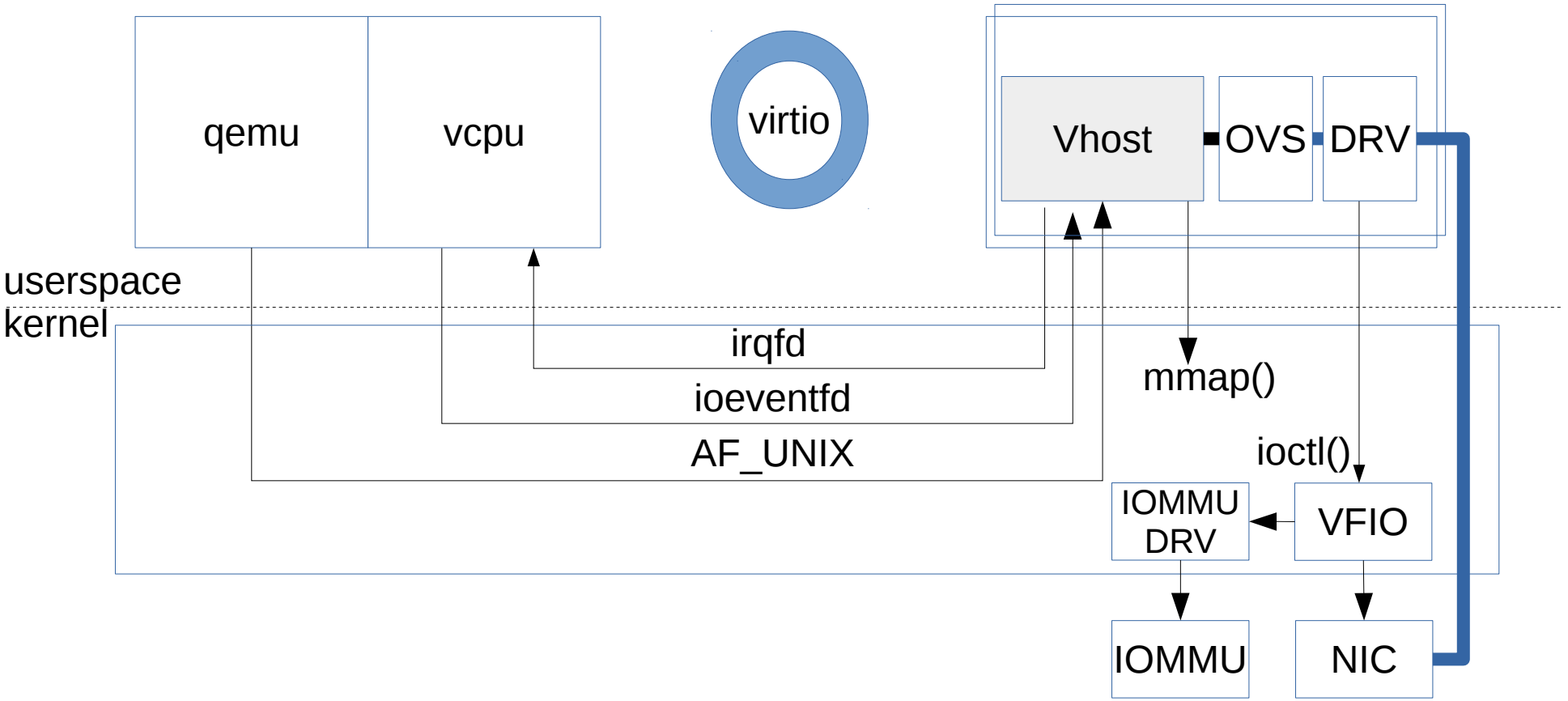
Userspace qemu networking is slow

- **Limitation of both qemu and backend**
  - Run inside mainloop
    - No real multiqueue
    - No dedicated thread, No busy polling
  - Extra data copy to internal buffer
  - TAP
    - syscall to send/receive message
  - IRQ/ioexit is slow
    - VCPU needs to be blocked
    - Slow path
  - No burst/bulking

Vhost kernel



Vhost user


qemu
vcpu
virtio
Vhost
OVS
DRV
userspace
kernel
irqfd
ioeventfd
AF_UNIX
mmap()
ioctl()
IOMMU
DRV
VFIO
IOMMU
NIC

VHost

- **Offload dataplane to another process**
  - kthread or userspace process
- **A set of API that did**
  - Features Negotiation
  - MEM Table
  - Dirty page logging
  - Virtqueues setting
  - Endianess
  - Device specific
- **An API transport**
  - ioctl()
  - AF_UNIX

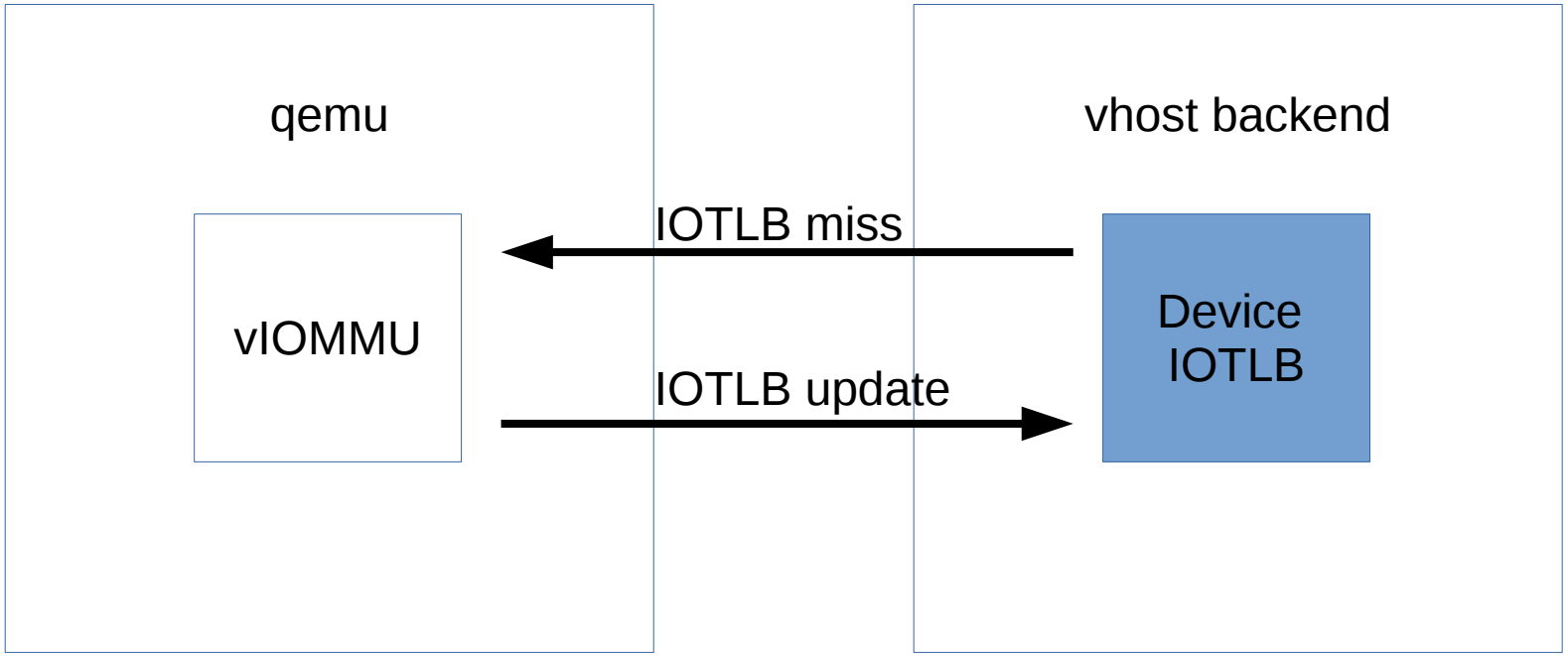
So far so good?

How hard for adding a new feature

- **Formalization in Virtio Specification**
- **Codes in qemu userspace virtio-net backend**
- **Vhost protocol extension:**
  - Vhost-kernel (uapi), vhost-user (has its own spec)
  - Versions, feature negotiations, compatibility
- **Vhost support codes in qemu (user and kernel)**
- **Features (bugs) duplicated everywhere:**
  - vhost_net, dpdk, TAP, macvtap, OVS, VPP, qemu

Even if we manage to do this

Device IOTLB

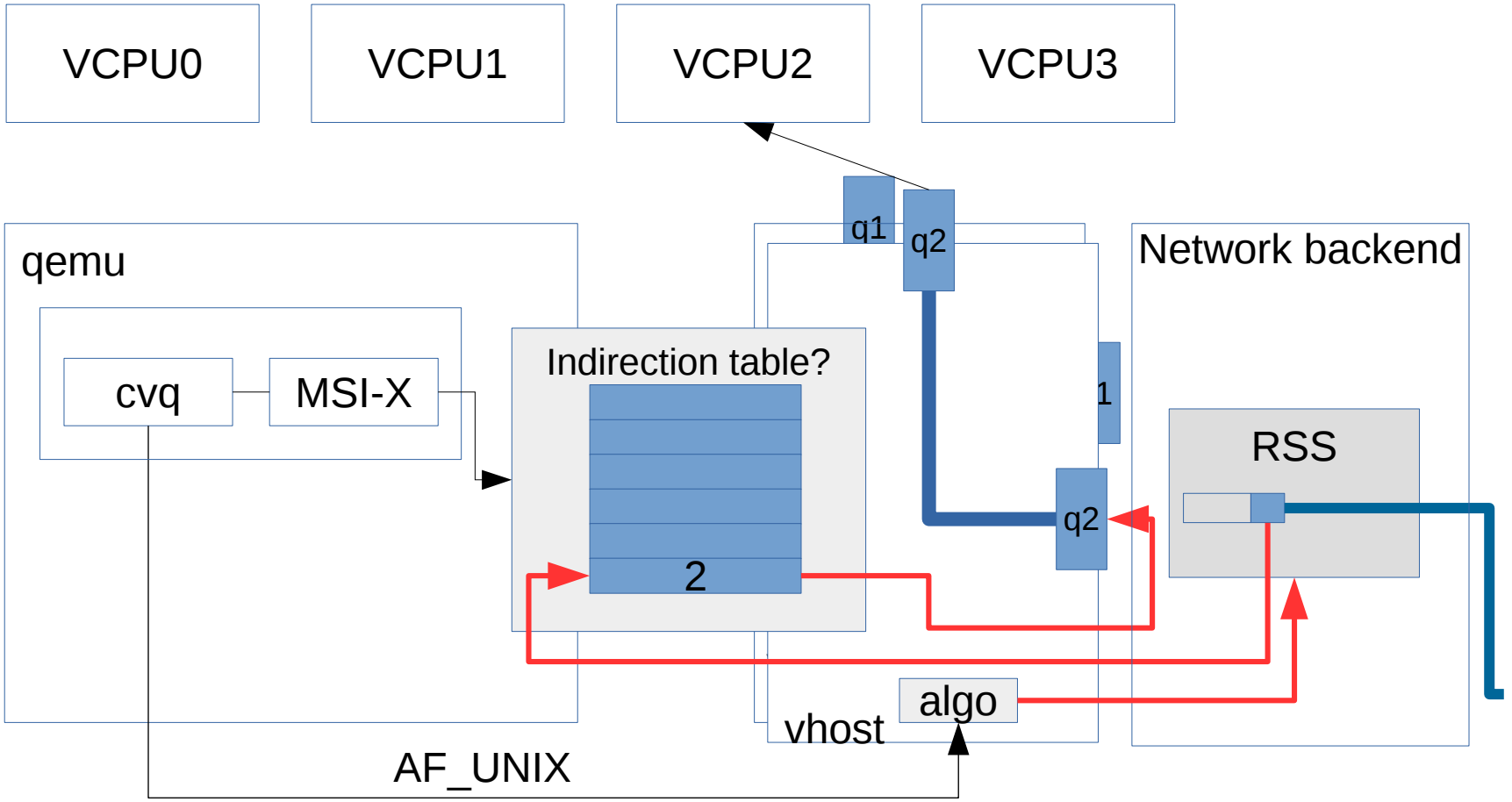


slow or even unreliable
Minor impact for static mapping
Poor performance for dynamic mapping

Issue

Datapath needs information from control path. But vhost control path is not designed for high performance.

Receive Side Scaling

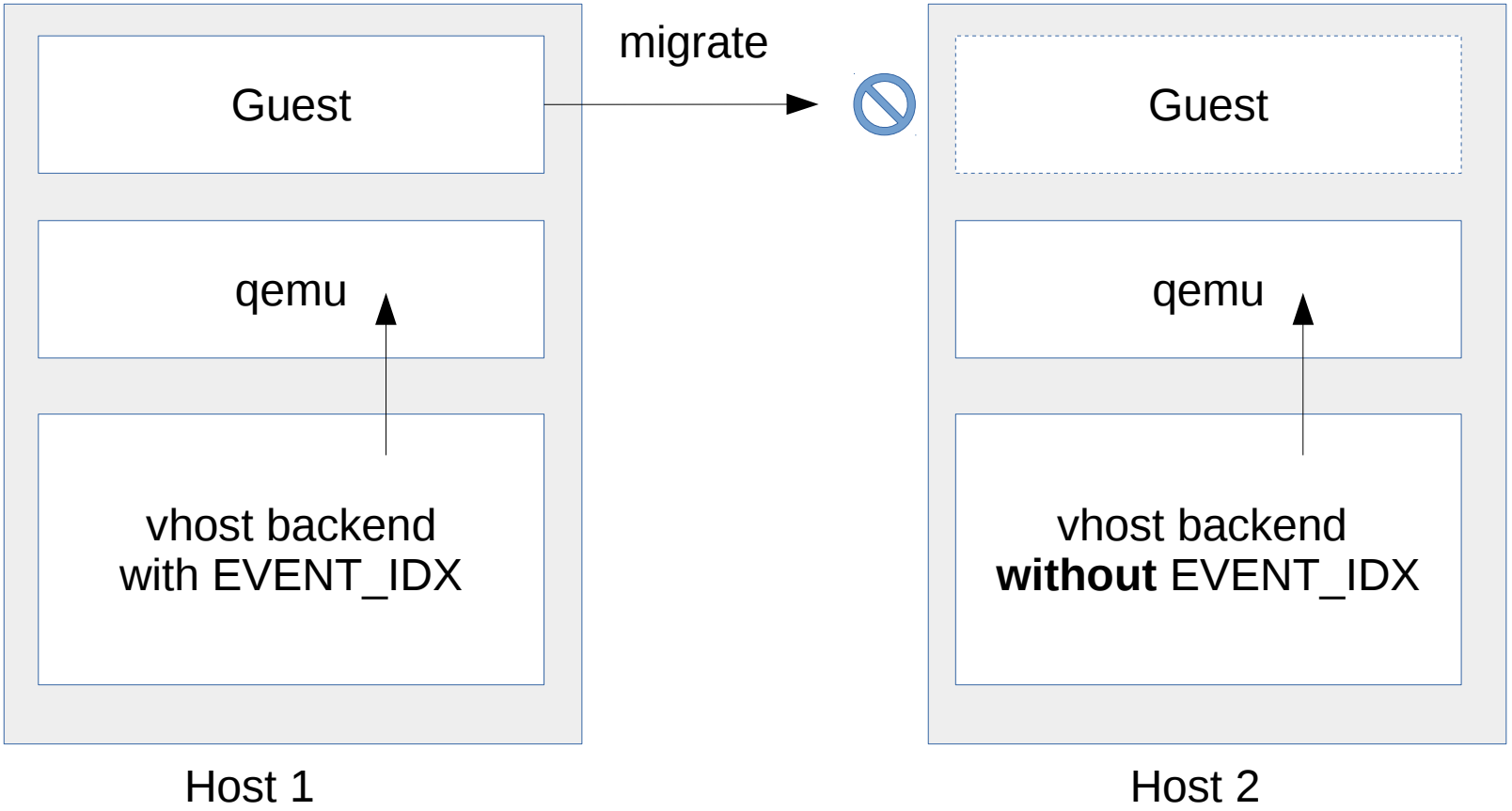


More kinds of steering policy?

Issue

Networking backend is transparent to qemu in the case of vhost-user. Net specific request through vhost-user.

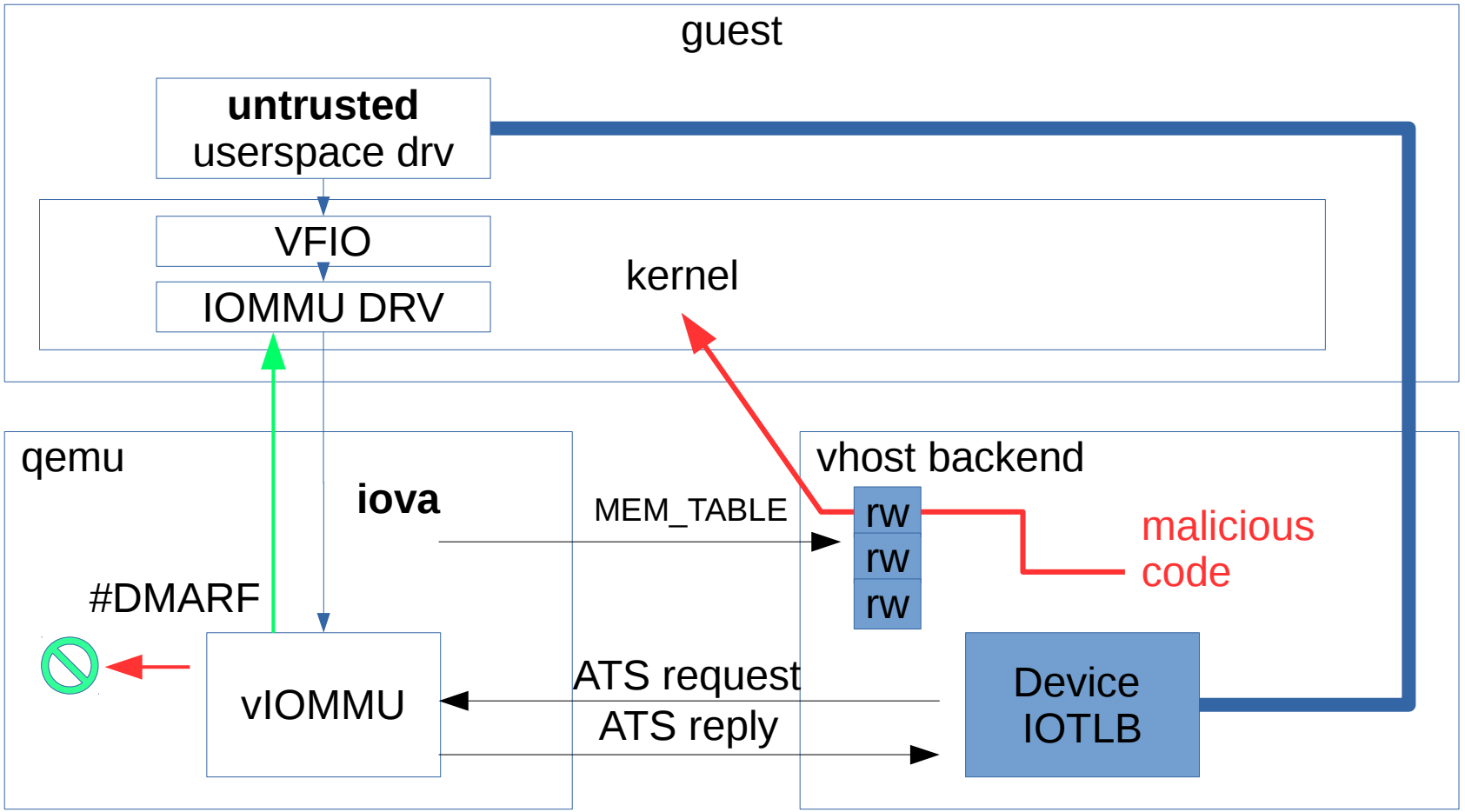
Migration compatibility



Issue

Though features was negotiated during startup. Backend needs to implement each features for providing migration compatibility.

Attack surface



can protect malicious guest usersapce driver.
but not malicious vhost-backend.

Issue

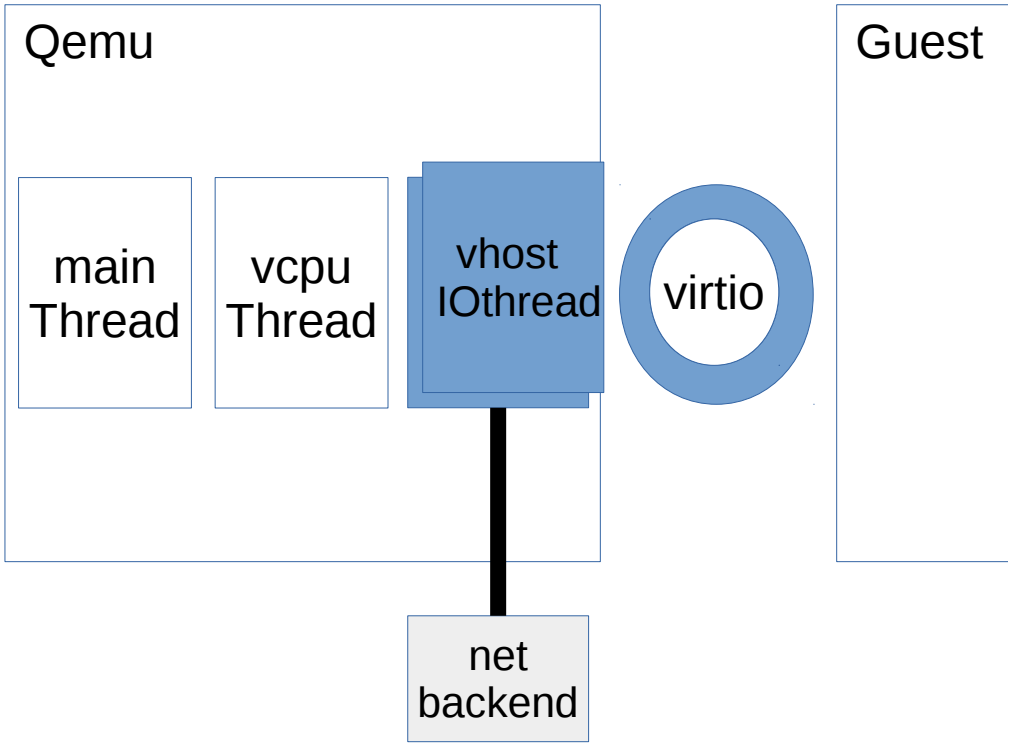
We don’t want to trust vhost-user backend
But we share (almost) all memory to it!

Issues with external vhost process

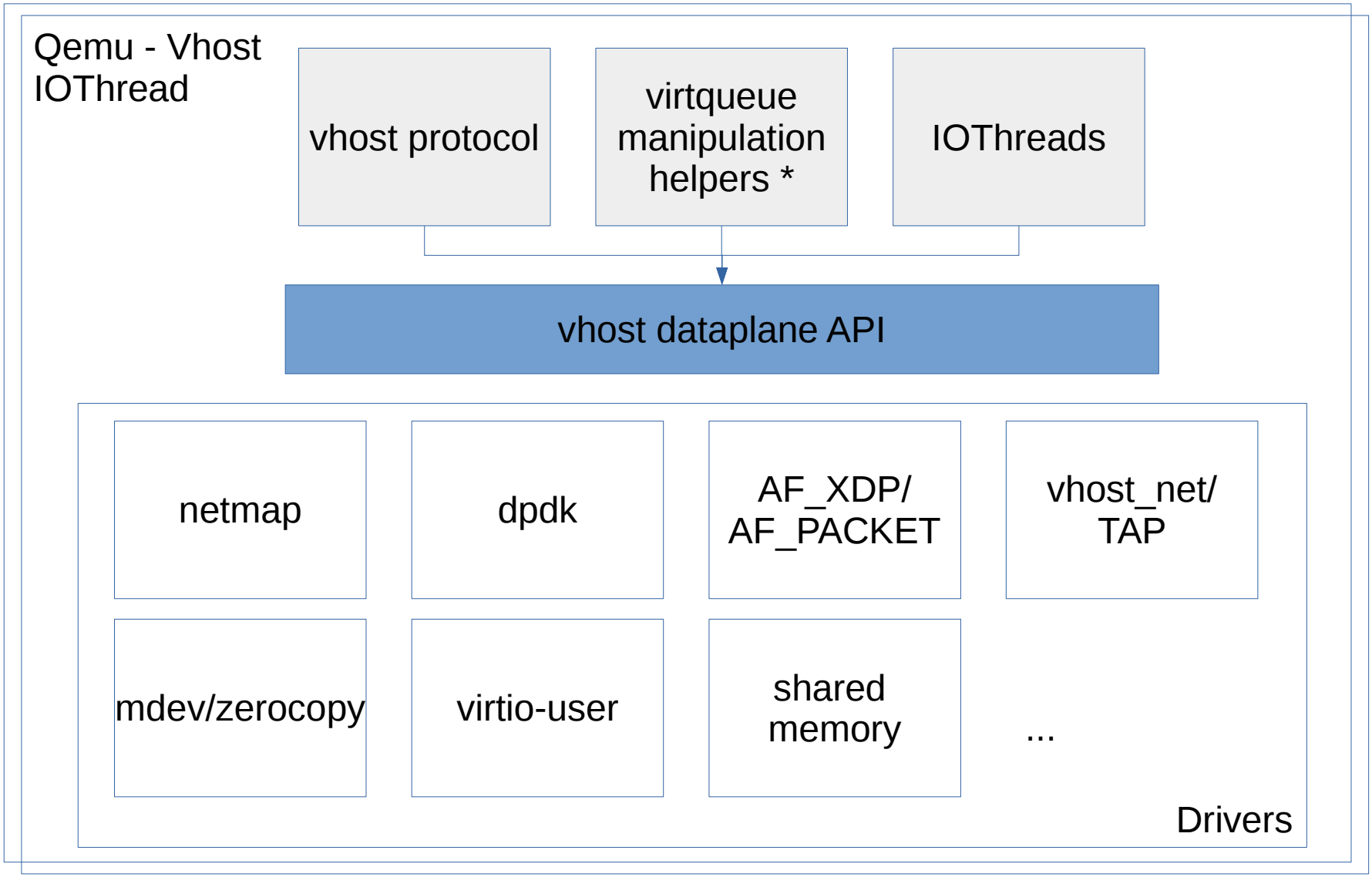
- **Complexity in Engineering**
  - Hard to be extended, duplicated codes(bugs) in many places
- **Performance is not always good**
  - Datapath can not be offloaded completely
- **Visibility of networking backend**
  - Re-invent wheels in vhost-user procotol
- **Divergence of protocol between vhost-kernel and vhost-user**
  - Workarounds, how to deal with the 3rd vhost transport?
- **Increasing of attack surface**

Vhost dataplane = Vhost through qemu IOThread

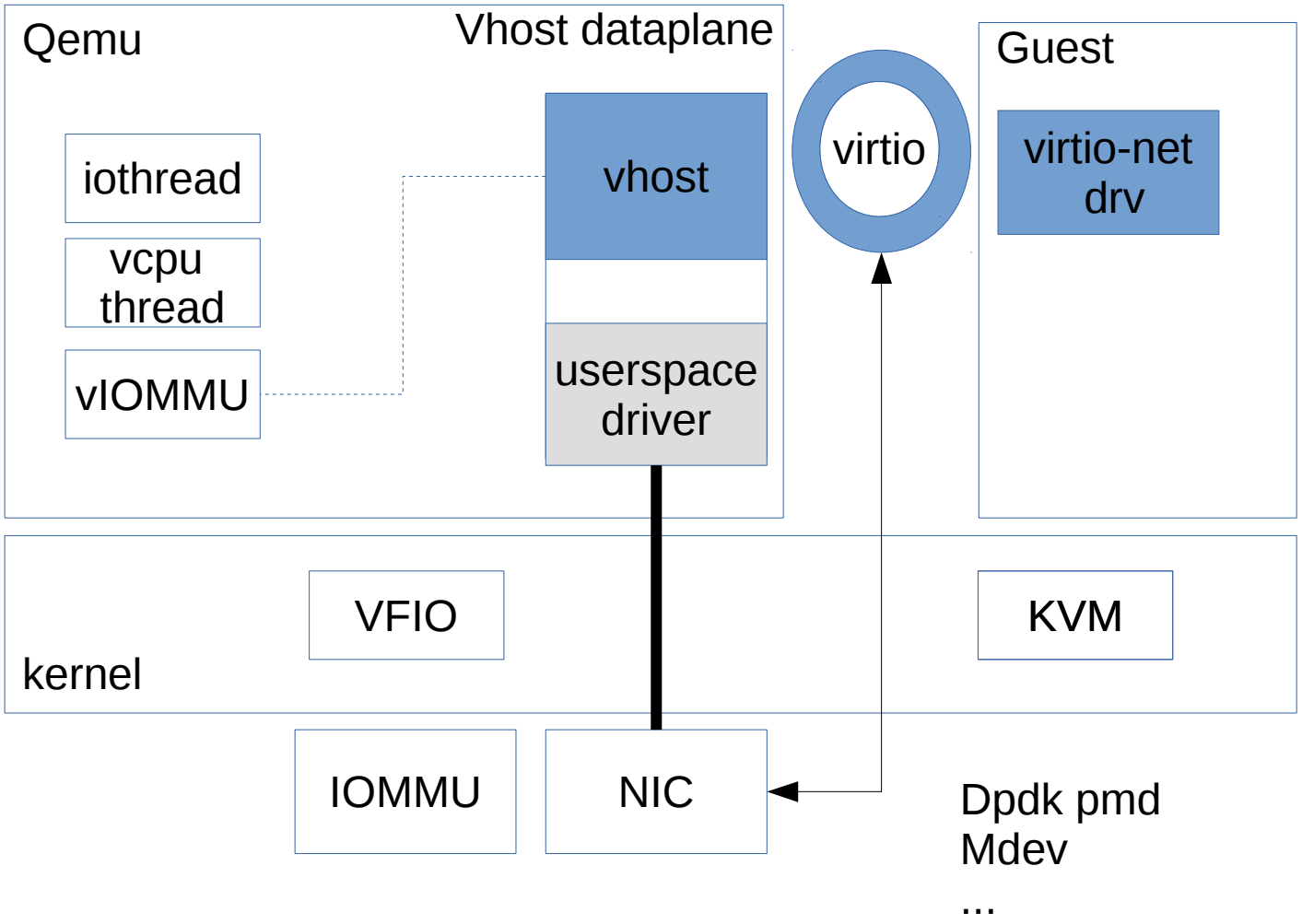
- **Vhost IOThreads**
  - Datapath in vhost IOThread
    - Hide VM state from backends
    - Function call for state accessing, better vIOMMU
    - Decouple vitio out of backends
  - Full functional features through control vq
  - Fast address translation (vhost memory table)
  - Copy inside qemu
  - Drivers for various backends
  - Multiqueue



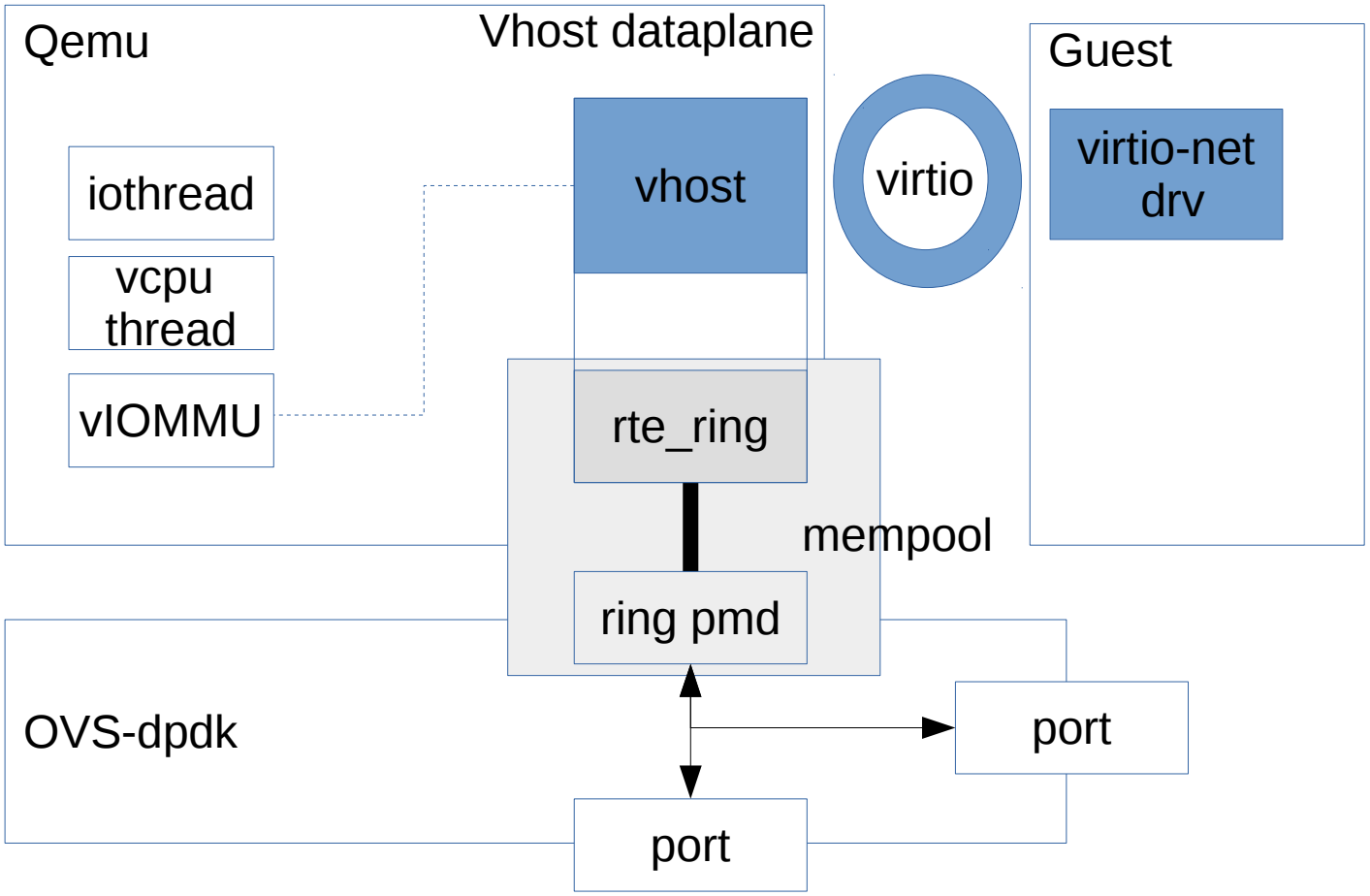
Vhost dataplane



Inline driver



Multi-process cooperation



Vhost friendly networking backend

- **Generic inline networking functions:**
  - TX/RX, Multiqueue, QOS, GSO, steering ...
- **Secure and efficient IPC**
  - No knowledge of virtio
  - Stable ABI
- **Programmability for userspace defined polices**
- **Do we have something existed?**
  - AF_XDP?

External vs vhost-dataplane

	remote dataplane	vhost-dataplane
VM metadata access	Slow, inter process communication	**fast, function call**
New feature development	Hard, New types of IPCs	**easy, limited to qemu (or programmibility from backend)**
Compatibility	Complex, extra works on the backend	**easy, limited to qemu**
New backend integration	Hard, need to know all about virtio	**easy, no need to know virtio**
Attack surface	Increased	**limited to qemu**
Backend visibility	May be transparent	**Visible**

Virtio-net = virtio + networking

- **Vhost dataplane**
  - Virtio functions in vhost IOThread
  - Networking functions in the backend
- **Limitation**
  - More cores for multi process cooperation
  - The ideal networking backends does not exist in real world
    - invent one?
  - ...

Status & TODO

- **Status**
  - prototype
    - Basic IOThreads / Virtqueue helpers
    - TAP drive
      - -device virtio-net-pci,netdev=vd0 -netdev vhost-dp,id=vd0,driver=tap-driver0 -object vhost-dp-tap,id=tap-driver0
  - RFC sent in next few months
- **TODO**
  - Dpdk static linking
  - vIOMMU, Multiqueue
  - Benchmarking

Thanks