Zero-copy Receive for vhost

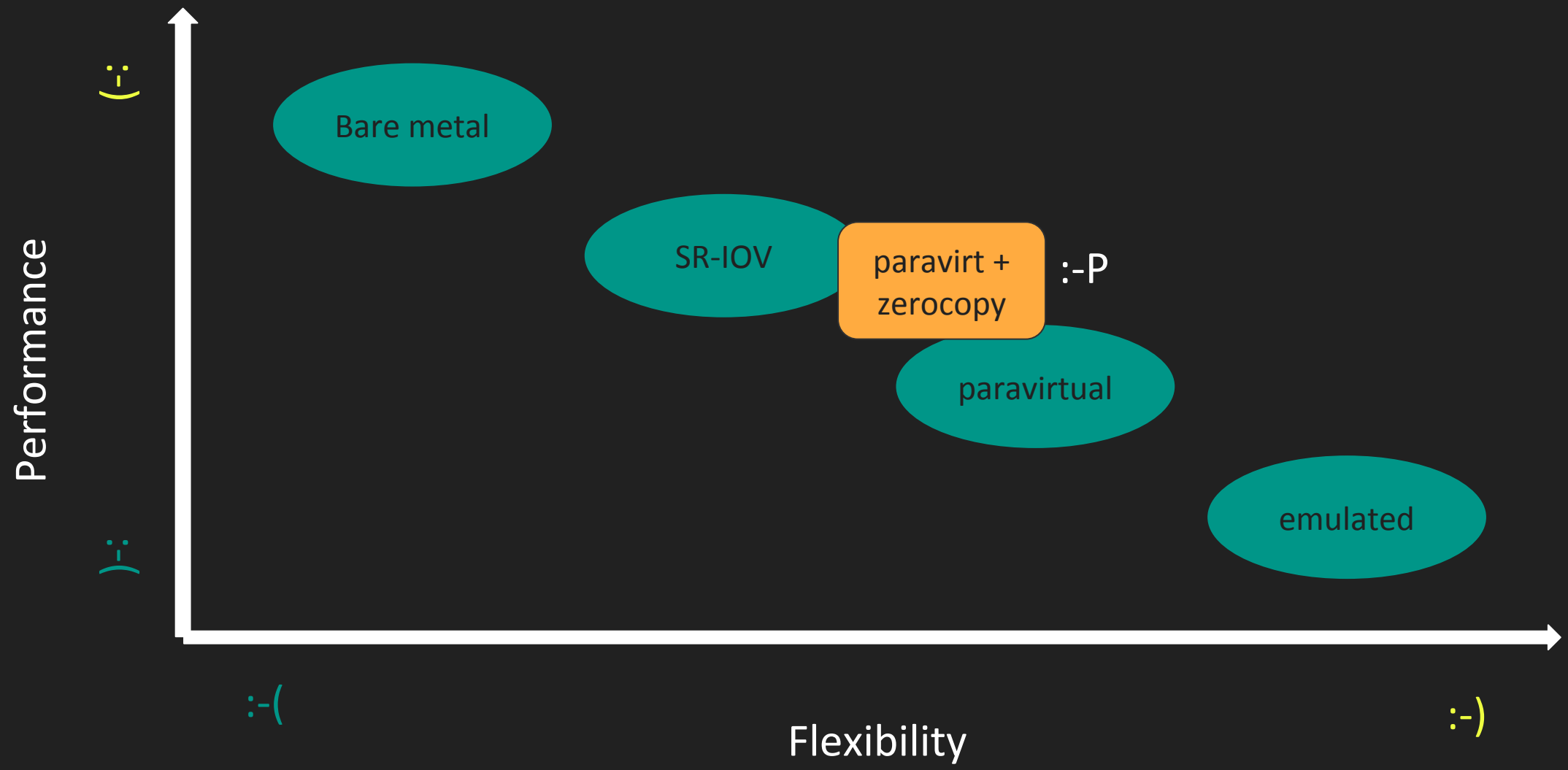
Kalman Meth, Mike Rapoport, Joel Nider

{meth,joeln}@il.ibm.com

rppt@linux.vnet.ibm.com

This project has received funding from the *European Union's Horizon 2020 research and innovation programme* under grant agreement No 645402.

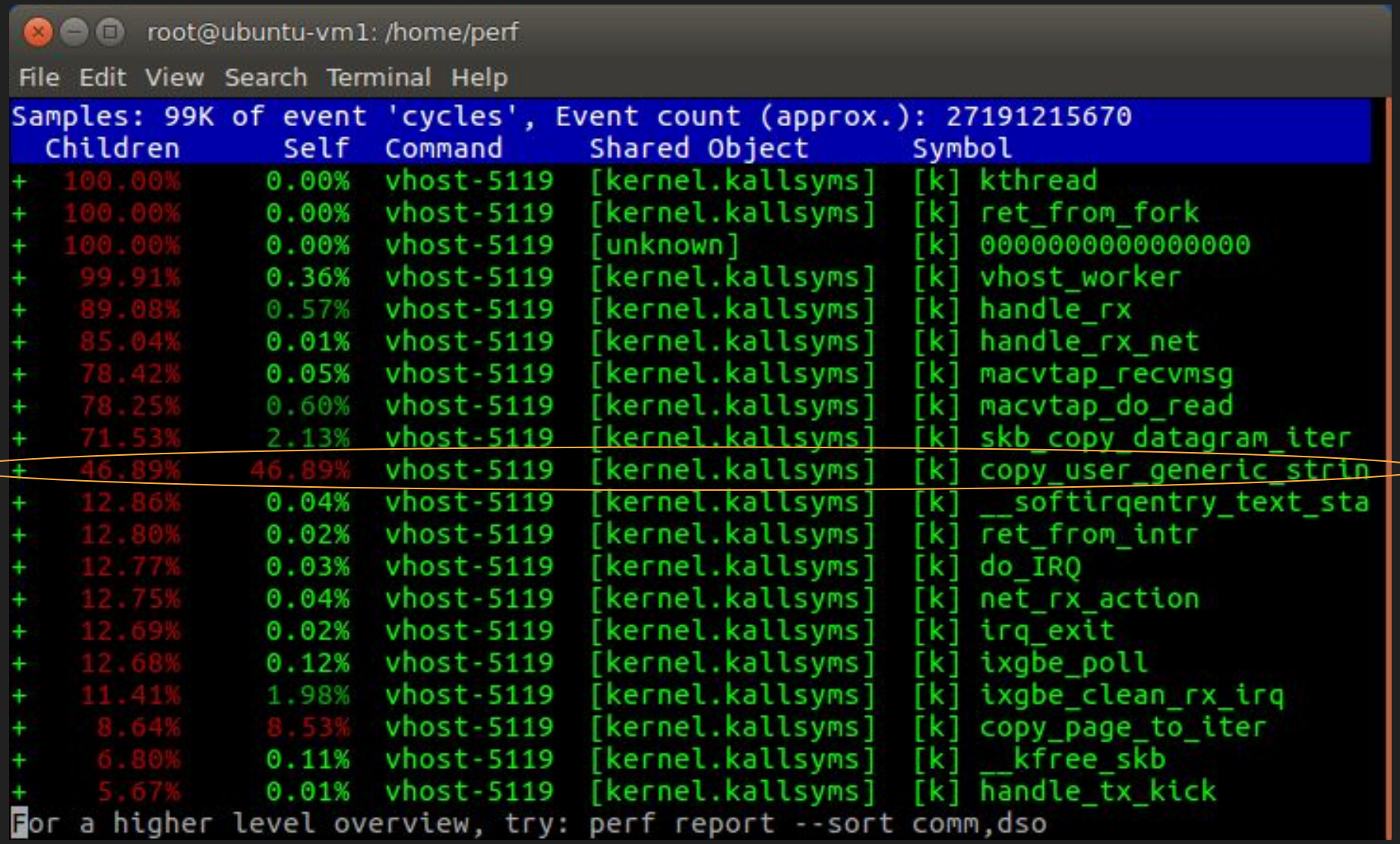
Virtualization and IO



Motivation

- No copy is better than copy
- Zerocopy TX without RX should feel lonely
- It was 7 years since the last attempt. Can we do better?

More motivation



Zerocopy: TX vs RX

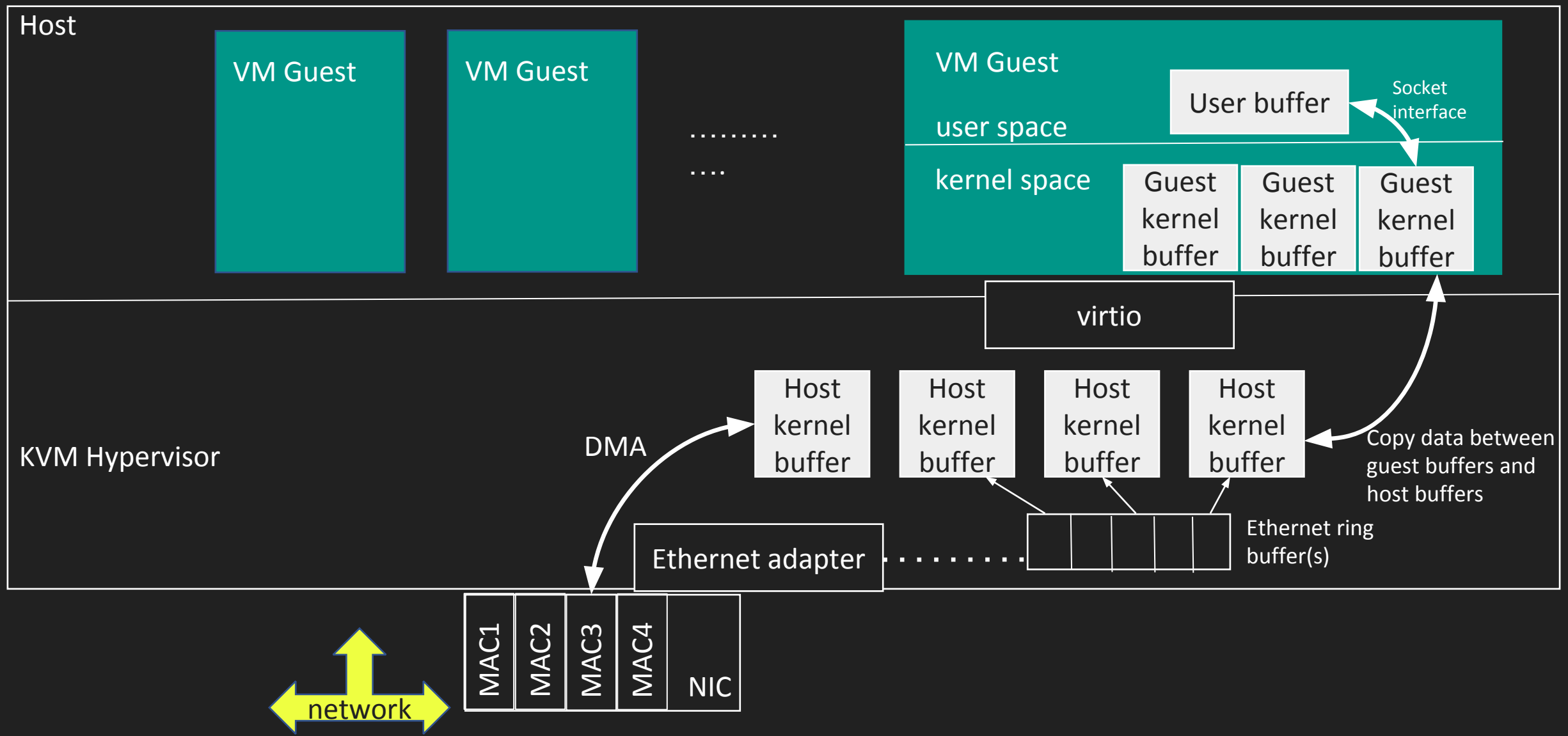
Transmit

- Downstream routing is easy
- Memory is always at hand
- ~~Still has problems~~
  - ~~head-of-line blocking~~

Receive

- Destination is not yet known
- Need memory for DMA
- Does not exist yet

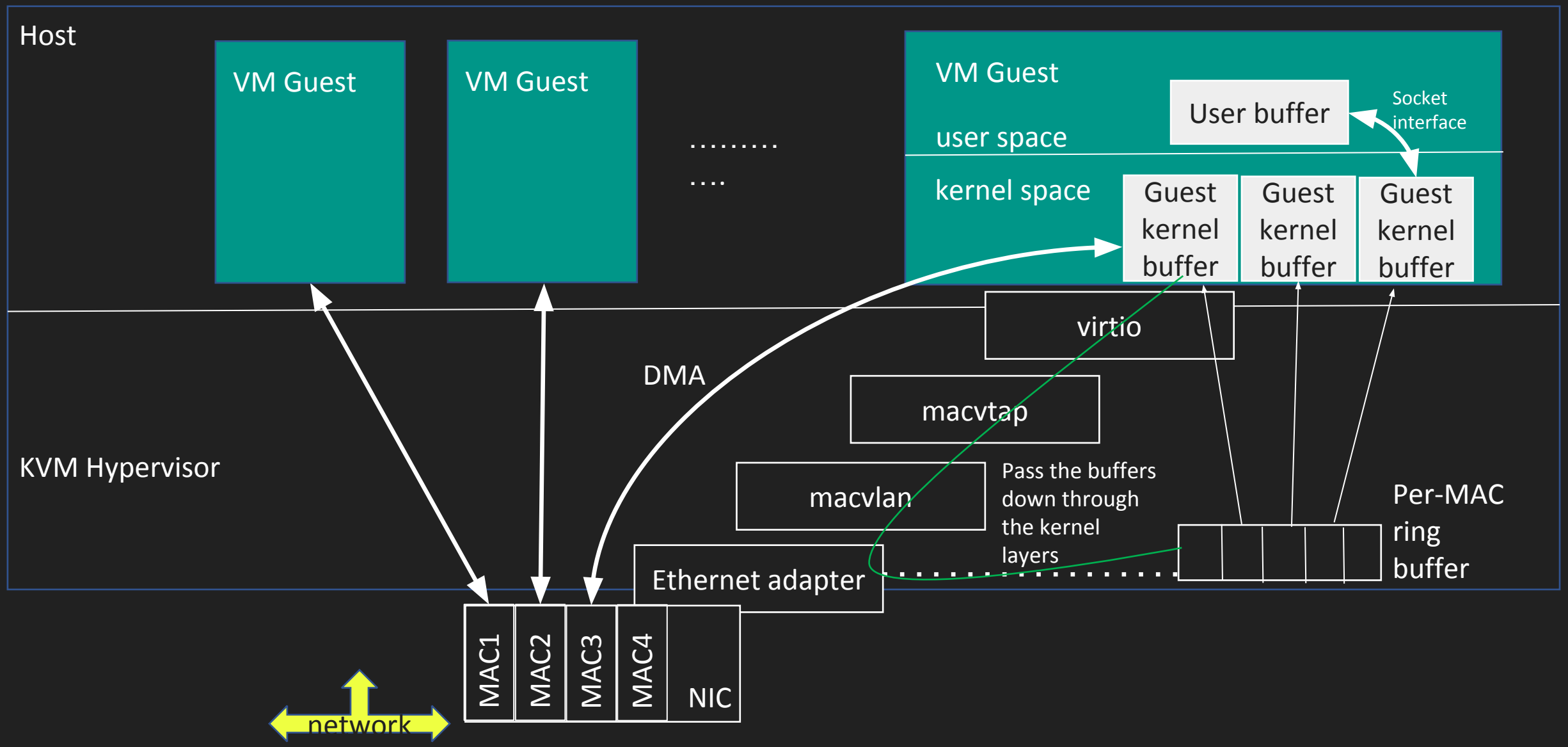
Virtio RX path



Assumptions

- Modern NICs are multiqueue
  - Dedicate queues to virtual NIC
- Guest allocates the buffers
  - Remapping DMA region to guest is more complex
- Tight coupling between physical and virtual NICs
  - Restrict zerocopy-RX to macvtap

Zero-Copy Rx Architecture



Initialization

- Isolate set of queues in physical NIC
- Create 1:1 correspondence between physical and virtual queues
- Clear RX descriptor ring
- Drop pre-allocated RX buffers in physical NIC driver

Memory allocation

- virtio-net (guest)
  - Allocate buffers
    - DMA'able memory (`PAGE_SIZE` granularity and page aligned)
- vhost-net
  - Post buffers to macvtap
    - New control flag `MSG_ZCOPY_RX_POST` for `macvtap_recvmsg()`
- macvtap
  - Allocate `skb`
  - Map `iovec` to `skb` (similar to `zerocopy_sg_from_iter`)
  - Pass the buffers to physical NIC
    - New method `ndo_post_rx_buffer()`
- Physical NIC driver adds new buffers to RX descriptor ring

Packet receive

- Physical NIC driver
  - DMA directly to the guest buffers
  - Setup skb structure
  - `netif_rx()` and friends
- macvtap
  - Queue skb as ready for the userspace
  - Inform vhost-net about the virtio descriptor associated with the `skb`.

Packet receive (cont)

- vhost-net
  - `handle_rx_zero_copy()`:
    - Update virtqueue
    - Kick macvtap with `->recvmsg(MSG_ZCOPY_RX)`
- macvtap (again)
  - `macvtap_do_read_zero_copy()`
    - skb_array_consume
    - Cleanup

API changes

netdev

- `->ndo_set_zerocopy_rx(struct net_device *pdev, struct net_device *vdev)`
  - Pass vdev down the stack to the ethernet adapter to bind physical and virtual queues.
  - Similar to `->ndo_dfwd_add_station()`, maybe just add flags there…
- `->ndo_post_rx_buffer(struct net_device *dev, struct sk_buff *skb)`
  - Passes a single (page aligned) buffer to the ethernet adapter
  - `skb` contains pointer to the upper level device and `ubuf_info`

API changes (cont)

macvtap

- `MSG_ZCOPY_RX_POST`
  - Control message from vhost-net to macvtap to propagate the buffers from guest to the lower levels
- `MSG_ZCOPY_RX`
  - Flag indicating that message contains preallocated buffers that should not be copied to userspace

API changes (cont)

virtio-net

- `add_recvbuf_full_page()`
  - Ethernet adapter driver expects page size aligned buffers
  - Existing `add_recvbuf_*()` do not care since the data was always copied

Issues

- No unified mechanism for RX memory allocation
  - Each driver wraps `alloc_page()` differently
  - Page pool is a savior?
- What to do when running out of buffers
  - Drop?
  - Fallback to copy?
- virtio does not know how to deal with fragment offsets
  - `skb_copy_datagram_iter()` takes care of `frag->page_offset`
- Allocating page for Ethernet frame is wasteful
  - Even worse for architectures with `PAGE_SIZE > 4K`

Implementation status

- Progress is slower than expected
- Initial implementation for ixgbe, plans to add other NICs later on
- Zerocopy packets reach the guest :-)
- Not stable enough to run benchmarks :-(

Thank you!