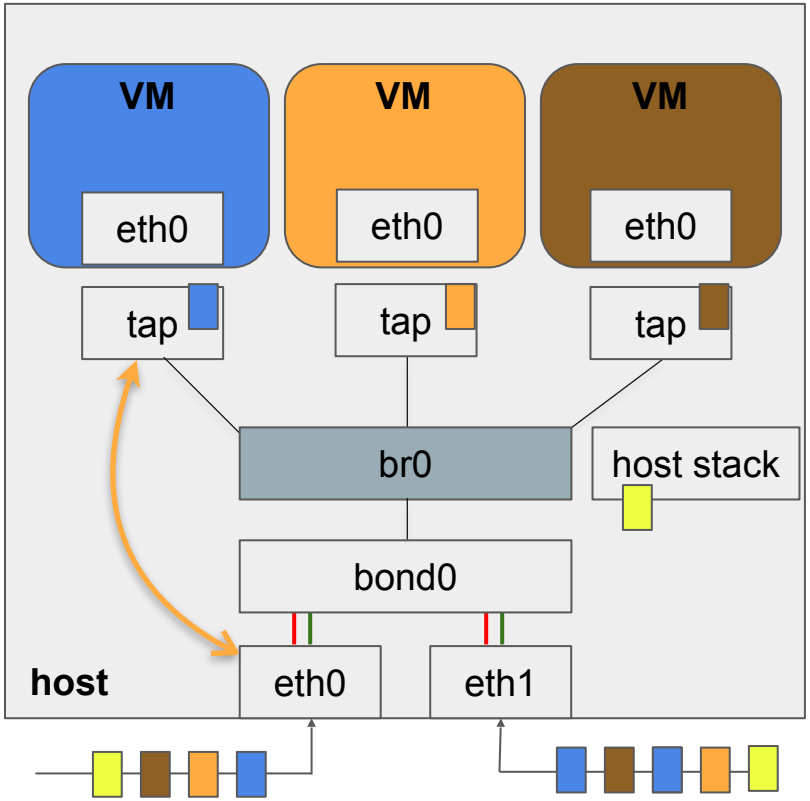
XDP and the Cloud

Scope of Tutorial

- Hypervisor networking
  - Move packets from network to VM and VM to network as fast as possible
- XDP for moving packets
  - co-exists with full stack
  - gotchas
- What is needed to use XDP in VMs



Packet Processing with XDP

- Bypasses skb allocations and host networking stack (e.g., bridge)
- Can have a similar XDP program on the tap device to redirect to the egress NIC
- HOW the redirect decision is made is up to the ebpf program
  - L2 / FDB lookup - tap into bridge fdb (need bpf helper)
  - L3 / FIB lookup - existing fib_lookup helper
  - Map with static <dmac,vlan> to next device mapping
    - *This tutorial focuses on this option*
- Per-packet and per-VM decision
- XDP programs can also do packet validations (VMs are untrusted), ACLs and crude bandwidth limiting

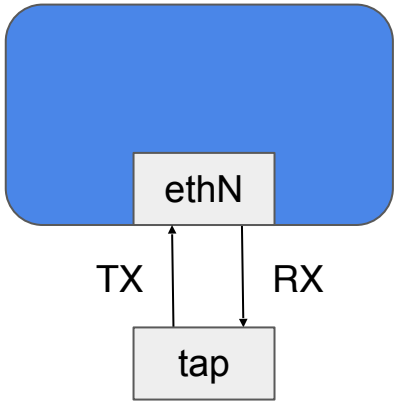
Host Setup

- v5.8 kernel, Ubuntu 18.04 OS
- Mellanox ConnectX-4 Lx NIC (25G) using mlx5 driver
- Typical host networking configuration
  - NIC ports into 802.3ad (LACP) bond
  - L3+L4 hash for Tx
  - OVS bridge
- VM can be on one or more networks
  - e.g., Public, private
  - VLANs for network separation - invisible to the VM
- VM networking is tap + vhost

VM Networking

- Directions for tap device:
  - Tx = Packets **to** the VM
  - Rx = Packets **from** the VM

- XDP on tap device means processing packets egressing a VM

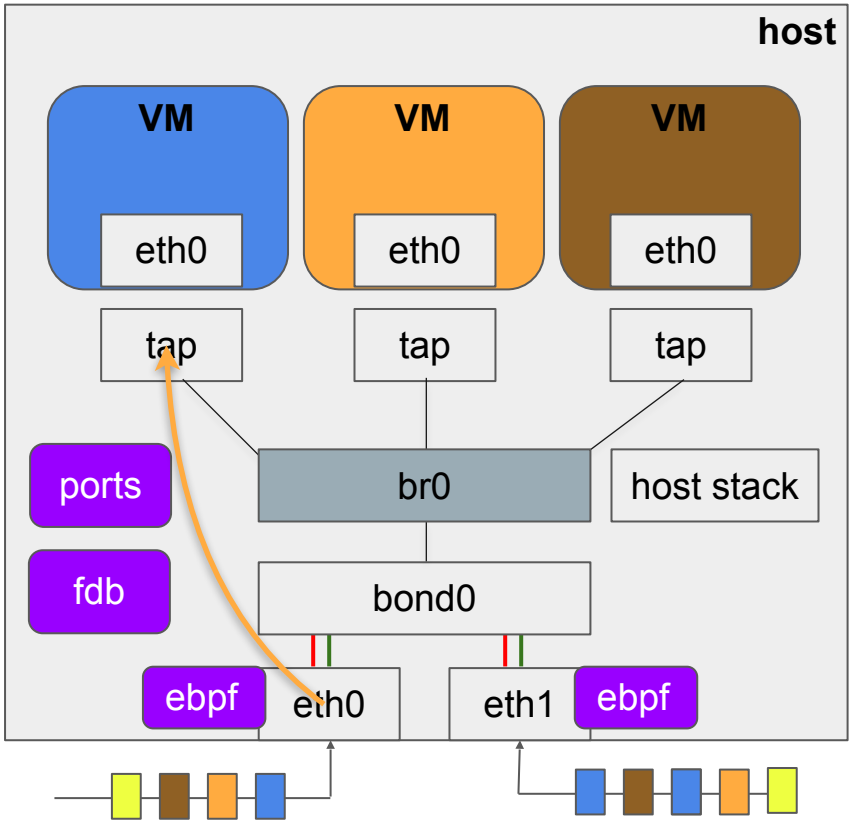


Host Ingress Traffic

DigitalOcean

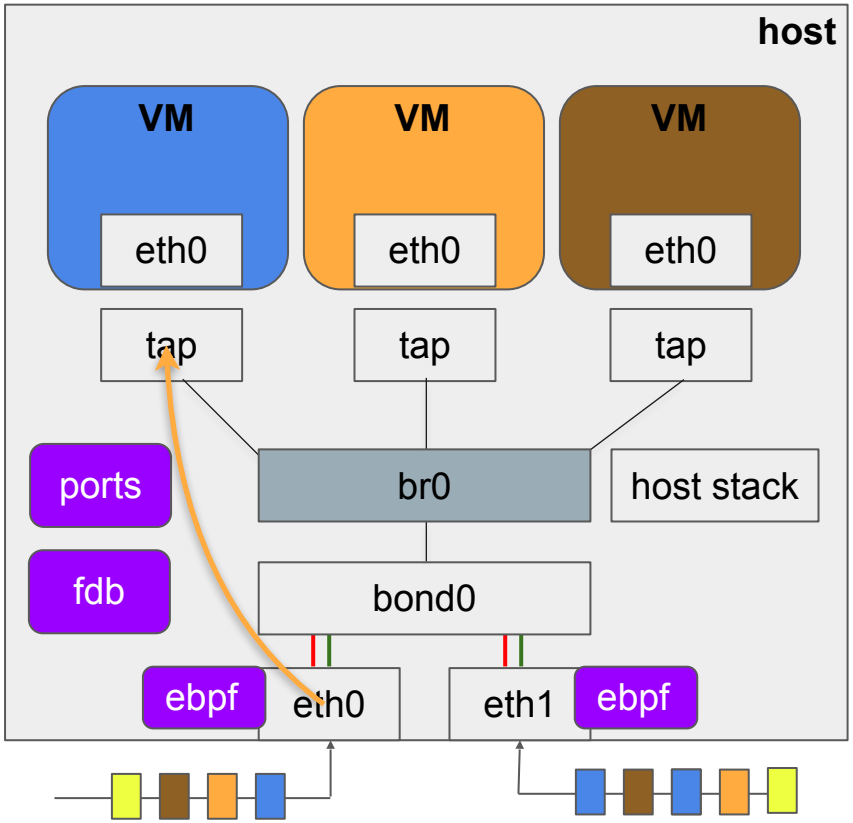
Layer 2 Forwarding in XDP

- Layer 2 forwarding program on ingress NICs
- Very simple virtual switch
  - Pulls vlan, dest mac from ethernet header
  - Looks up next device for packet
- FDB map
  - key: <vlan, dest mac>
  - value: device index
- Ports map



Layer 2 Forwarding in XDP

- Basic Premise:
  - Known traffic takes the fast path (XDP)
  - BUM traffic uses full stack
- "Known traffic" defined as entry exists in FDB map
- Goal is for 90+% of traffic to take the fast path

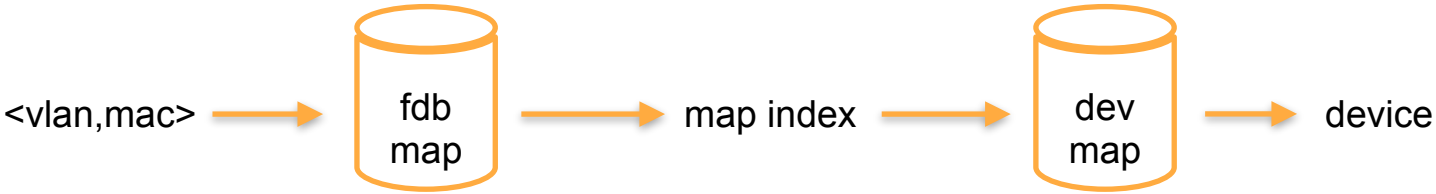


FDB as a Hash Map

- Hash Map (BPF_MAP_TYPE_HASH)
- MAC addresses are relative to a VLAN
  - Key: <vlan, mac>
  - Value: index
- Userspace updates FDB map as VMs start and stop
  - e.g., hypervisor agent or control script

Device Map - BPF_MAP_TYPE_DEVMAP

- XDP_REDIRECT requires next device for packet
  - Best performance requires use of bpf_redirect_map()
- DEVMAP is a pointer to netdevices
- Simplest design is for DEVMAP index == netdevice index
  - FDB lookup returns device index
- Map sizes declared up front - at load time
  - tap device index can grow over time
- Better approach is for FDB lookup to return index into DEVMAP
  - Needs coordination between maps



Device Map - BPF_MAP_TYPE_DEVMAP_HASH

- New in v5.4
- Avoids the need to manage mappings between fdb and dev maps
  - Map index can be device index regardless of device index value and size of map

Redirect with Kernel v5.6

- v5.6 performance of bpf_redirect is on par with map variant
  - No need for port map - or coordination between maps
  - FDB lookup can return tap device index
  - Loss of direct error reporting

Layer 2 Forwarding in XDP

- https://github.com/dsahern/bpf-progs/blob/master/ksrc/xdp_l2fwd.c
- Very simple virtual switch like processing

XDP and VLANs

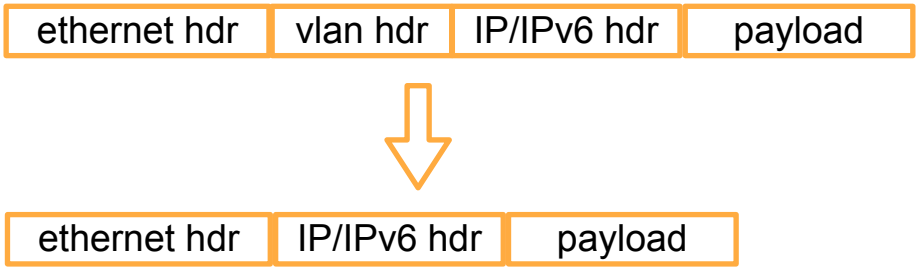
- Some NICs have VLAN acceleration enabled by default
  - mlx5 and i40e
  - sfc may - mixed results
- VLAN header stripped in hardware
  - VLAN tag accessed via descriptor, set in skb

- Does not work with XDP
  - Buffer accessible via XDP context does not have vlan hdr
  - BPF program expecting VLAN based decision will fail

XDP and VLANs

- Disable VLAN acceleration
  - ethtool -k <DEV> rxvlan off

- XDP program needs to strip VLAN header

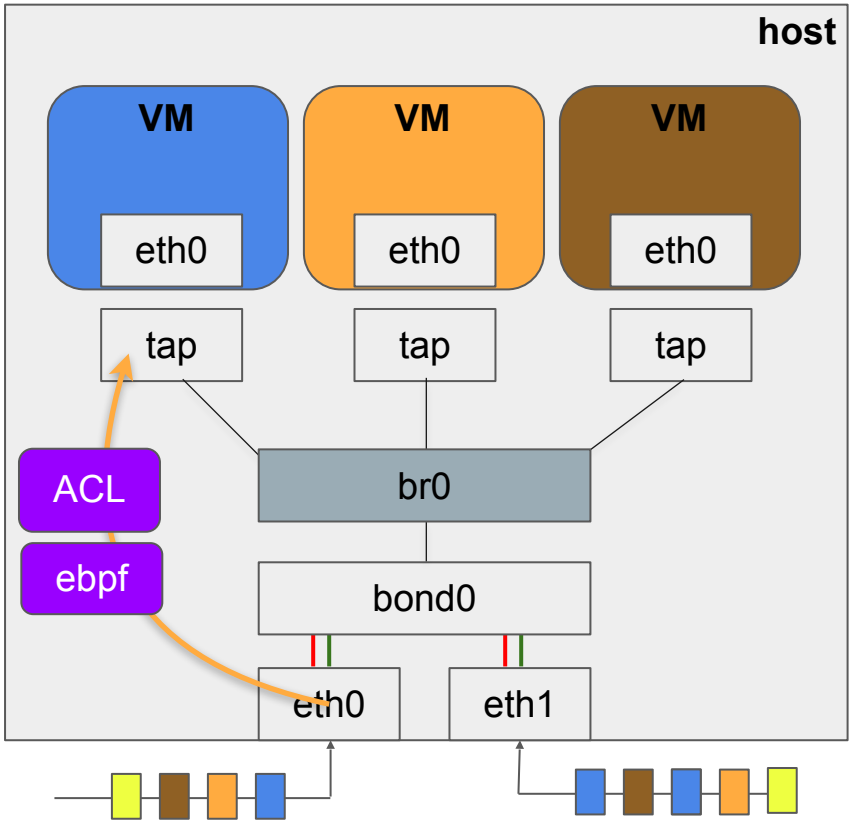


ACL at VM Ingress

DigitalOcean

Host ACL for Packets to VM

- Host side, VM-specific "ingress" ACL
  - Looks up VM data in VM info map
  - VM specific ACL map
- "Tx" path for the tap device
  - No "Tx" XDP option
- BPF program attached to DEVMAP entry



BPF program on DEVMAP Entries

- v5.8 allows programs to be attached to DEVMAP entries
- Program is run on XDP_REDIRECT
- Context has both Rx device and Tx (redirect) device
- net device delete removes map entry and any program

VM Ingress ACL Example

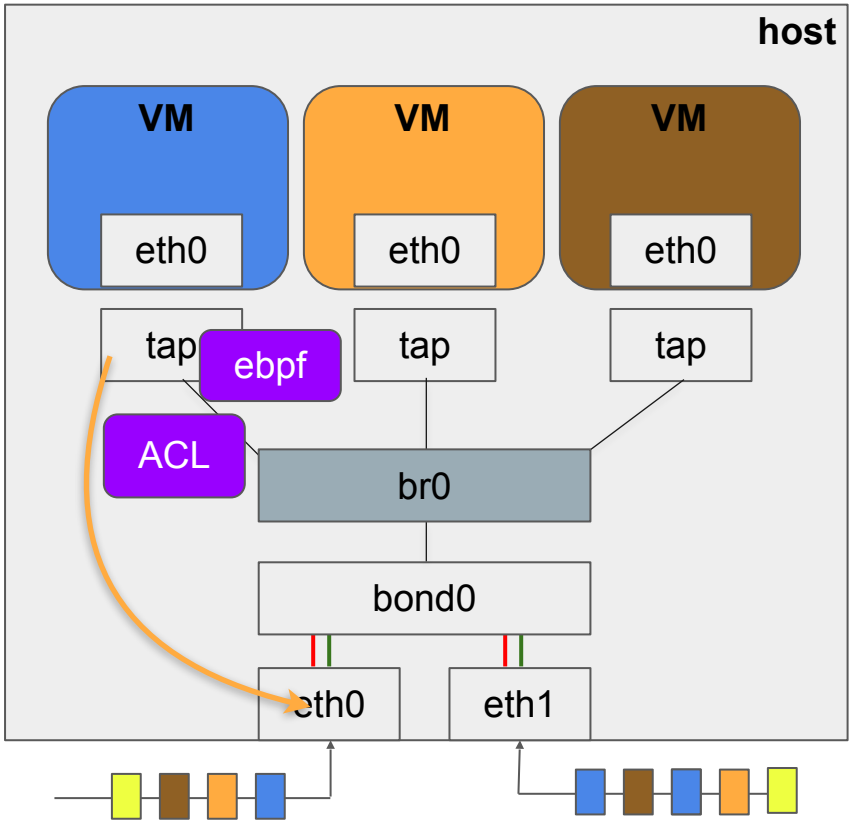
- https://github.com/dsahern/bpf-progs/blob/master/ksrc/acl_vm_tx.c
- https://github.com/dsahern/bpf-progs/blob/master/ksrc/acl_vm_common.h
- https://github.com/dsahern/bpf-progs/blob/master/ksrc/flow.h

VM Egress Traffic

DigitalOcean

Handling Traffic from VM

- Packets *from* VM
  - Rx for tap device
- Program attached to tap device
  - Validate packet (e.g., source mac, network address)
  - Egress ACL for VM
- Redirects packet to host device
  - Could redirect to another VM if allowed
- BUM traffic takes the bridge path
- Program and map cleaned up when tap device is deleted



Redirecting VM Egress Traffic

- Egress through bond
  - Need to specify bond port in redirect
  - Which leg of bond to use?
  - Bond policy: active/backup, L3+L4 hashing?
- BPF helper?
  - it's complicated
- One option is to put that knowledge in the ebpf program and maps
  - active leg of bond or computing hash and picking leg
  - https://github.com/dsahern/bpf-progs/blob/master/ksrc/xdp_vmegress.c#L43

Redirecting VM Egress Traffic

- Egress through bond
- VLAN tag
  - No VLAN acceleration for XDP_REDIRECT and Tx
  - XDP program on tap device needs to insert VLAN header

Redirecting VM Egress Traffic

- Egress through bond
- VLAN tag
- Tx offloads in guest
  - Host is the "hardware" + "network" to the guest
  - Do not work with XDP - no hardware offload support
  - libvirt config:

```
<driver name='vhost'>
    <host tso4='off' tso6='off' ecn='off' ufo='off' csum='off'/>
</driver>
```

Redirecting VM Egress Traffic

- Egress through bond
- VLAN tag
- Tx Checksum offload in guest
- Number of Rx/Tx queues needs to be equal to the number of CPUs
  - ethtool -L eth0 combined $(nproc)
  - ethtool -L eth1 combined $(nproc)
  - Does not work for large systems - more cpus than queues!
    - Can tell you from experience, it is baffling to debug: some packets work fine and others are dropped when vhost thread migrates to cpu that does not have a queue
    - Affine vhost threads to CPUs with associated queues
      - /proc/interrupts

VM Egress Example

- https://github.com/dsahern/bpf-progs/blob/master/ksrc/xdp_vmegress.c

Demo Time

Demo

- https://github.com/dsahern/bpf-progs/blob/master/scripts/l2fwd-demo.sh

Using XDP in VMs

XDP in VMs

- Two common problems for KVM with virtio_net
  - Machine type
  - vhost threads and queue requirements

XDP in VMs: Machine type

- This error:

  $ bpftool net attach xdp id 14 dev eth0

  Kernel error message: virtio_net: XDP expects header/data in single page, any_header_sg required

- Machine type is too old (e.g., pc-i440fx-1.5)
- Qemu needs to use a modern machine (e.g., pc-i440fx-4.1)
  - <type arch='x86_64' machine='pc-i440fx-4.1'>hvm</type>

XDP in VMs: NIC queues

- This error:

  $ bpftool net attach xdp id 13 dev eth0

  Kernel error message: virtio_net: Too few free TX rings available

- Number of queues on the tap device needs to be 2 * Nvcpu
  - e.g., 16 vcpu VM needs 32 queues

    <model type='virtio'/>

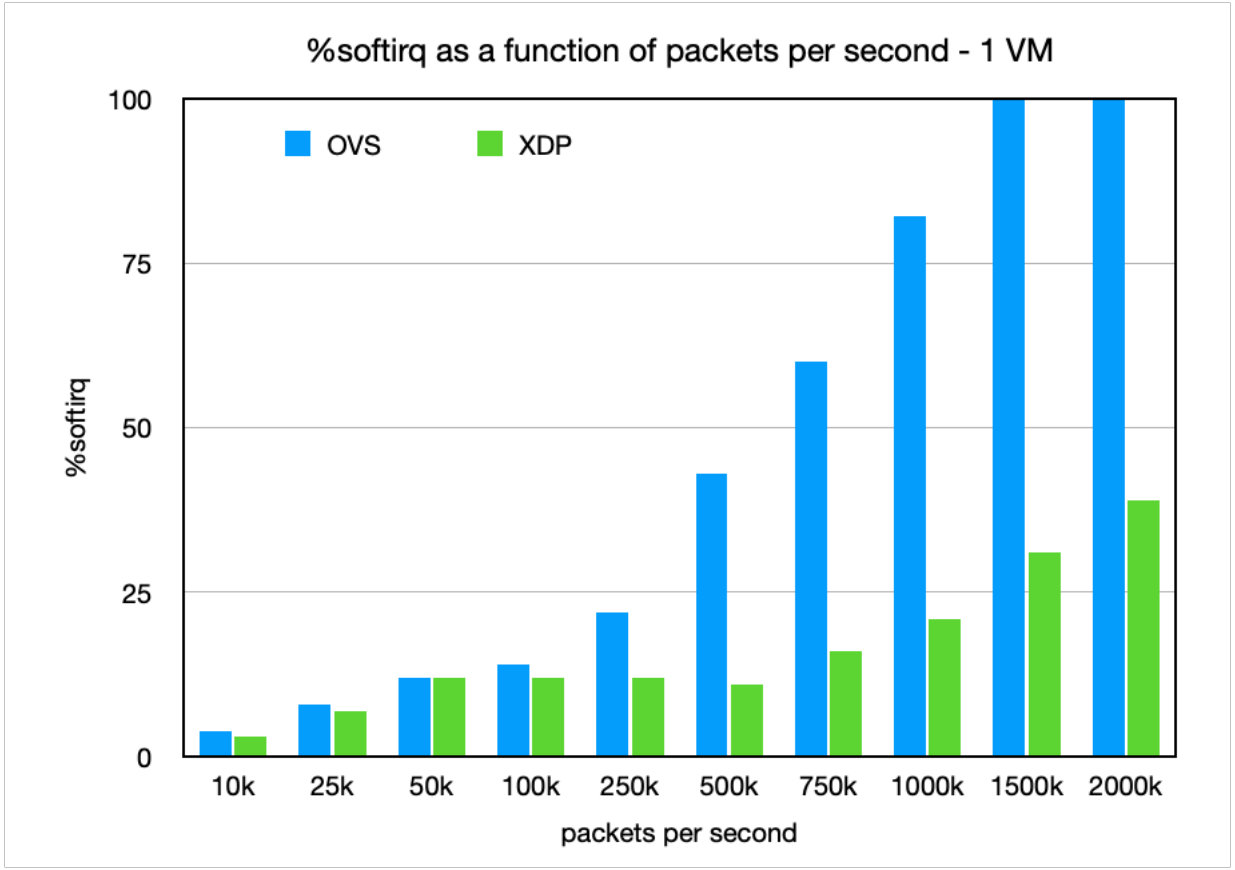
    <driver name='vhost' queues='32'/>
- Applies to **every** tap device an xdp program is to be allowed

XDP in VMs: H/W offloads

- Adding XDP program to VM netdevice disables Tx checksum and TSO on **tap** device in host
  - significant hit on guest Rx performance
- Impacts the ability to do performance measurements with an XDP program attached
  - e.g., pps to a VM

Final Comments

Motivation: XDP-vs-OVS Performance Comparison



Bypassing full stack means ...

- Loss of S/W RPS, RFS and ARFS
  - Relying on RSS in H/W to distribute packets across queues and cpus
- Bridge learning (if relevant)
- No hardware timestamps
  - e.g., PTP timestamping of packets in H/W


DigitalOcean