Veth XDP

XDP for containers

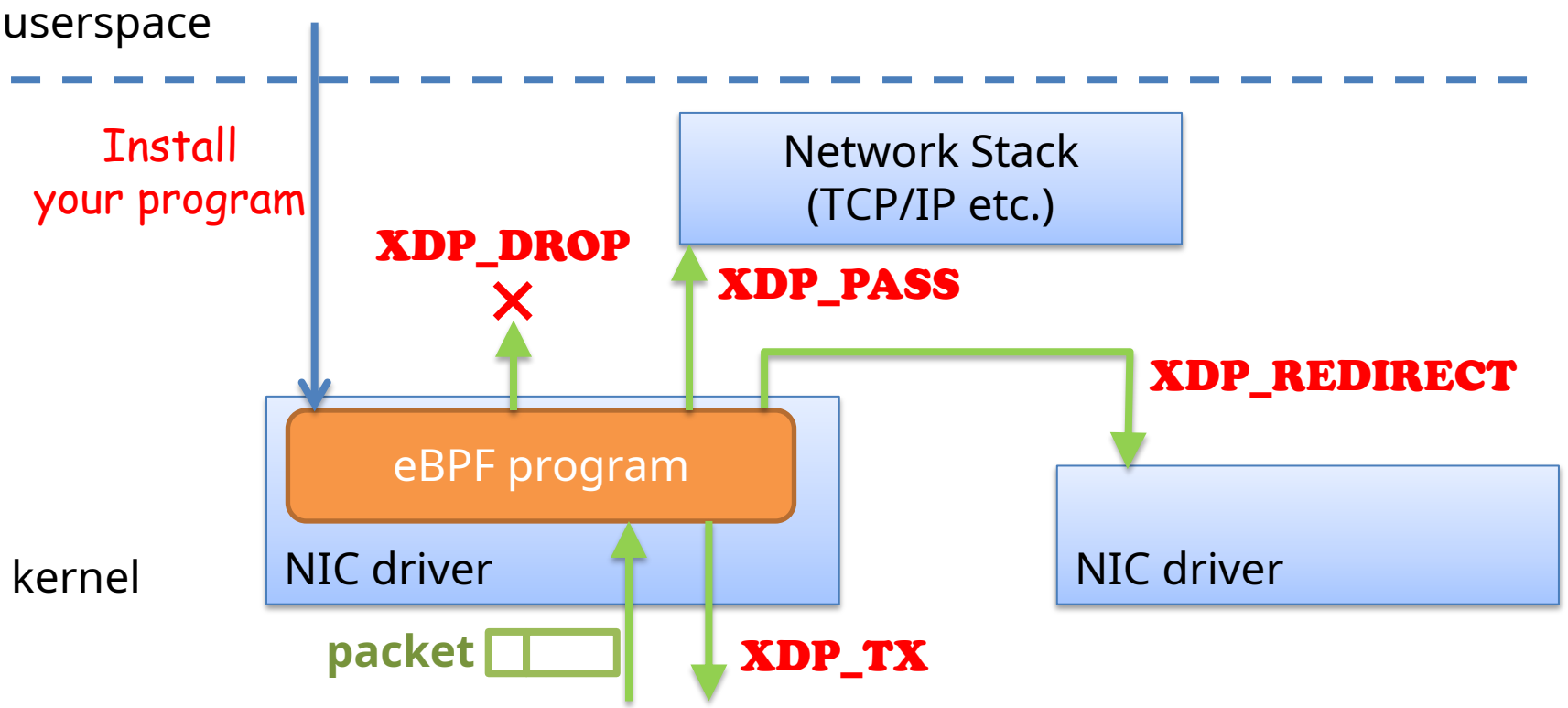
Toshiaki Makita (NTT Open Source Software Center)
William Tu (VMware)

Outline

- XDP support for veth
- AF_XDP support for veth
- Summary

XDP (eXpress Data Path)

- In-kernel fast (express) data path
- Execute your eBPF program at driver
  - Immediately after it receives packets
  - Low overhead: No metadata (skb) allocation



Generic XDP

- XDP requires implementation in each driver
  - Need to choose XDP-supported driver
  - Not so handy

- Generic XDP allows you to use XDP on any driver (kernel 4.12)
  - XDP implemented in network stack
    - Convert skb to xdp buffer
  - Not as fast as native (non-generic) XDP
    - Need skb allocation at drivers
    - Packet buffer copy to meet XDP requirements
  - Good for functionality testing, etc.

Generic XDP for virtual devices

- Generic XDP was extended to use on virtual devices (kernel 4.14)
  - including veth

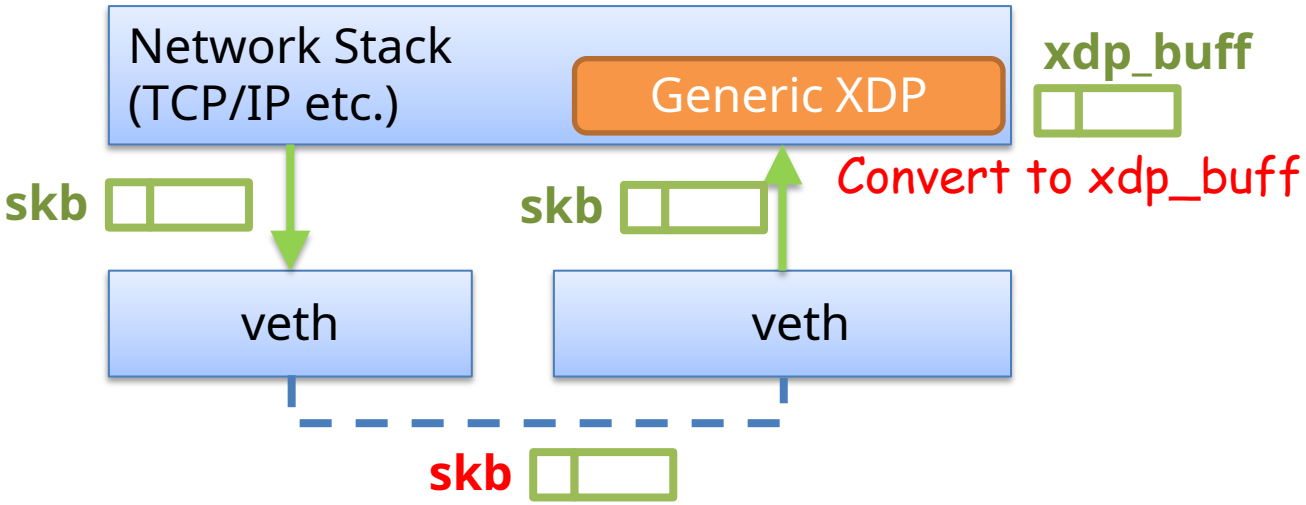
- Veth got (generic) XDP support

Native XDP support for veth

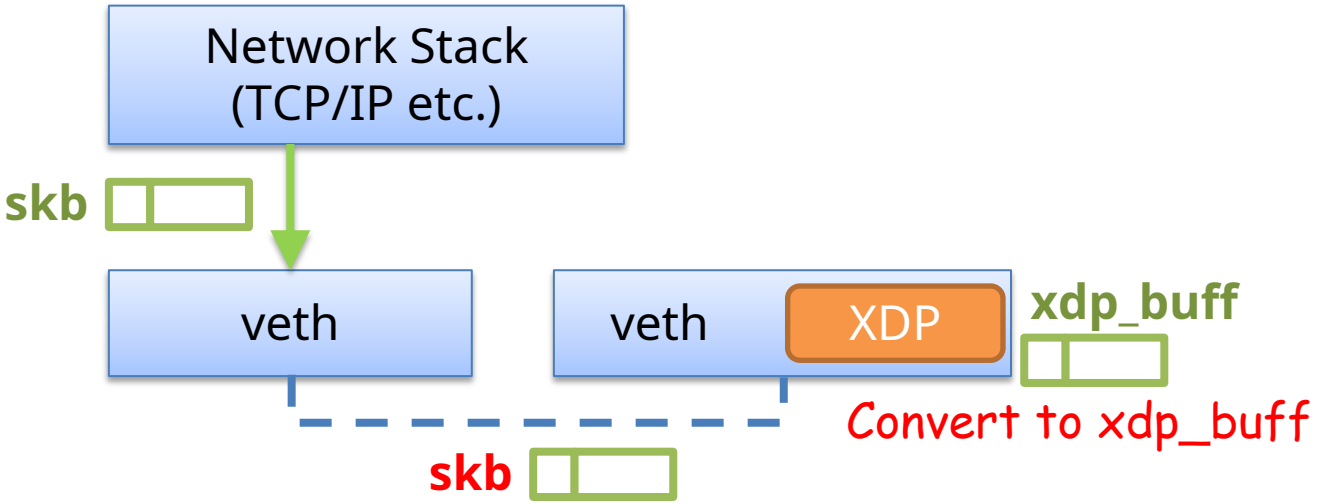
- Added in kernel 4.19
- Driver level implementation of XDP for veth

Veth XDP: Generic vs native

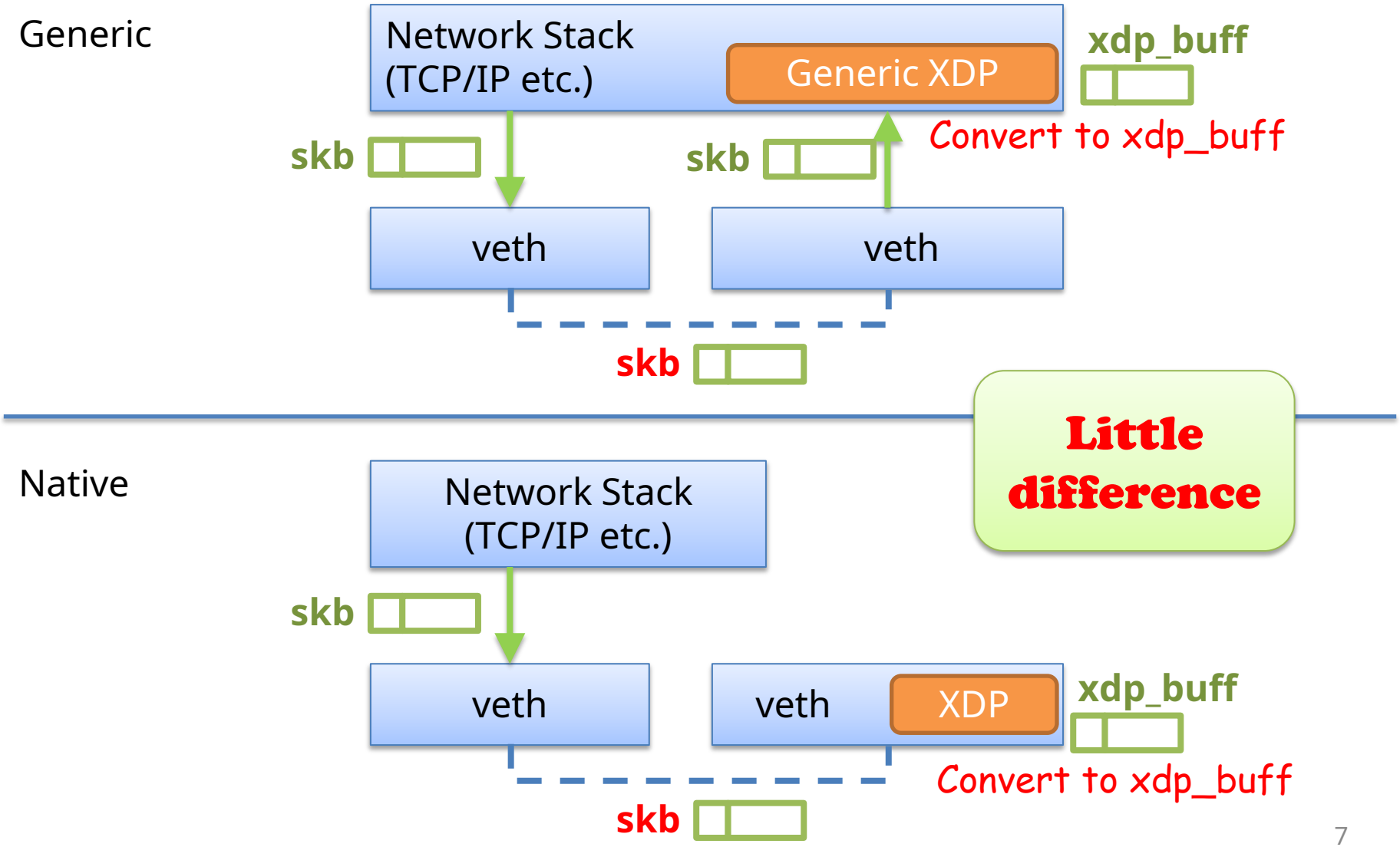
Generic



Native



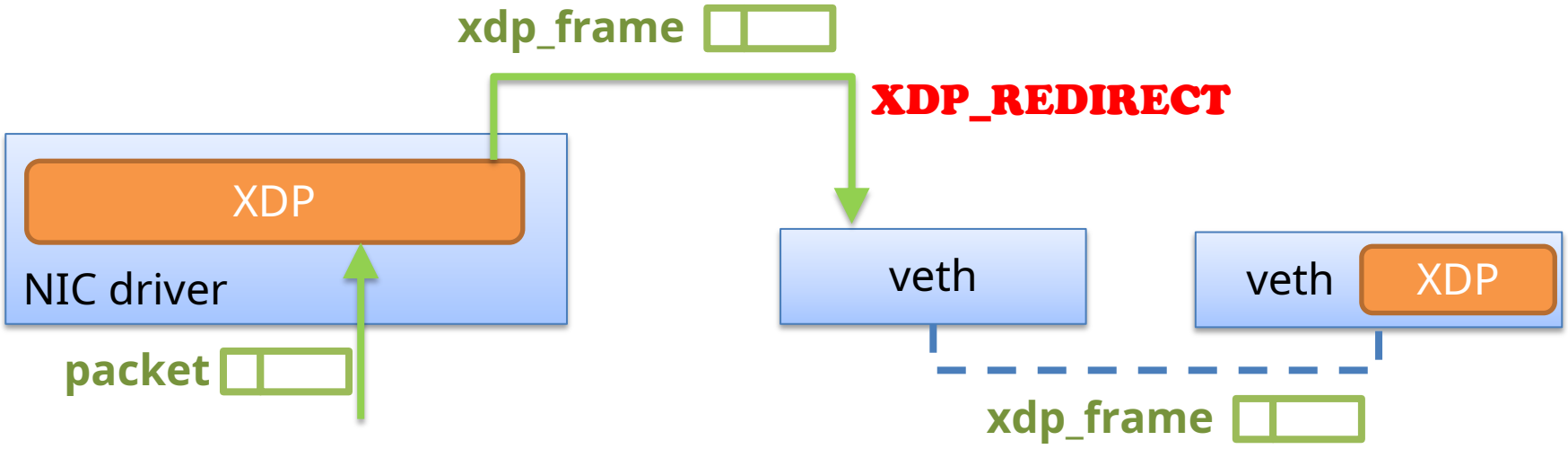
Veth XDP: Generic vs native



What's the point of implementing native XDP in veth?

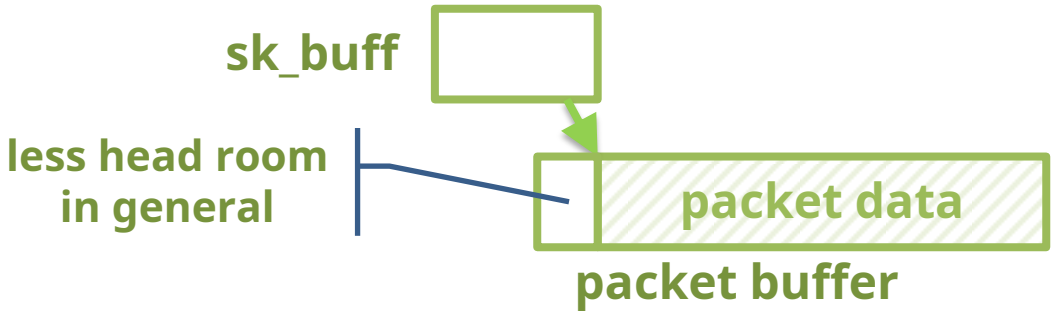
XDP_REDIRECT

- Physical NIC drivers can redirect packets to another interface **without** allocating sk_buff

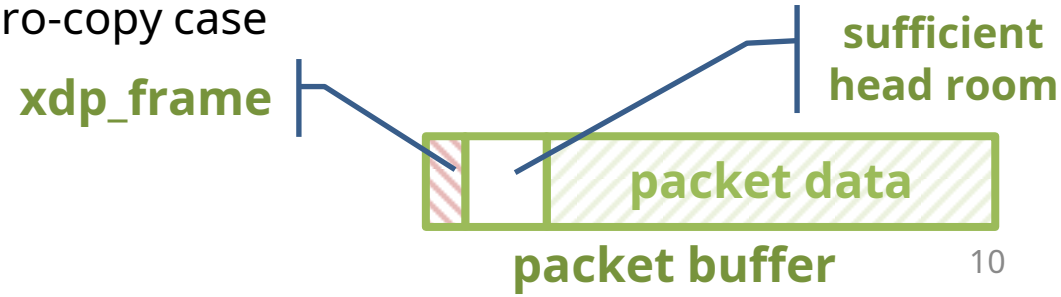


xdp_frame

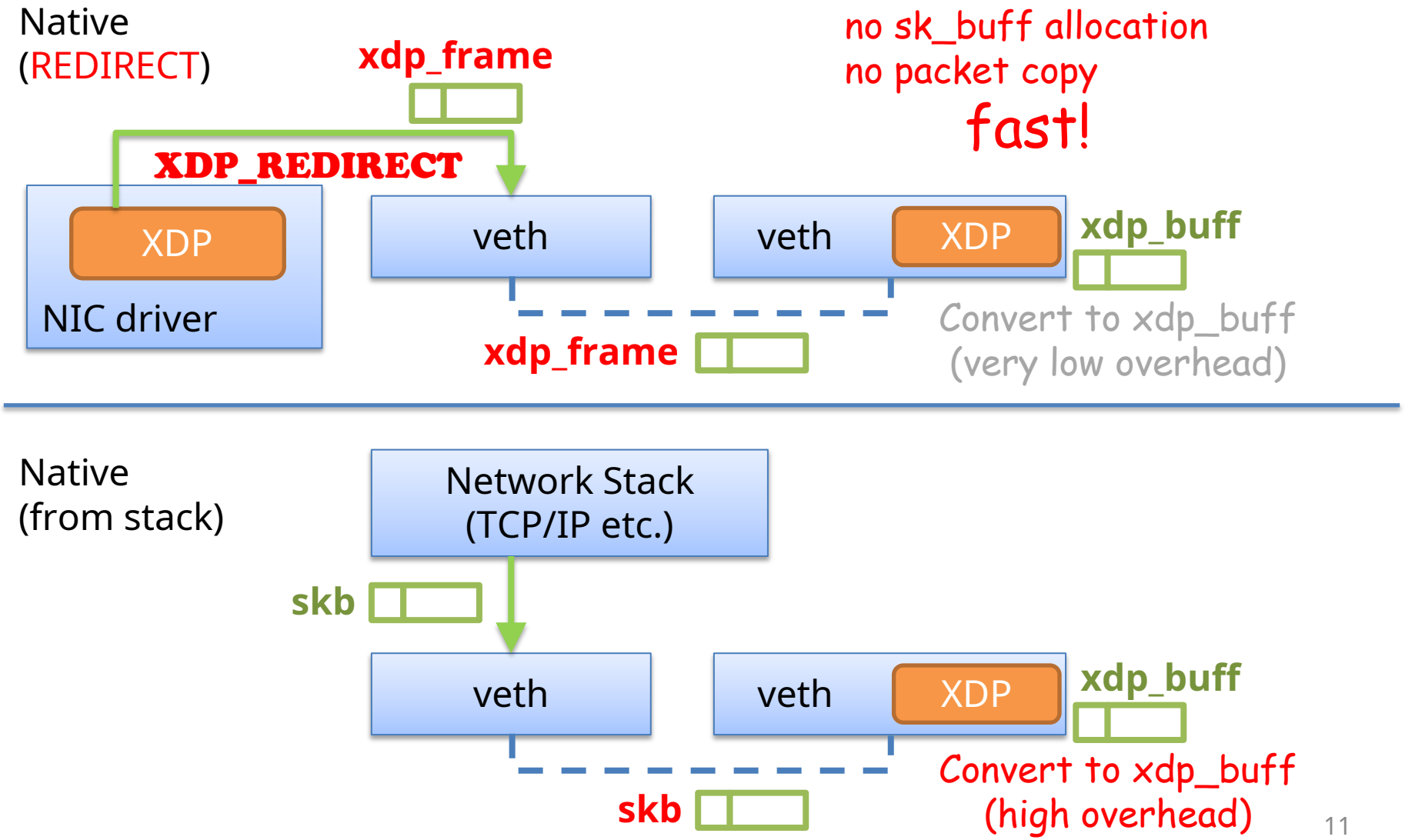
- sk_buff (traditional format)
  - Separate metadata object pointing to packet buffer
    - Need allocation/free
  - In many cases packet copy is needed to convert into xdp_buff (due to insufficient head room)



- xdp_frame
  - Reuses packet buffer head room
    - No metadata allocation/free
  - No need for packet copy to convert from/into xdp_buff
    - except for AF_XDP zero-copy case

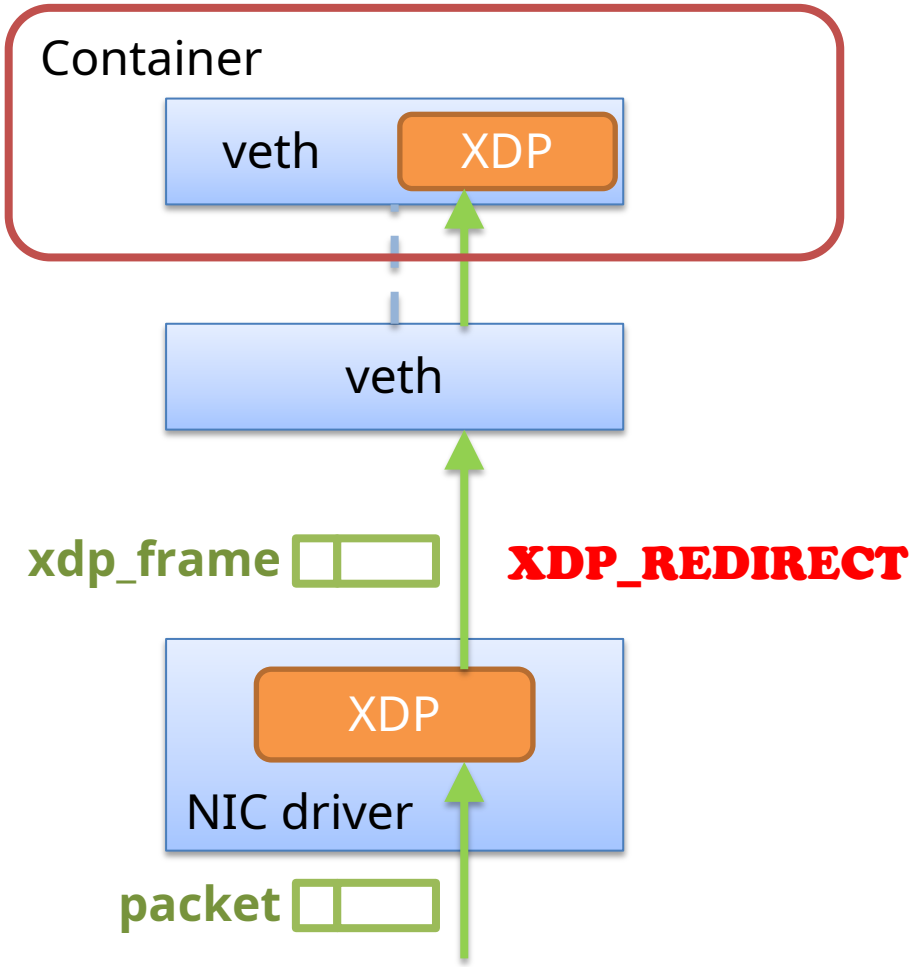


Veth native XDP again



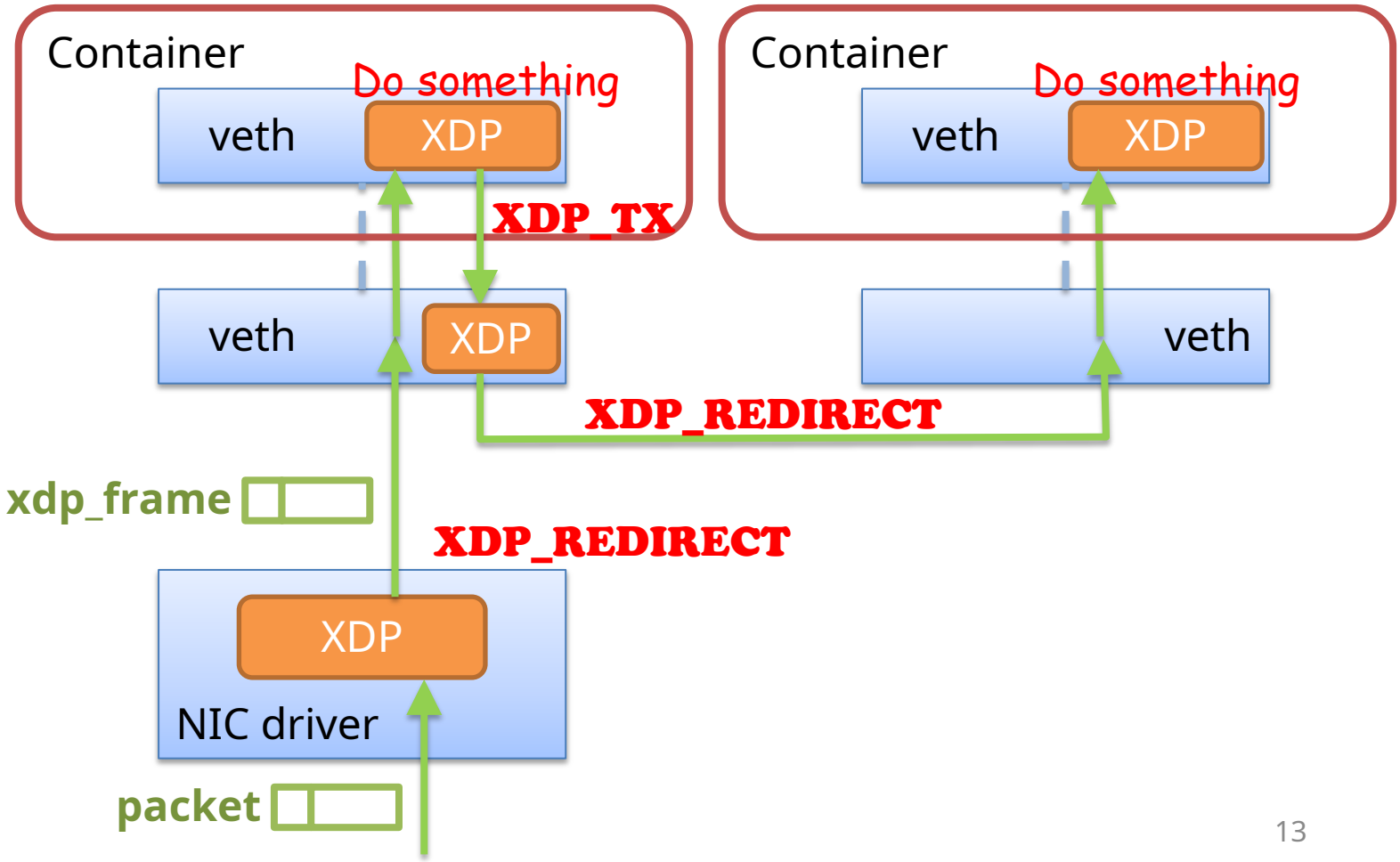
Use cases 1: containers

- XDP in containers



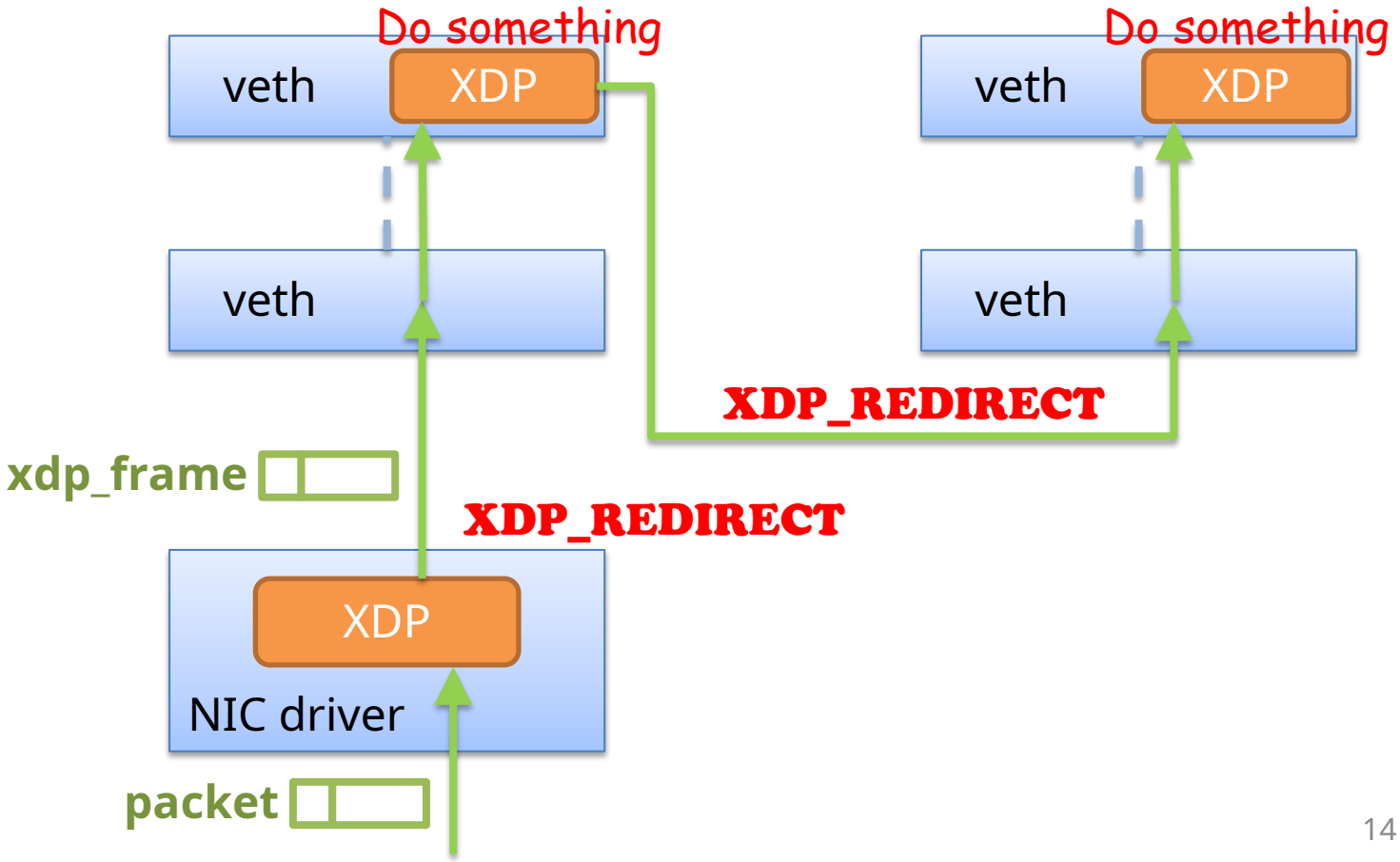
Use cases 1: containers

- XDP in containers: Service chaining



Use cases 2: program chaining

- Call another program from a program
  - by using XDP_REDIRECT

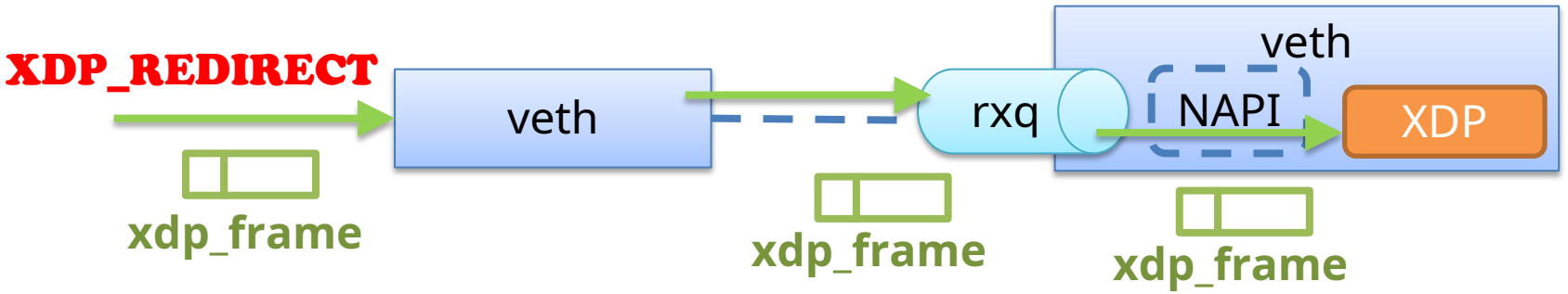


Design

- Original veth code used general softirq backlog to enqueue rx packets from its peer
  - Common routine handling only sk_buff
  - No point to call XDP program

Design (cont.)

- Veth native XDP added rx queues and NAPI handler
  - Entering this mode only when XDP is installed
  - Tx side enqueues packets into peer rx queue
  - NAPI handler on peer drains rx packets and runs XDP program
  - NAPI avoids infinite loop and stack overflow due to misconfigured XDP_REDIRECT chain



Usage

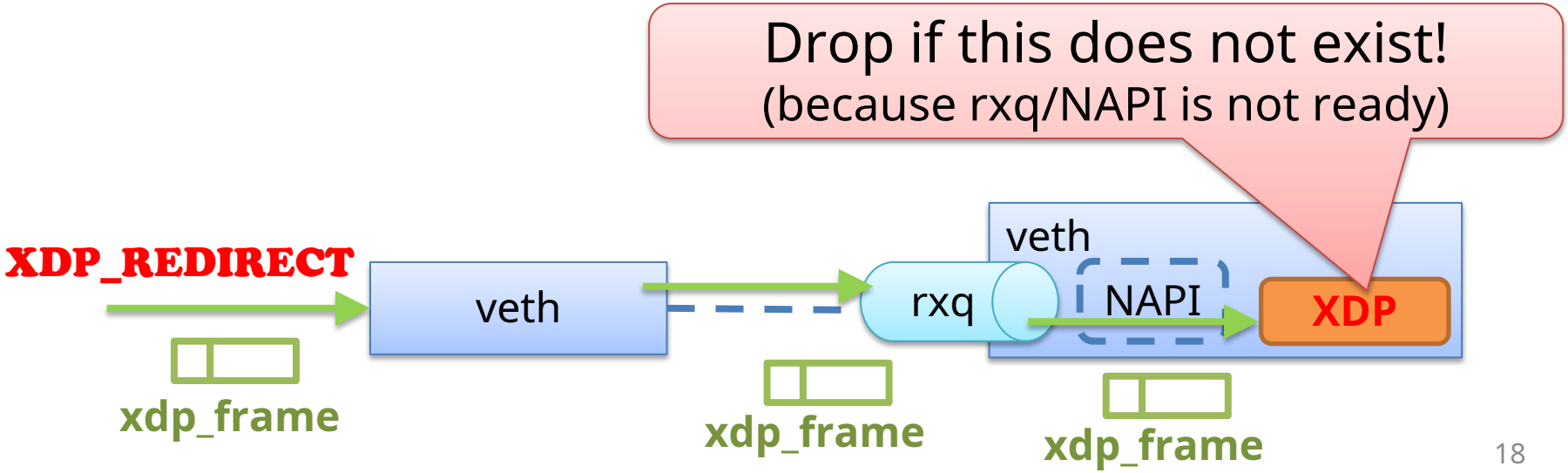
- Prerequisites
  - root user (Privileged containers!)
  - For the best performance allocate the same number of queues as cpus
  - Turn off vlan/tx checksum offloading features on related devices
    - Offloaded vlan or checksum is not visible from XDP
    - Currently veth does not automatically take care of them
  - Add unicast filter for veth rx side in phy interface

```
# Let's say the number of cpus is 20
# ip netns add ns0
# ip link add veth0 numrxqueues 20 numtxqueues 20 type veth \
> peer name veth1 netns ns0 numrxqueues 20 numtxqueues 20
# ethtool -K veth0 tx off txvlan off
# ip netns exec ns0 ethtool -K veth1 tx off txvlan off
# ethtool -K PHY_NIC rxvlan off
# bridge fdb add MAC_OF_VETH1 dev PHY_NIC self # Unicast filter
```

Usage

- Non-XDP_REDIRECT case (slow sk_buff path)
  - Just install XDP program on veth

- XDP_REDIRECT (and XDP_TX) case
  - Need to install XDP on peer as of kernel 5.0
  - Otherwise redirected packets will be dropped



Performance numbers

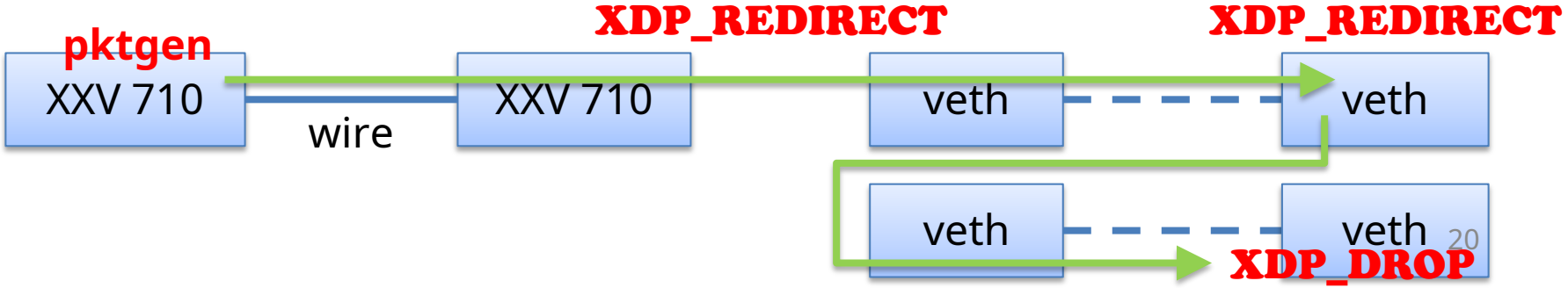
- Test environment
  - Intel Xeon Silver 4114 2.20 GHz 10 cores x2
  - Intel XXV710 (25G, i40e)
  - kernel 4.20.13

Performance numbers

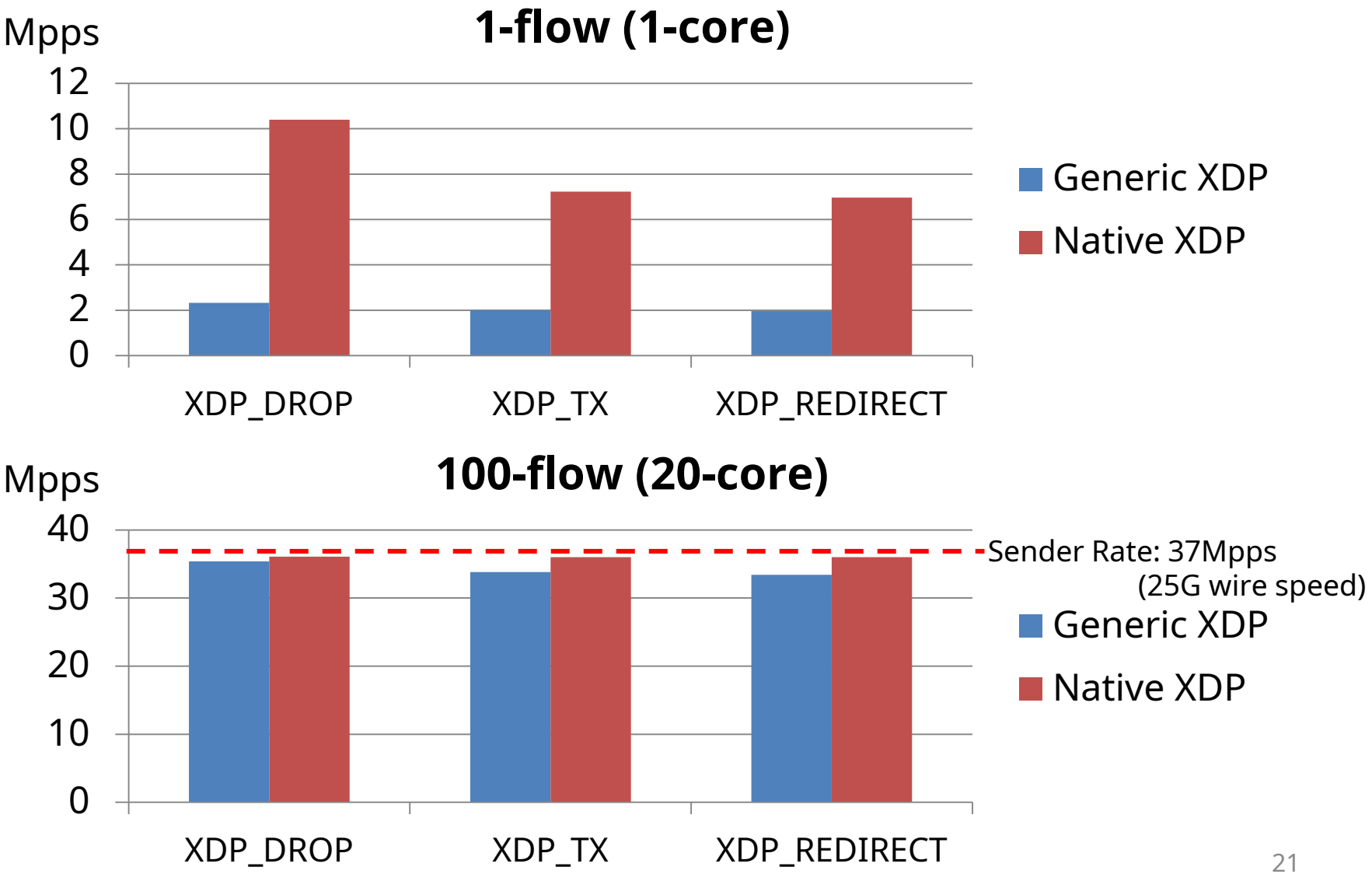
- Test patterns
  - DROP test

  - TX test

  - REDIRECT test



Performance numbers



Challenges

- Improve XDP_TX performance
  - XDP_TX lacks batch processing
  - Currently acquiring queue lock per packet
    - Private experimental patch shows 10% boost with batch
- More intuitive way to enable XDP_REDIRECT to veth
  - Installing XDP on peer is unintuitive
- Need a good XDP virtual switch implementation for containers
  - XDP_REDIRECT between phy and veth
  - OVS? p4c-xdp?

AF_XDP Support for veth

- AF_XDP
  - Allow redirecting raw XDP frames into userspace
  - Zero-copy mode directly DMA into user's buffer
    - Currently supported driver: i40e and ixgbe
- Use cases for vSwitch
  - Process packets from AF_XDP frames in userspace, and forward packets to other netdev/ports
  - Problems: no virtual port support AF_XDP zero-copy mode

AF_XDP OVS Use Cases

- OVS receives packets from a physical port/netdev using AF_XDP raw frame
- Flow entry decides to forward to a virtual ports, ex:
  - A veth port connecting another container
  - A tap/vhost port connecting another VM
- Without veth AF_XDP supports, packets have to copy/re-inject into the kernel
  - Performance drops significantly
  - RFC implementation for veth[1] and OVS[2]

[1] https://www.spinics.net/lists/netdev/msg542047.html
[2] https://patchwork.ozlabs.org/cover/1004837/

Some more thoughts about OVS XDP…

- AF_XDP pros/cons
  - Pros: Full flexibility by reusing OVS userspace datapath avoiding full re-implementation
  - Cons: Need packet copy in user-space when redirecting
- Another possible high-speed OVS idea: partially offload OVS to XDP
  - Support minimal set of actions
  - Unsupported flows are passed to upper layer openvswitch module (by XDP_PASS)
  - TC-like offload mechanism in ovs-vswitchd, or
  - Reuse TC offload mechanism
    - TC offload to XDP instead of HW
    - Use UMH to insert XDP program like bpfilter

Summary

- Veth native XDP support: available since kernel 4.19
  - Improve XDP performance in containers
    - 10Mpps with 1core
  - Can be used for service function chaining
- AF_XDP: work in progress
  - Can be used for OVS