Evolution of DSR implementation for containerized applications

Lalit Gupta
Pavel Dubovitsky
Raman Shukhau

Agenda

- Layer-4 Load balancers and Direct Server Return (DSR)
- L4 XDP Implementation
- XDP & Decapsulation
- Challenges for the heavily stacked containerized applications
- Evolution of the DSR support
- Lessons learned
- Q&A

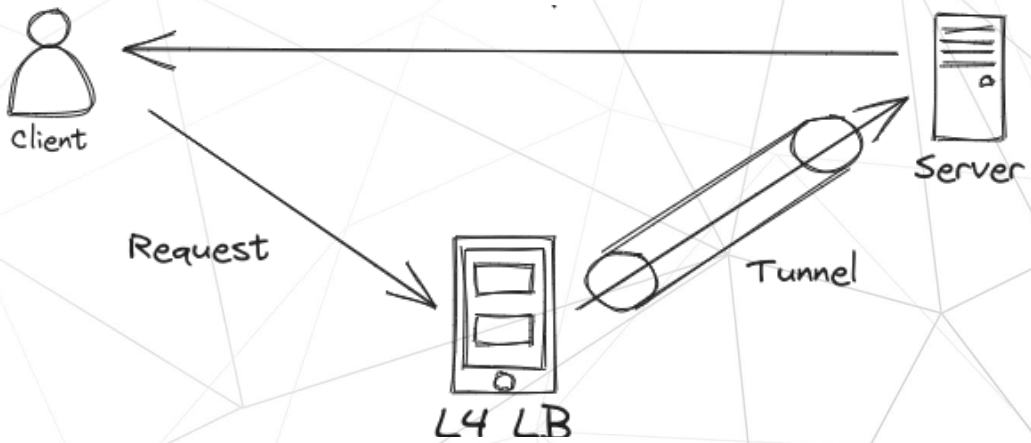
Layer 4 Load Balancer

Distributes network traffic based on information in the transport header across backends

- High Availability
- Scalability
- Session Persistence

Direct Server Return (DSR)

- Bypasses the LB on the way out
- Tunneling packets on the way in



Layer 4 Load Balancer

IPVS Implementation

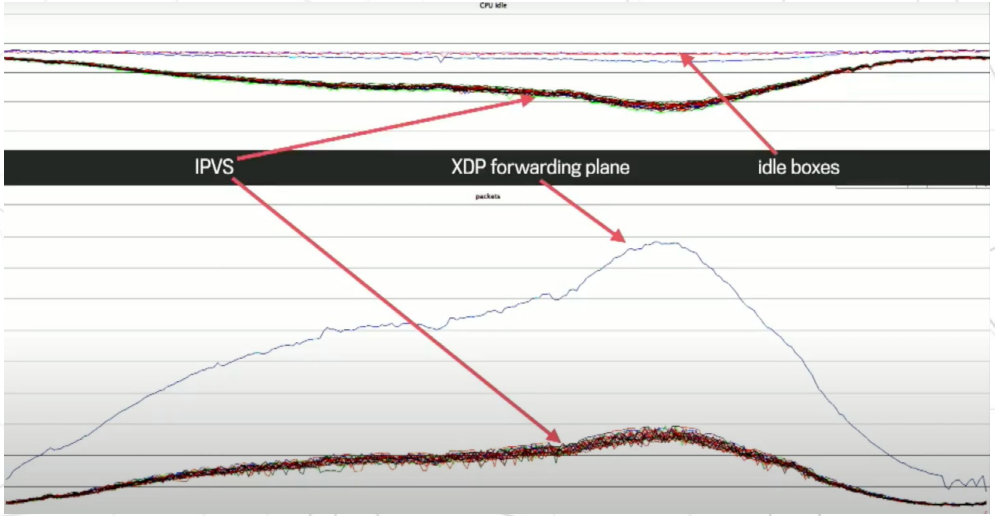
- CPU-heavy
- Bad performance for large number of new connections
- IPVS is a part of the Linux kernel. New features require new kernel.

Layer 4 Load Balancer

BPF XDP Implementation

- XDP runs before expensive memory allocation needed by the network stack.
- Super CPU effective: 3x packets with 7x less CPU
- BPF program release process is independent from kernel

Katran project was open-sourced
https://github.com/facebookincubator/katran



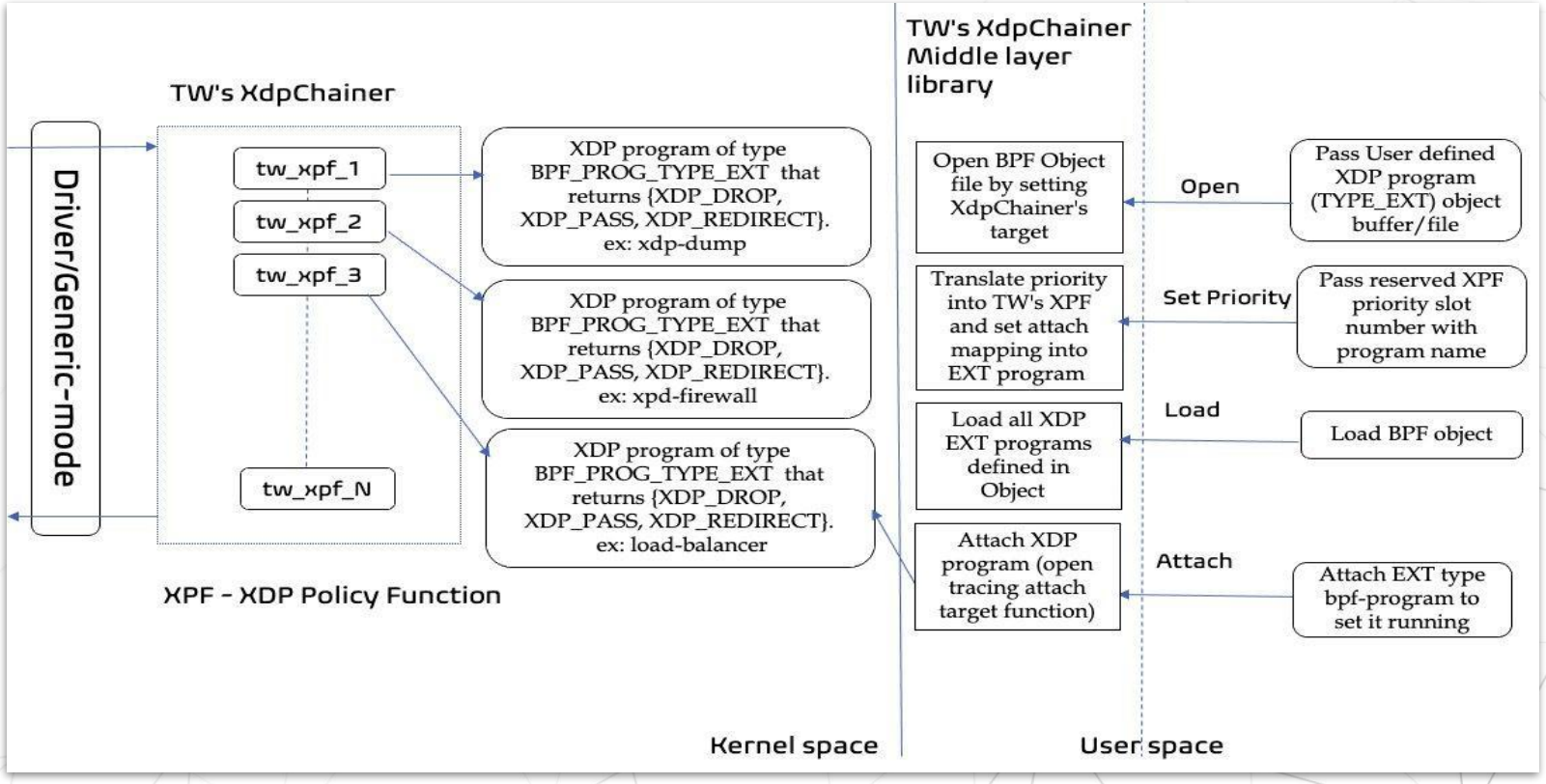
Challenges of using XDP in production

- Only one XDP program is allowed per network device, but many programs needs to be attached. E.g. firewall, load balancing, traffic capture.
- For multiple programs, execution order should be predictable.
- Each program should be able to return XDP_DROP to prevent further execution.

Multiple XDP programs support

- “XDP Chainer” was built internally to support this functionality
- It defines 20 slots for XDP programs, team must reserve specific slot in advance.
- XDP Programs attach to hooks inside XDP Chainer with type BPF_PROG_TYPE_EXT.
- Programs always execute in the same order, based on allocated slot number.

XDP Chainer


TW's XdpChainer
Driver/Generic-mode
tw_xpf_1
tw_xpf_2
tw_xpf_3
tw_xpf_N
XDP program of type BPF_PROG_TYPE_EXT that returns {XDP_DROP, XDP_PASS, XDP_REDIRECT}. ex: xdp-dump
XDP program of type BPF_PROG_TYPE_EXT that returns {XDP_DROP, XDP_PASS, XDP_REDIRECT}. ex: xpd-firewall
XDP program of type BPF_PROG_TYPE_EXT that returns {XDP_DROP, XDP_PASS, XDP_REDIRECT}. ex: load-balancer
XPF - XDP Policy Function
TW's XdpChainer Middle layer library
Open BPF Object file by setting XdpChainer's target
Translate priority into TW's XPF and set attach mapping into EXT program
Load all XDP EXT programs defined in Object
Attach XDP program (open tracing attach target function)
Open
Pass User defined XDP program (TYPE_EXT) object buffer/file
Set Priority
Pass reserved XPF priority slot number with program name
Load
Load BPF object
Attach
Attach EXT type bpf-program to set it running
Kernel space
User space

Decapsulation

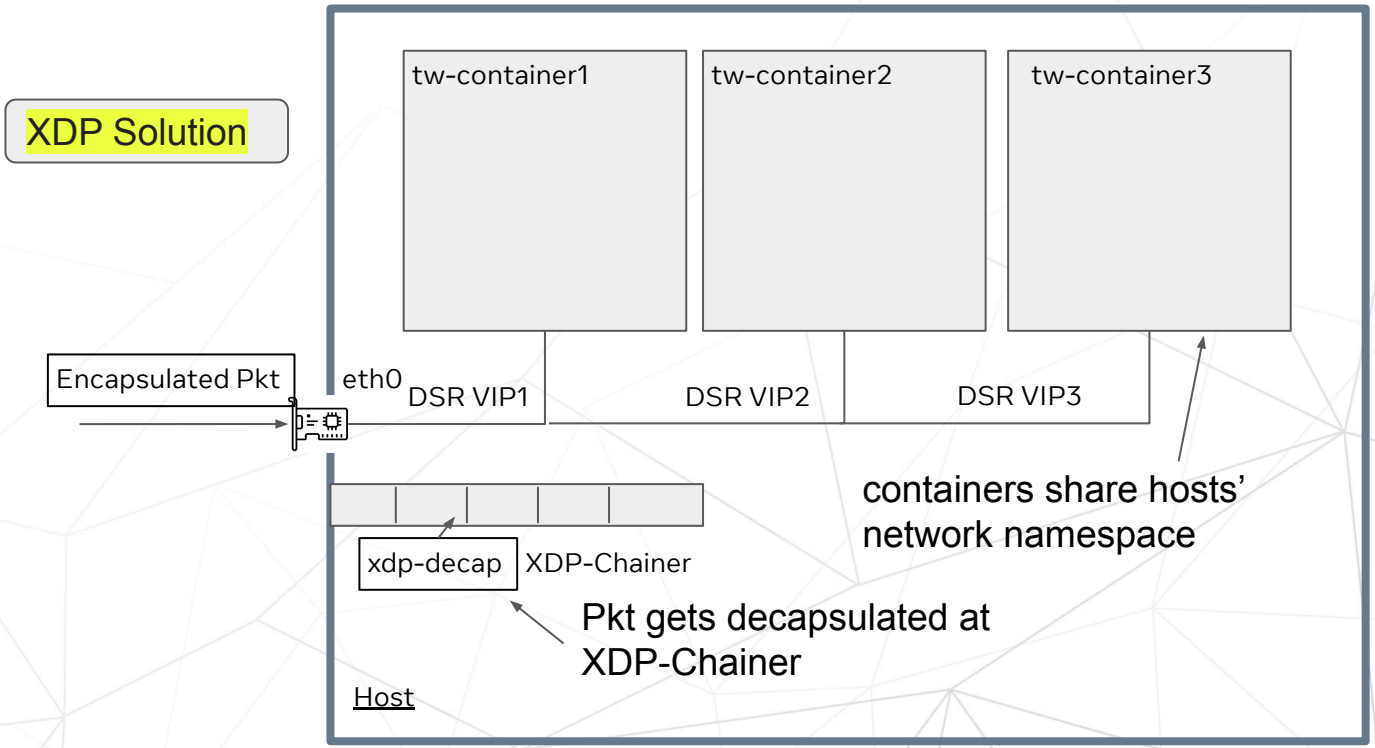
Original implementation was based on the kernel modules

- IPv4 and IPv6 tunneling interfaces for containers
- ip6_tunnel in “external” mode for the bare metal, supporting IPv4 and IPv6

XDP decapsulation

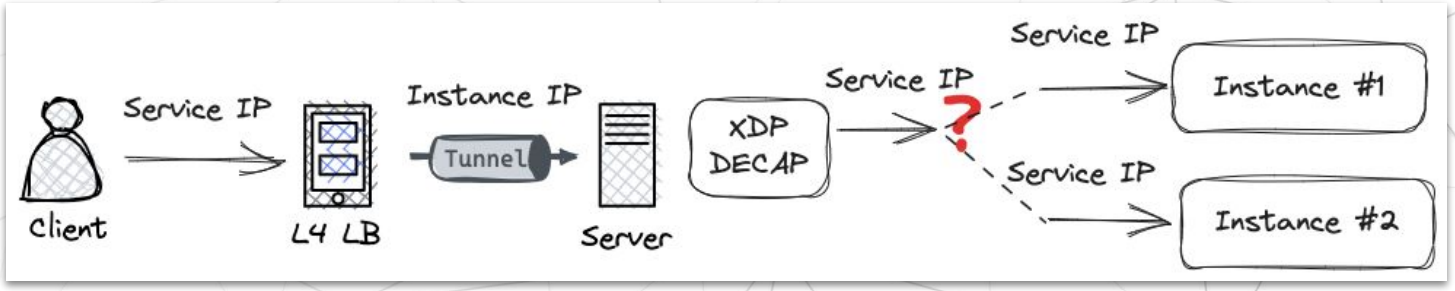
- Minor performance gains, 5% softirq CPU
- Simplified setup
- Provided decapsulation statistics

XDP Decapsulation Solution



Issues with XDP approach

- Security. Container that is using XDP program need hosts' root access to attach XDP program
- Release cycle. Main XDP program can only be updated after all children are detached
- If DSR decapsulation done on XDP level and we have 2 containers using the same DSR VIP, we will have issues on the host with routing packets to the right container



Service cross-impact in heavily stacked environment

- Traffic Black-holes
  - A local VIP or routing table entry created on the host and discards traffic
- Accidental exposure to the internet
  - One service setup a VIP that is open to the public
  - Another service binds to the [::]:<port_from_the_open_range>
- Performance impact
  - A service opens multiple UDP sockets
  - A collocated service suffer, udp_lib_lport_inuse

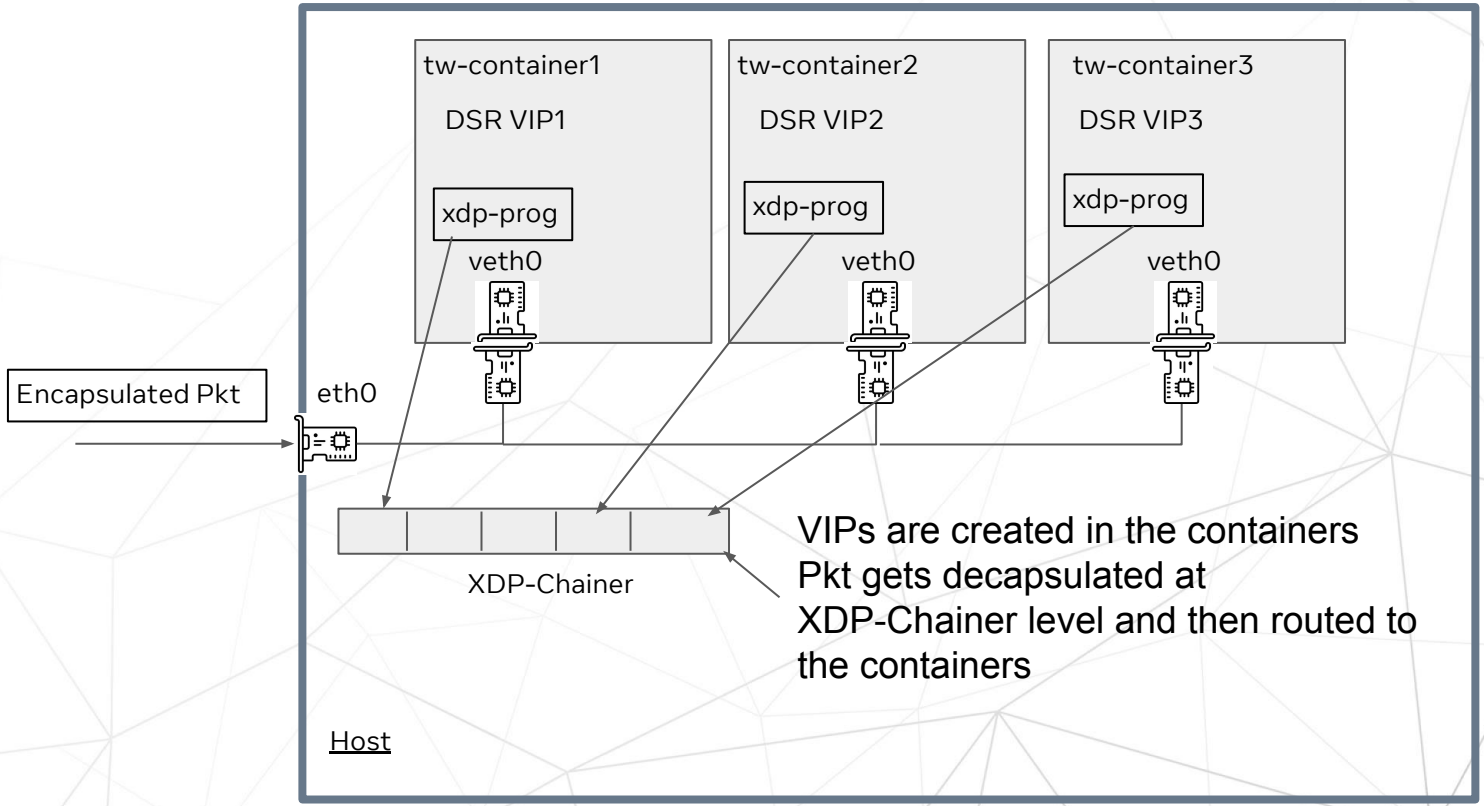
Network Namespaces

Isolation of the system resources associated with networking

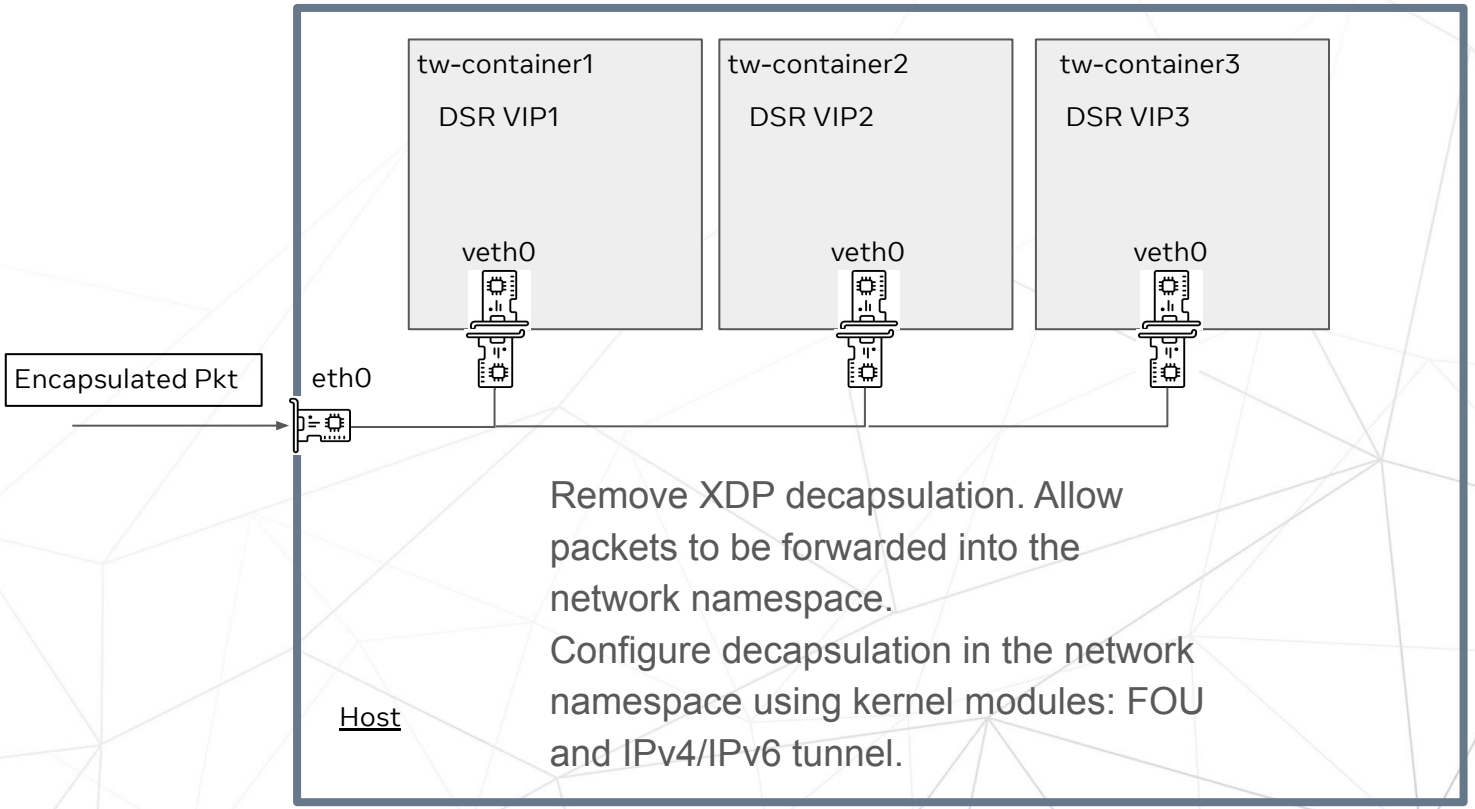
- IPv4 and IPv6 protocol stacks.
- Some of the network sysctls.
- Routing tables.
- Its own set of interfaces, including loopback interface with both IPv4 127.0.0.1 and IPv6 [::1] addresses assigned.
- Allows multiple service to listen to same port
- Enables to use container firewall so is useful even in the case service uses entire host

Resolve most of the cross-impact problems.

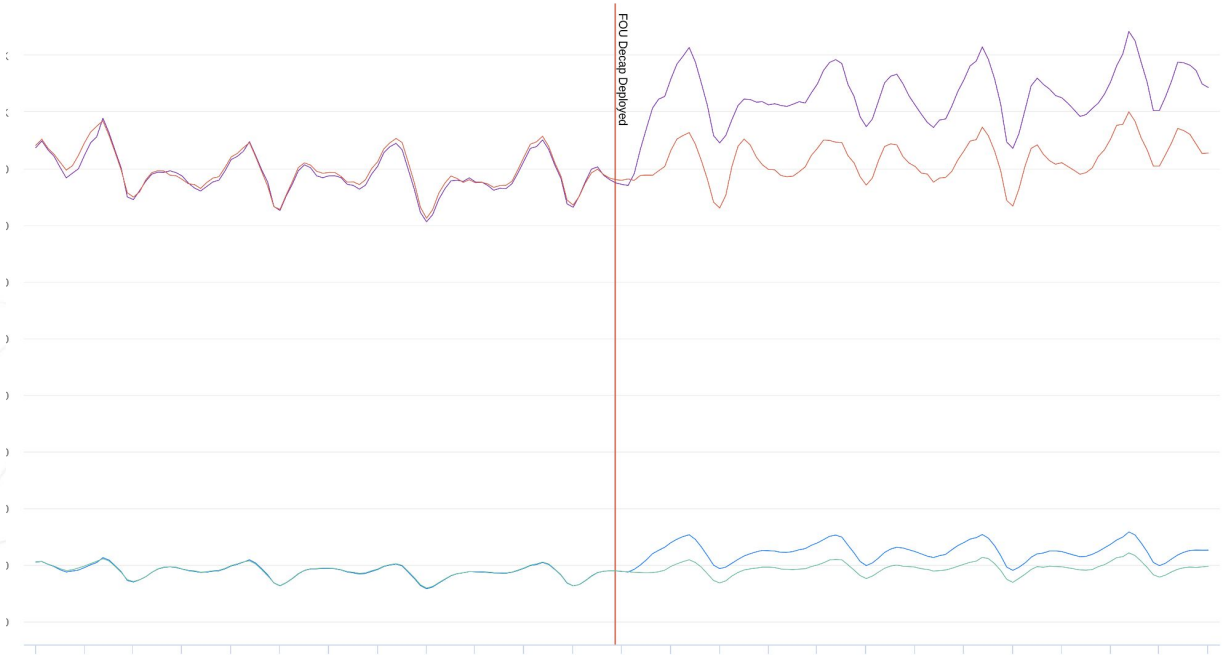
XDP Decapsulation and Network Namespaces



Back to the Kernel modules

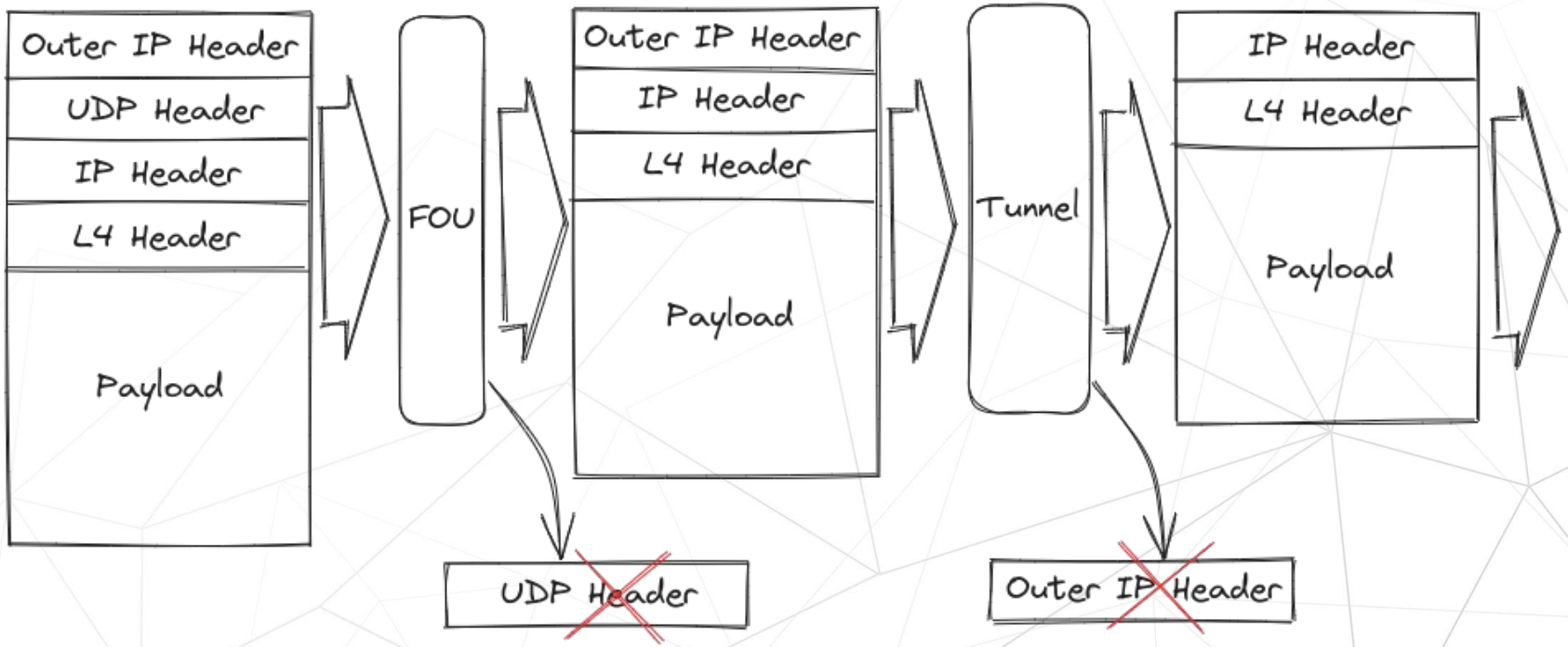


Back to the kernel modules



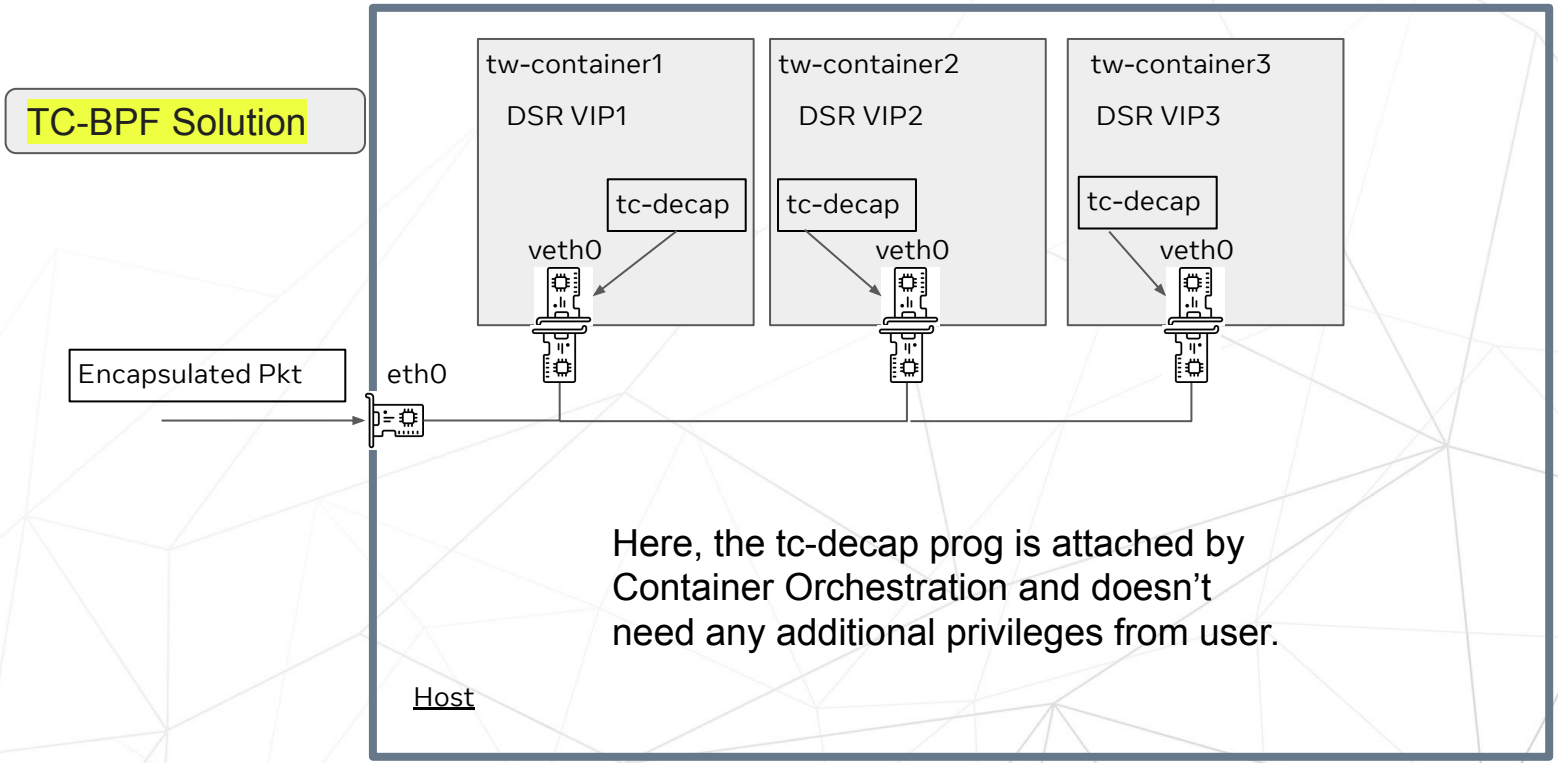
The pilot deployment resulted in 8% latency increase under load.

FOU + Tunneling Driver Decapsulation

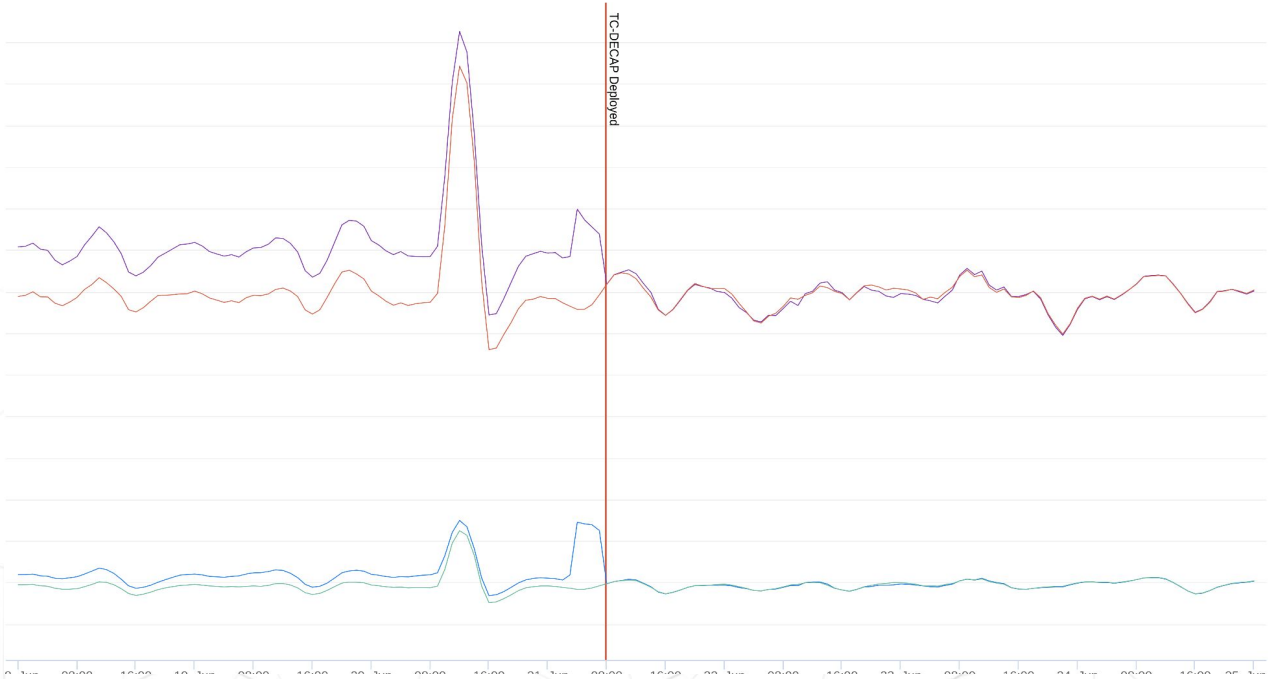


Multiple iterations of the ingress packets through the stack.

TC-BPF Decapsulation



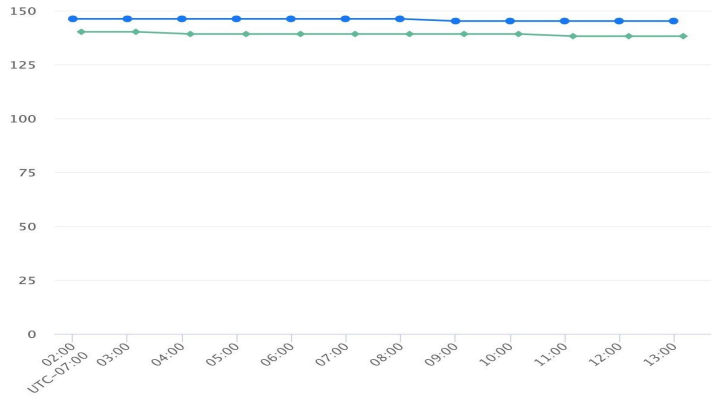
TC-BPF Decapsulation - Latency



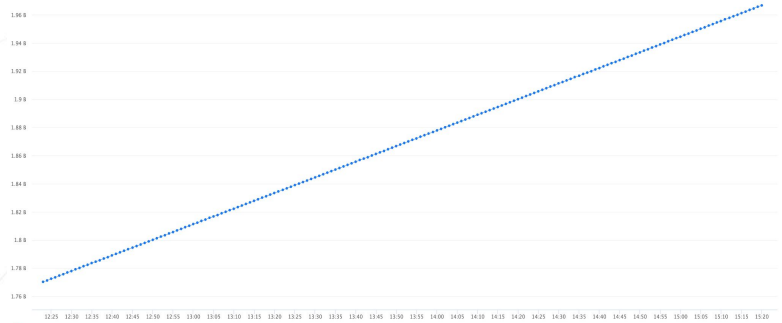
Latency issue is resolved.

Tc-decap vs Xdp-decap

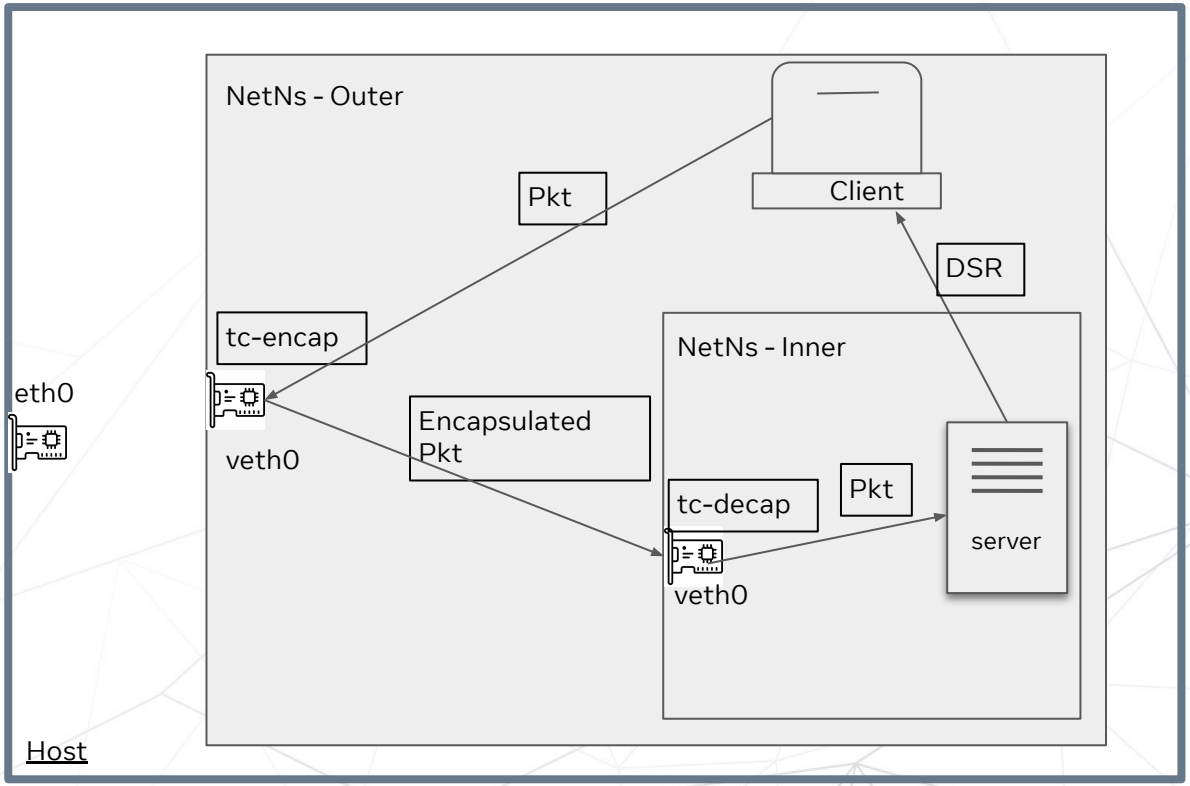
- It’s runtime is very close to xdp-decap



- It expose the same decap counter that we had in xdp-decap



TC-BPF Decapsulation - Easy Isolated Testing



Hardware Matters

Different vendors expose different behavior

- Mellanox provides CHECKSUM_COMPLETE
- Broadcom might not pull all headers
- Hard to mimic in testing
- Some bugs would manifest on specific hardware only

Future work

- Replace veth-device with meta-device to eliminate softirq increase when service runs in netns.

Q&A