XDP - challenges and future work

Jesper Dangaard Brouer (Red Hat)
Toke Høiland-Jørgensen (Karlstad University)

LPC Networking Track
Vancouver, Nov 2018



Outline

Introduce scientific XDP paper

Lots of XDP activity – Especially at this conference

Evolving XDP – without killing performance


XDP-paper

Scientific XDP paper accepted:

- "The eXpress Data Path: Fast Programmable Packet Processing in the Operating System Kernel" - Available on GitHub
- Conference: ACM CoNEXT 2018, Dec 4-7, Heraklion, Greece

Purpose

- Describe the full XDP system design in one place
- Make scientific research community notice XDP
- First head-to-head comparison with DPDK

This talk came out of the "Limitations and Future Work" section

- Purpose: soliciting feedback and ideas from the community

What is XDP?

XDP basically: New layer in the kernel network stack

- Before allocating the SKB
- Driver level hook at DMA level

Means: Competing at the same “layer” as DPDK / netmap

- Super fast, due to
  - Take action/decision earlier (e.g. skip some network layers)
  - No memory allocations

Not kernel bypass; data-plane is kept inside the kernel

- Via eBPF: makes early network stack run-time programmable
- Cooperates with the kernel stack


XDP is a huge success

XDP is maturing in upstream kernels

- Still under active development

Popular topic at this LPC network track miniconf

- 12 talks XDP and eBPF-related! (incl. this)
- 4 related to non-XDP eBPF issues
- 8 directly XDP-related (incl. this)
  - We will list these talks and frame them in the bigger XDP picture


Production use-cases

XDP has seen production use

- CloudFlare publically say they use XDP
- Suricata have XDP plugins
- Facebook released the Katran load balancer

Other talks will cover this:

- (Facebook) XDP 1.5 years in production. Evolution and lessons learned
- (Facebook) eBPF / XDP based firewall and packet filtering


Performance Graps: XDP vs. DPDK

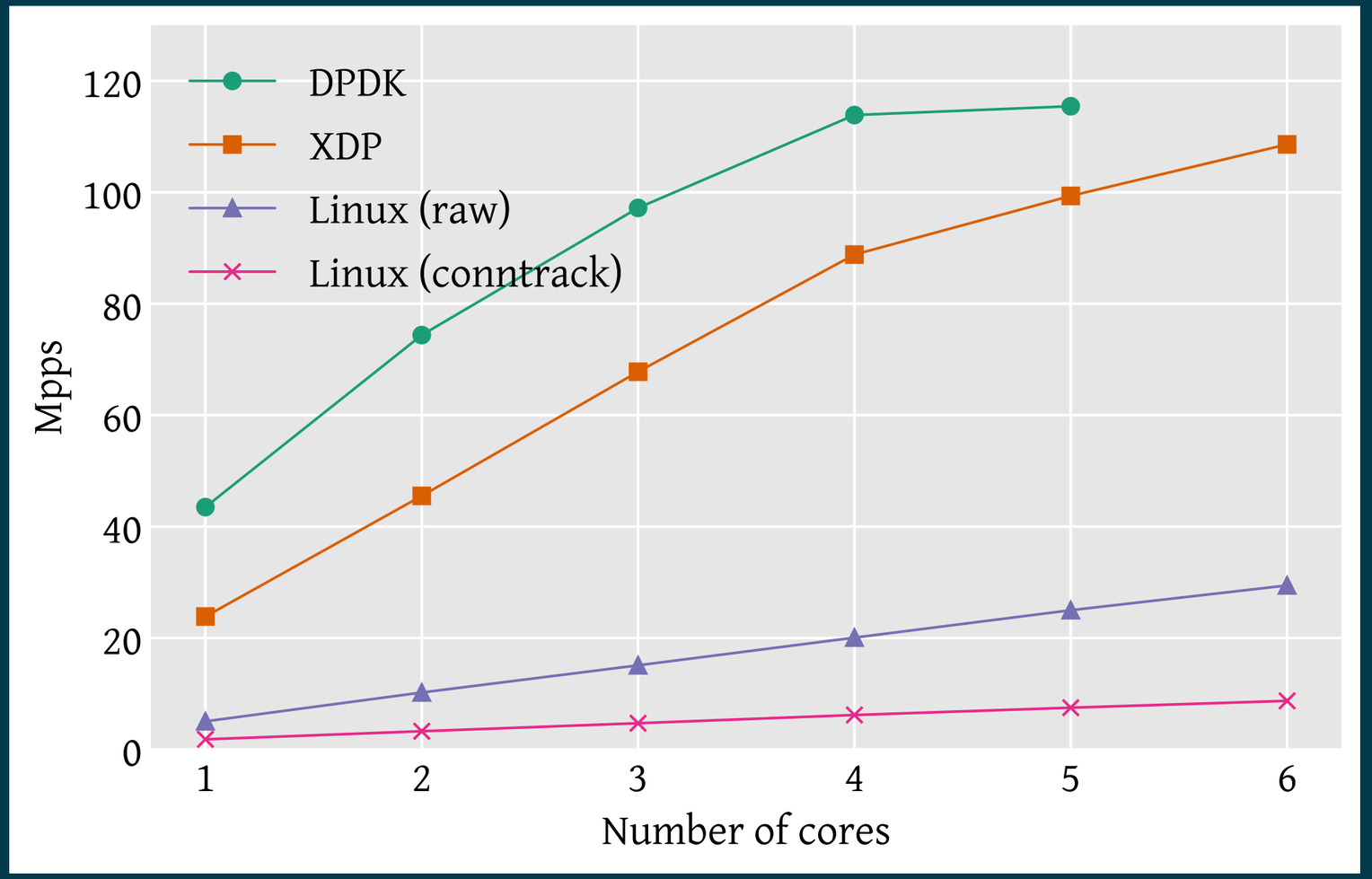
Don’t have time look at performance details

- Go read our XDP paper

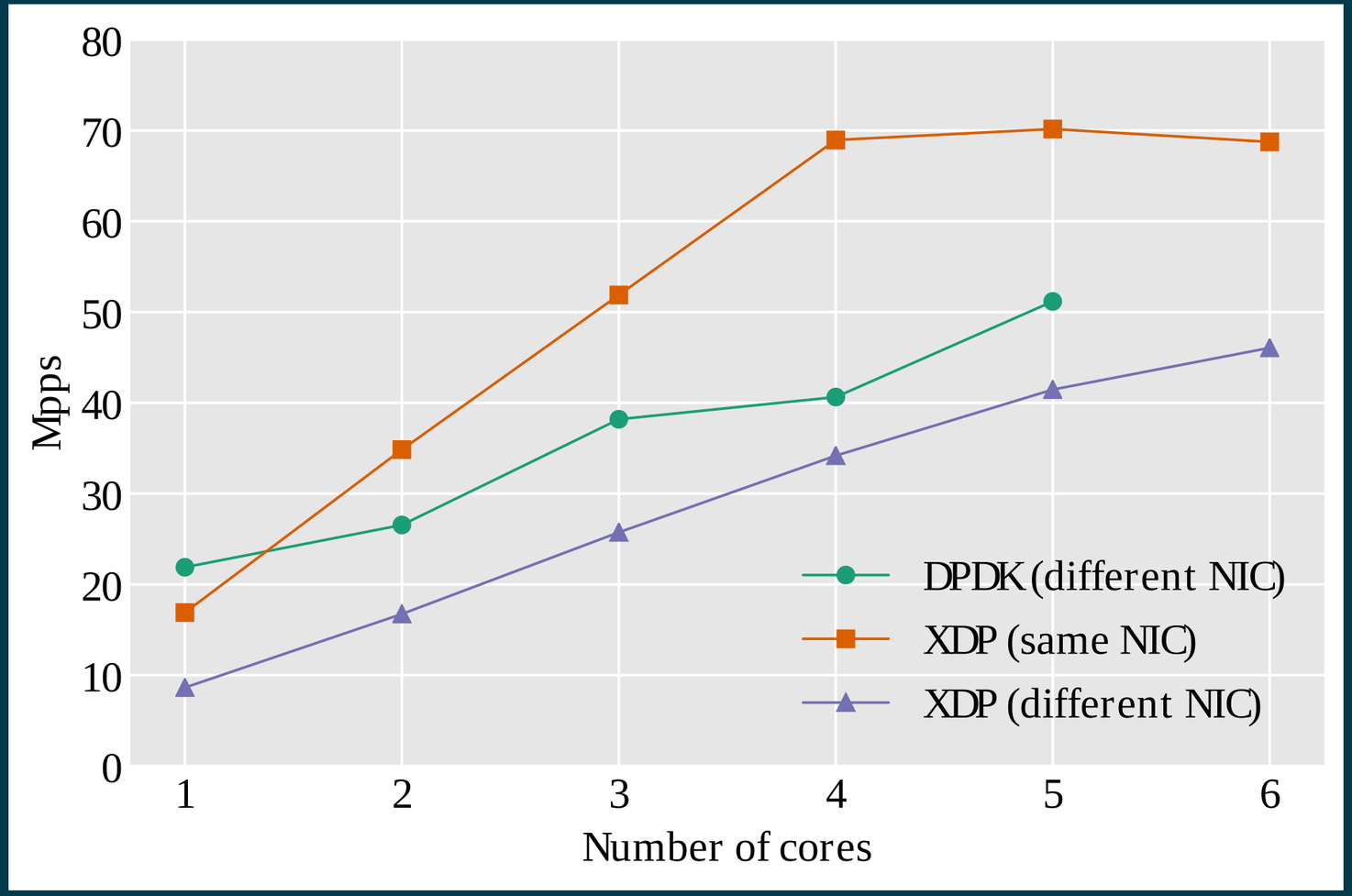
Following graphs show:

- XDP has narrowed the gap to DPDK, but not quite caught up
- For packet forwarding, XDP_TX is faster than DPDK


Packet drop performance (from paper)



Packet redirect performance (from paper)



Evolving XDP

While XDP is a disruptive and innovative technology

- Still needs to follow kernel's evolutionary development style
- Far from finished, improvements and features in every kernel release
- Beneficial to cooperate with the kernel community
  - Yes, you actually need to describe and argue for your use-case!


New XDP features vs high performance

XDP is all about processing many (M)PPS

- Watch out so feature creep doesn't kill performance!

Guiding principle: New features must not negatively affect baseline XDP perf

- Optimisation hint: move runtime checks to setup time

Issue: Who is monitoring XDP performance?

- We need CI tests for this… but who will do this work?


Evolving XDP via BPF-helpers

Think of XDP as a software offload layer for the kernel network stack

- Setup and use Linux kernel network stack
- But accelerate parts of it with XDP

IP routing good example:

- Let Linux handle routing (daemons) and neighbour lookups
- Access routing table from XDP via BPF helpers
- This is covered in David Ahern's talk: Leveraging Kernel Tables with XDP

We should encourage adding helpers instead of duplicating data in BPF maps


XDP as a building block

XDP is a core kernel layer building block

- Open Source projects should build and innovate on top
- Only time will tell what people use it for


Building block for Guest OS delivery?

Directions we hope to see XDP go

- Faster packet delivery into Guest OS
  - Already possible (with copy) via tuntap driver into virtio_net
  - The AF_XDP approach might offer zero-copy


Building block for P4

P4 vs eBPF/XDP is the wrong attitude

- Instead compile your P4 code into eBPF and run it with XDP
- Talk on this approach by William Tu (VMware):
  - P4C-XDP: Programming the Linux Kernel Forwarding Plane using P4


AF_XDP: Zero-copy to userspace

Regular XDP performance benefit comes from in-kernel processing

AF_XDP is for faster raw packet delivery to userspace

- Unlike tcpdump, as it owns/steals the packet
  - Might want to add a copy mode for XDP_PASS
- Hooking into XDP provides:
  - Packet filter flexibility (avoid reinjecting into network stack)
  - Fallback mode (with copying) on any driver supporting XDP_REDIRECT
- Performance tricks:
  - Userspace preallocates memory, passes to kernel for NIC RX-rings
  - NetChannel-like SPSC queues between kernel and userspace


AF_XDP talks

Two talks about AF_XDP at LPC:

- The Path to DPDK Speeds for AF_XDP
  - By Björn Töpel and Magnus Karlsson (Intel)
- Bringing the Power of eBPF to Open vSwitch
  - By William Tu (VMware)


Future directions for XDP

Warning: Crazy ideas ahead!


Moving SKB allocation out of drivers

Long term goal: Remove need for SKB allocations in NIC drivers

Actually supported today! – can avoid allocating SKB in driver

- Via XDP_REDIRECT into CPU-map or tun-driver

Missing part: driver offloads (e.g., csum-info, rxhash, HW-mark)

- Needs metadata in a vendor neutral format
- Way forward: Teach the core stack about BTF struct metadata?

Hope this will be covered in the talk by P.J. Waskiewicz and Neerav Parikh (Intel):

- XDP acceleration using NIC metadata, continued


Resource allocation for ndo_xdp_xmit()

When redirecting to another device, XDP calls target driver's `ndo_xdp_xmit()`

- But XDP TX resources are not allocated by default
  - Because these are sparse HW resources: 1 TX-queue per CPU core

Current hack: allocate resources on XDP program load

- Even when device doesn't need to receive redirect traffic
- Requires dummy XDP program on the egress device to use redirect
  - Removing it can crash the kernel!

We need an explicit API for enabling XDP TX on a device

- Would be natural to trigger when device is added to DEVMAP
  - But how to handle non-map redirect variant?

What does an XDP driver support?

Userspace cannot query which XDP features a driver supports

- Original goal: support all features in all drivers. Is this still realistic?
  - Only 3 HW-drivers have implemented XDP_REDIRECT.
  - Some users are happy with (only) XDP_DROP and XDP_TX.

Userspace needs this information. For example, in Suricata:

- If XDP_REDIRECT is not supported, either:
  - Reject config at startup
  - Or use alternative TC-based solution

XDP egress hook

Issue: Programs can't predict if XDP_REDIRECT will succeed

- If destination TX ring is full, packets are silently dropped
  - Only way to detect this is with tracepoints
- Especially problematic for fast ➔ slow device redirect

Idea: Add a new egress hook to XDP

- Execute just before packets are put into TX ring
- Can access TX ring status
  - When full, selectively drop or signal ingress programs to back off
- Also useful for implementing QoS, policing, AQM
- Crazy idea: Allow new XDP action, including redirect out other device

Memory and DMA


Memory models

Recent (4.18): XDP memory models per driver RX queue

- New flexibility to innovate
- Also opportunity to share common code between drivers
  - page_pool is an example, need more drivers using it

Planned changes:

- Extend xdp_return_frame API with bulking, to amortise overhead
- Keep pages DMA mapped in page_pool (almost supported)


DMA mapping

More optimizations for DMA mapping needed

- Was low priority, due to almost zero cost on Intel CPUs
- But Spectre-V2 mitigation makes DMA calls more expensive

To summarise...


XDP paper

- "The eXpress Data Path: Fast Programmable Packet Processing in the Operating System Kernel"
- Will be presented at ACM CoNEXT 2018, Dec 4-7, Heraklion, Greece
- Available on GitHub - feel free to share!


XDP at LPC

- XDP in production:
  - XDP 1.5 years in production. Evolution and lessons learned
  - eBPF / XDP based firewall and packet filtering
- Kernel helpers:
  - Leveraging Kernel Tables with XDP
- Metadata:
  - XDP acceleration using NIC metadata, continued
- XDP as a building block:
  - P4C-XDP: Programming the Linux Kernel Forwarding Plane using P4
- AF_XDP:
  - The Path to DPDK Speeds for AF_XDP
  - Bringing the Power of eBPF to Open vSwitch


Future directions for XDP

- Moving SKB allocation out of drivers
- Resource allocation for ndo_xdp_xmit()
- Discovering supported XDP features for a device
- Adding an egress hook to XDP
- NIC memory models and DMA mapping


End

Thanks to all contributors

- XDP + eBPF combined effort of many people