The Path to DPDK Speeds for AF_XDP

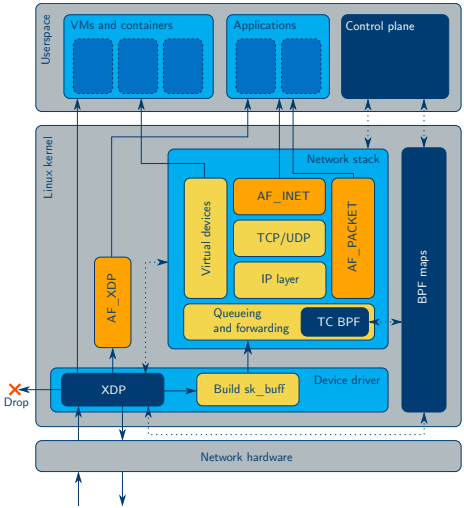
Magnus Karlsson, magnus.karlsson@intel.com
Björn Töpel, bjorn.topel@intel.com

Linux Plumbers Conference, Vancouver, 2018

Legal Disclaimer

- Intel technologies may require enabled hardware, specific software, or services activation. Check with your system manufacturer or retailer.
- No computer system can be absolutely secure.
- Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.
- Cost reduction scenarios described are intended as examples of how a given Intel- based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.
- All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.
- No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.
- Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.
- Intel, the Intel logo, and other Intel product and solution names in this presentation are trademarks of Intel.
- Other names and brands may be claimed as the property of others.
-

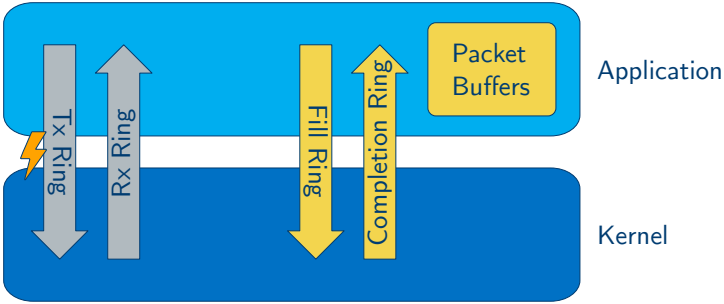
XDP 101



AF_XDP 101

- Ingress
  - Userspace XDP packet sink
  - `XDP_REDIRECT` to socket via `XSKMAP`
- Egress
  - No XDP program
- Register userspace packet buffer memory to kernel (UMEM)
- Pass packet buffer ownership via descriptor rings

AF_XDP 101



- Fill ring (to kernel) / Rx ring (from kernel)
- Tx ring (to kernel) / Completion ring (from kernel)
- Copy mode (DMA to/from kernel allocated frames, copy data to user)
- Zero-copy mode (DMA to/from user allocated frames)

Baseline and optimization strategy

- Baseline
  - Linux 4.20
  - 64B @ ~15-22 Mpps
- Strategy
  - Do less (instructions)
  - Talk less (coherency traffic)
  - Do more at the same time (batching, i$)
  - Land of Spectres: fewer retpolines, fewer retpolines, fewer retpolines

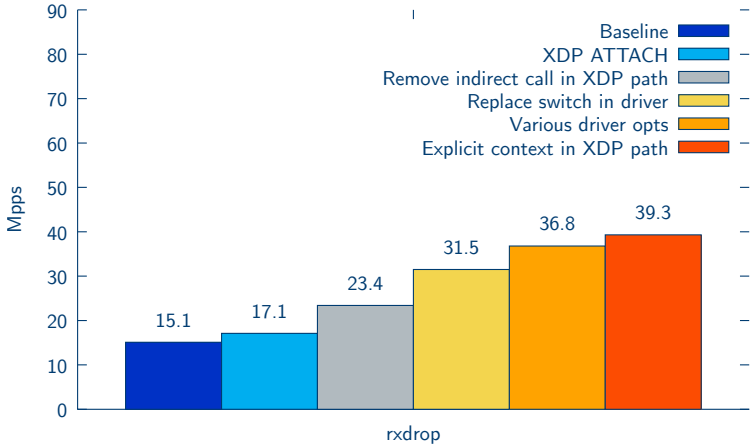
Experimental Setup

- Broadwell E5-2660 @ 2.7GHz
- 2 cores used for run-to-completion benchmarks
- 1 core used for busy-poll benchmarks
- 2 i40e 40GBit/s NICs, 2 AF_XDP sockets
- Ixia load generator blasting at full 40 Gbit/s per NIC

Ingress

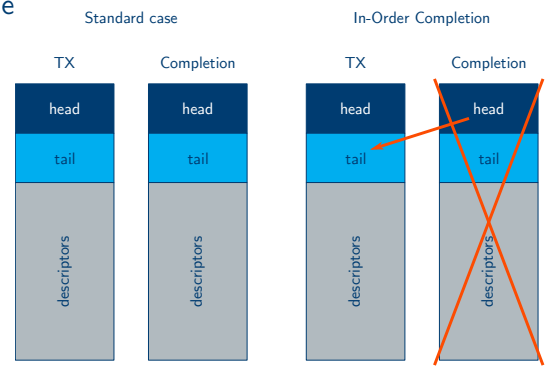
- `XDP_ATTACH` and `bpf_xsk_redirect`, attach at-most one socket per netdev queue, load built-in XDP program, 2-level hierarchy
- Remove indirect call, `bpf_prog_run_xdp`
- Remove indirect call, XDP actions switch-statement ($>= 5 \implies$ jump table)
- Driver optimizations (batching, code restructure)
- `bpf_prog_run_xdp`, `xdp_do_redirect` and `xdp_do_flush_map`: per-CPU `struct bpf_redirect_info` + `struct xdp_buff` + `struct xdp_rxq_info` vs explicit, stack-based context

Ingress, results, data not touched

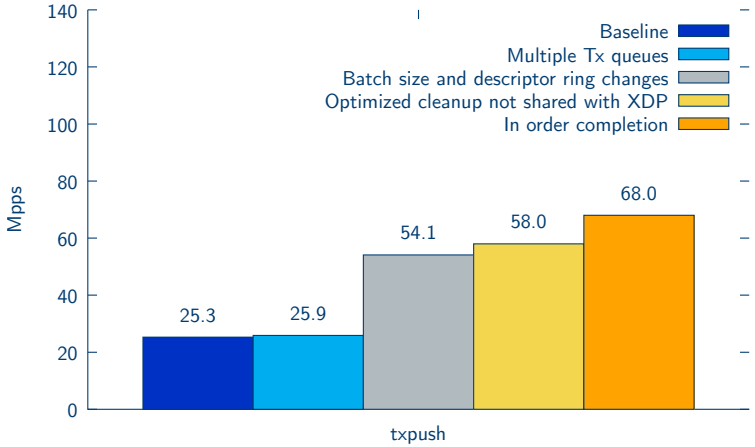


Egress

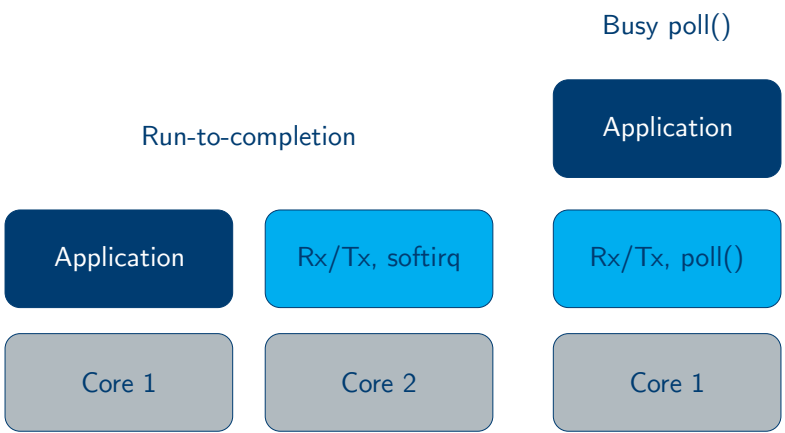
- Tx performance capped per HW queue $\Longrightarrow$ multiple Tx sockets per UMEM
- Larger/more batching, larger descriptor rings
- Dedicated AF_XDP HW Tx queues
- In-order completion, `setsockopt XDP_INORDER_COMPLETION`



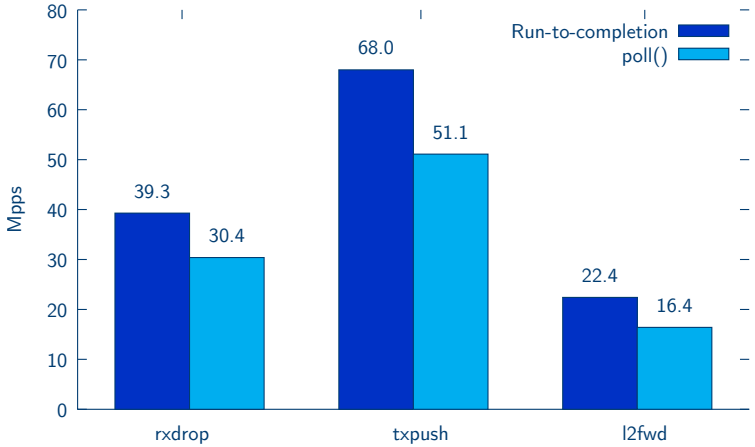
Egress, results, data not touched



Busy poll() vs run-to-completion



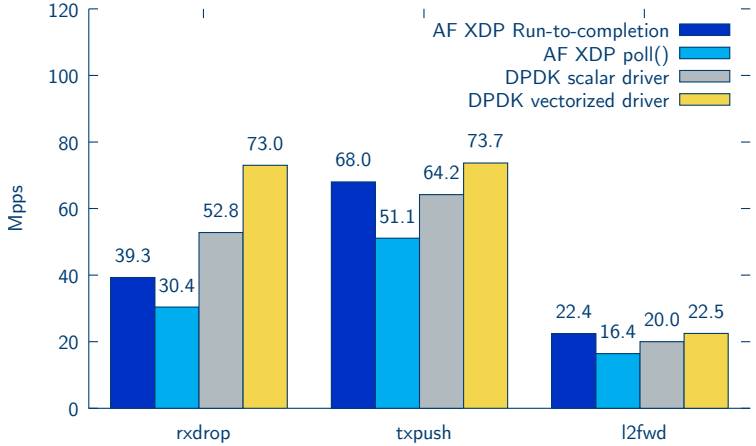
Busy poll() vs run-to-completion, results



Comparison with DPDK

- Userspace, vectorized drivers
- "Learning from the DPDK" `http://vger.kernel.org/netconf2018_files/StephenHemminger_netconf2018.pdf`

Comparison with DPDK, results



Next steps

Upstream!

- XDP: switch-statement
- Rx/Tx: drivers
- Rx: `XDP_ATTTACH` and `bpf_xsk_redirect`
- libbpf AF_XDP support
- Tx: multiple Tx sockets per UMEM
- selftest, samples

Future work

- Hugepage support, less fill ring traffic (`get_user_pages`)
- fd.io/VPP work vectors (i$, explicit batching in function calls)
- "XDP first" drivers
- Collaborate/share code with RDMA (e.g. `get_user_pages`)
- Type-writer model (currently not planned)

Summary

- Rx 15.1 to 39.3 Mpps (2.6x)
- Tx 25.3 to 68.0 Mpps (2.7x)
- Busy poll() promising
- DPDK still faster for “notouch”, but AF_XDP on par when data is touched
- Drivers need to change when skb is not the only consumer

Thanks!

- Ilias Apalodimas
- Daniel Borkmann
- Jesper Dangaard Brouer
- Willem De Bruijn
- Eric Dumazet
- Alexander Duyck
- Mykyta Iziumtsev
- Jakub Kicinski
- Song Liu
- David S. Miller
- Pavel Odintsov
- Sridhar Samudrala
- Yonghong Song
- Alexei Starovoitov
- William Tu
- Anil Vasudevan
- Jingjing Wu
- Qi Zhang


AF
XDP