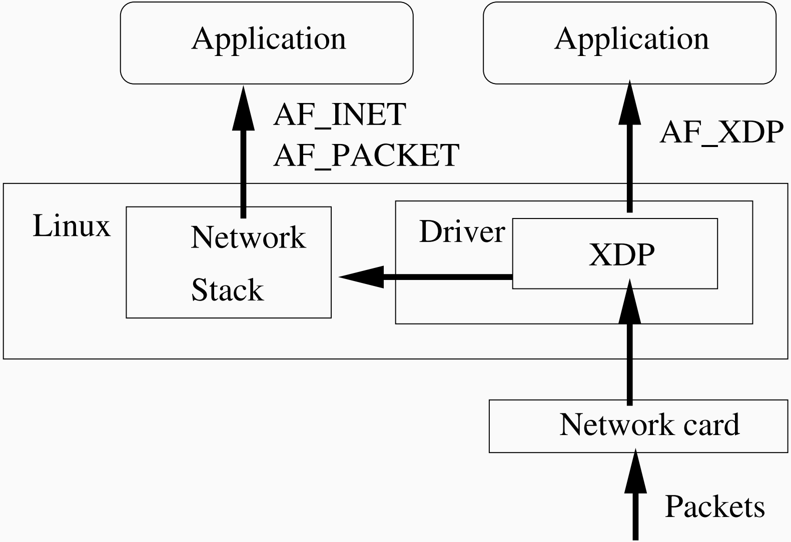
The Path to DPDK Speeds for AF_XDP

magnus.karlsson@intel.com, bjorn.topel@intel.com

October 24, 2018

LPC 2018

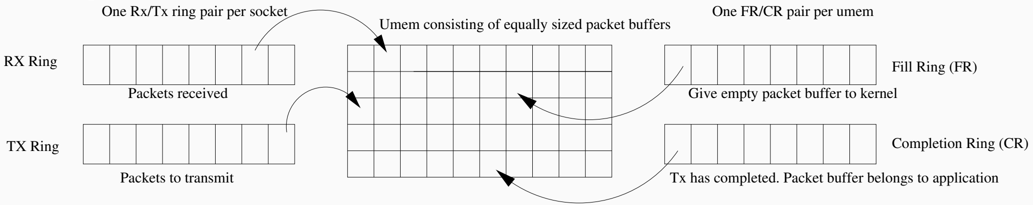
XDP 101


Application
Application
AF_INET
AF_PACKET
AF_XDP
Linux
Network
Stack
Driver
XDP
Network card
Packets

AF_XDP 101

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

AF_XDP 101


One Rx/Tx ring pair per socket
Umem consisting of equally sized packet buffers
One FR/CR pair per umem
RX Ring
Fill Ring (FR)
Packets received
Give empty packet buffer to kernel
TX Ring
Completion Ring (CR)
Packets to transmit
Tx has completed. Packet buffer belongs to application

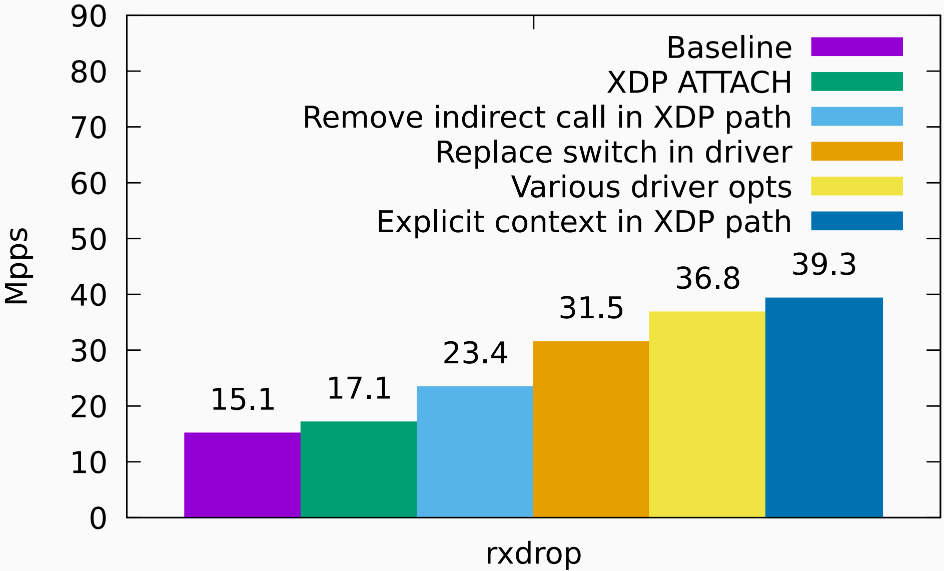
Baseline and blueprint

- Baseline: 64B @ ~15-22 Mpps
- Blueprint
  - do less (instructions)
  - talk less (coherency traffic)
  - do more at the same time (batching, i$)
  - Land of Spectres: fewer retpolines, fewer retpolines, fewer repolines

Ingress

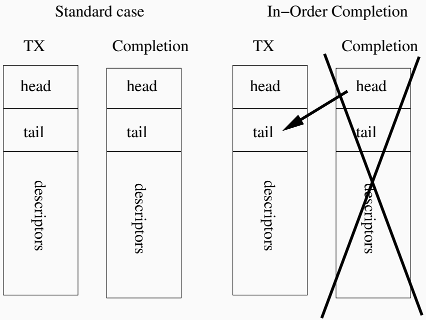
- `XDP_ATTACH` and `bpf_xsk_redirect`, attach at-most one socket per netdev queue, load built-in XDP program, 2-level hierarchy
- remove indirect call, `bpf_prog_run_xdp`
- remove indirect call, XDP actions switch-statement ($>= 5 \implies$ jump table)
- driver optimizations (batching, code restructure)
- `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

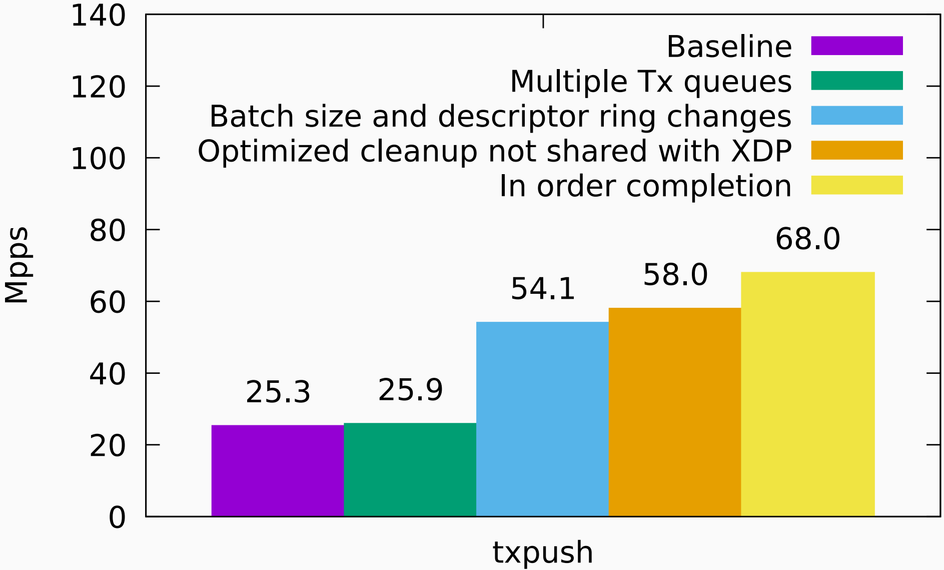


Egress

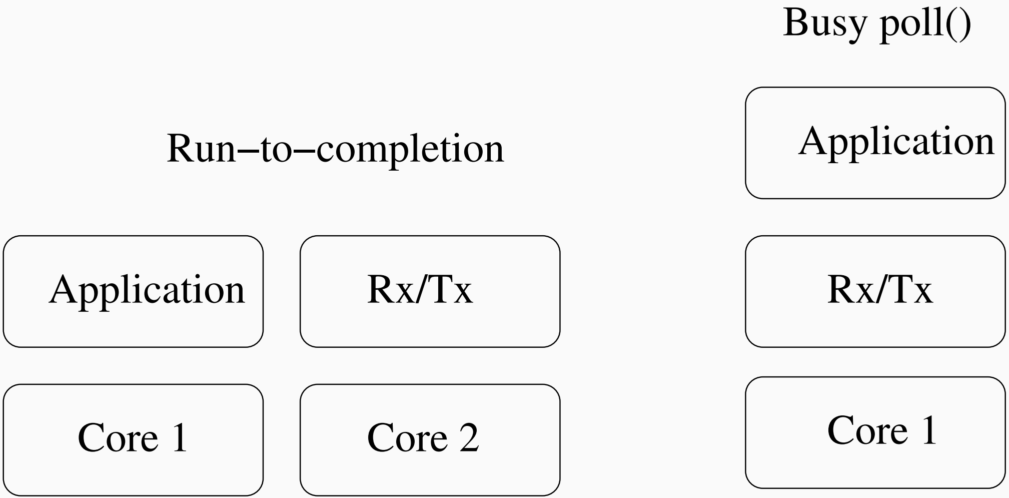
- Tx performance capped per HW queue
  $\Longrightarrow$ multiple Tx sockets per UMEM
- Larger/more batching, larger descriptor rings
- Dedicated AF_XDP Tx queues
- In-order complettion, `setsockopt` `XDP_INORDER_COMPLETION`



Egress, results, data not touched



Busy poll() vs run-to-completion



Run–to–completion

Application

Rx/Tx

Core 1

Core 2

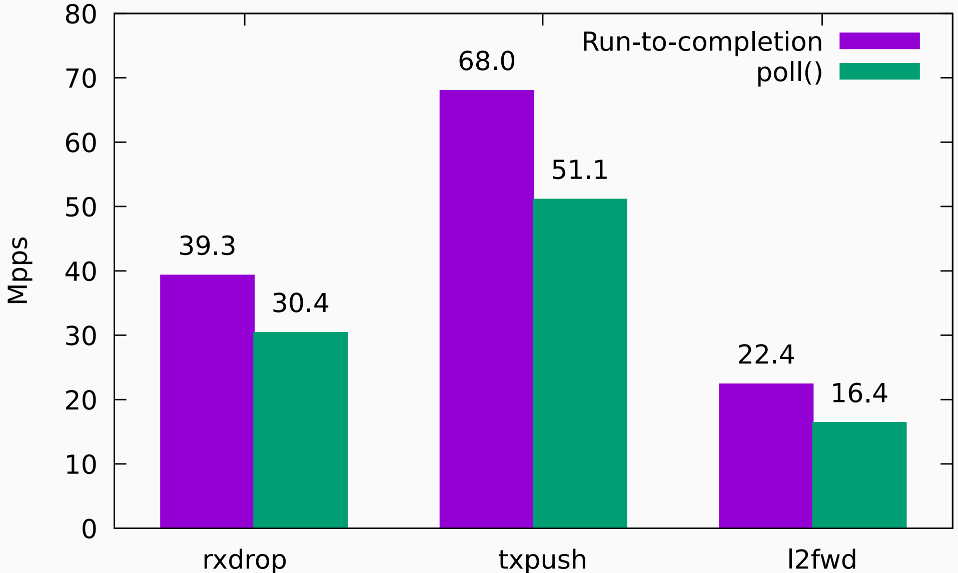
Busy poll()

Application

Rx/Tx

Core 1

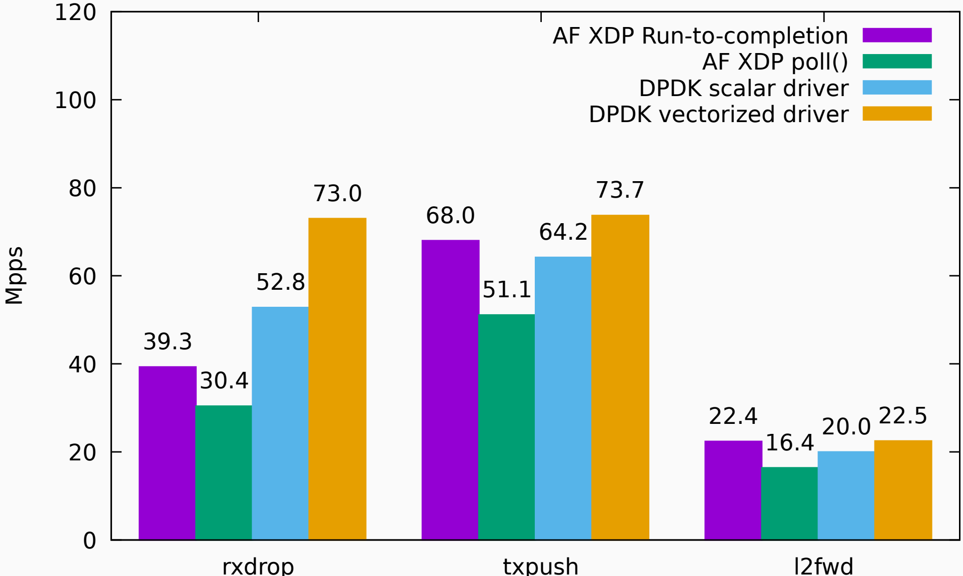
Busy poll() vs run-to-completion, results



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`
- Tx: multiple Tx sockets per UMEM
- General leftovers still to-be-upstreamed: libbpf AF_XDP support (easier to consume), selftest

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)

Thanks!

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

Questions?