eXpress Data Path and eBPF

ECE/CS598HPN

Radhika Mittal

Performance overhead in kernel stack

- Protocol processing
- Data copy
- Cache contention (between flows sharing same NUMA node)
- CPU scheduling overheads (locking, context switching)
- Interrupts
- Managing heavy datastructures (skbs)

Alternatives

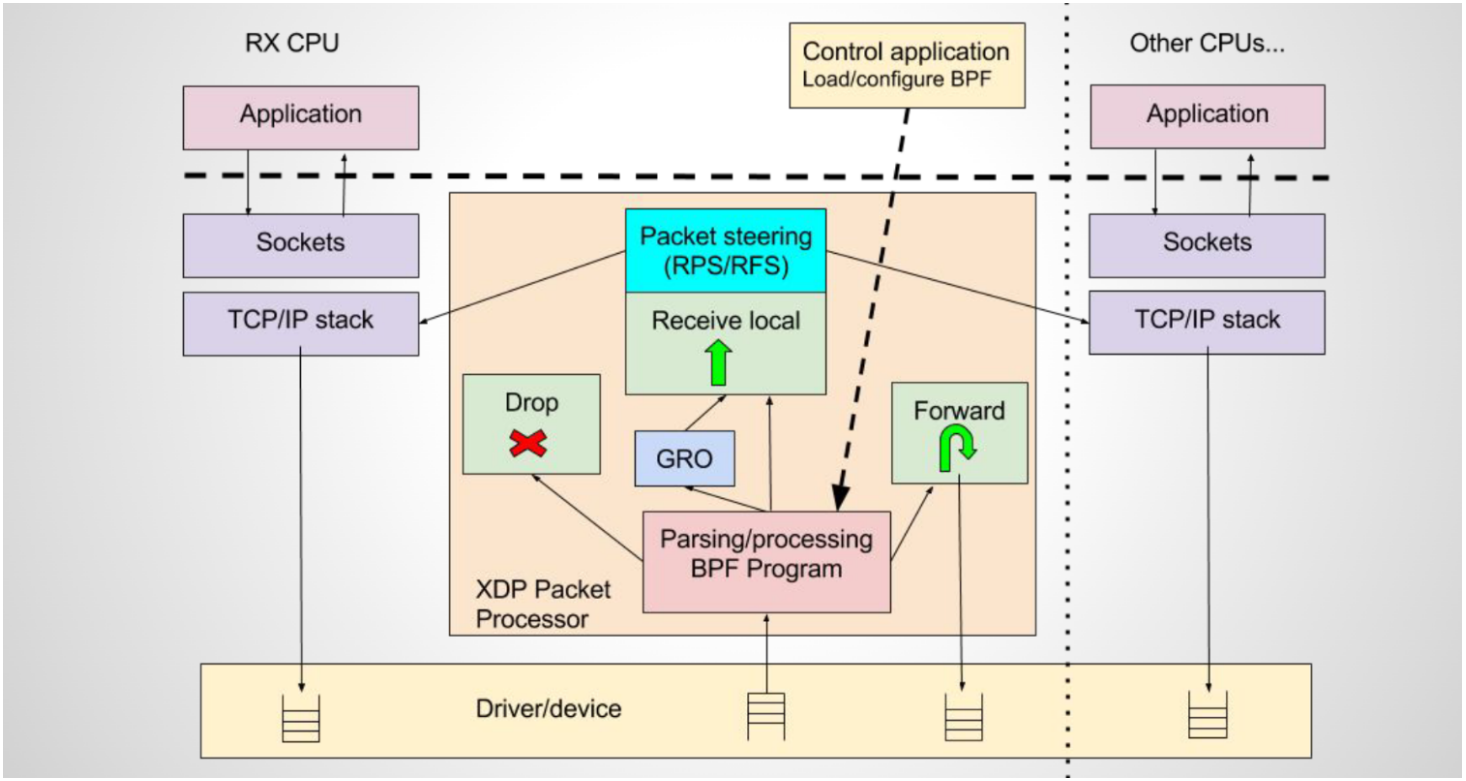
- Kernel bypass in software (user-space):
  - Enabled by high-speed packet I/O engine (DPDK)
  - Network stacks over DPDK: mTCP, IX, TAS….
  - Software NIC offload over DPDK (next week)

- Kernel/CPU bypass using specialized NIC
  - e.g. RDMA (next class)
  - other smartNIC based offloads (later in the course)

- Augmenting kernel datapath with programmable, high performance, packet processor: **XDP**

XDP (Express Datapath)

- Baremetal packet processing at the lowest point in the software stack.



Contents from Herbert and Starovoitov XDP presentation.

Benefits of XDP over DPDK?

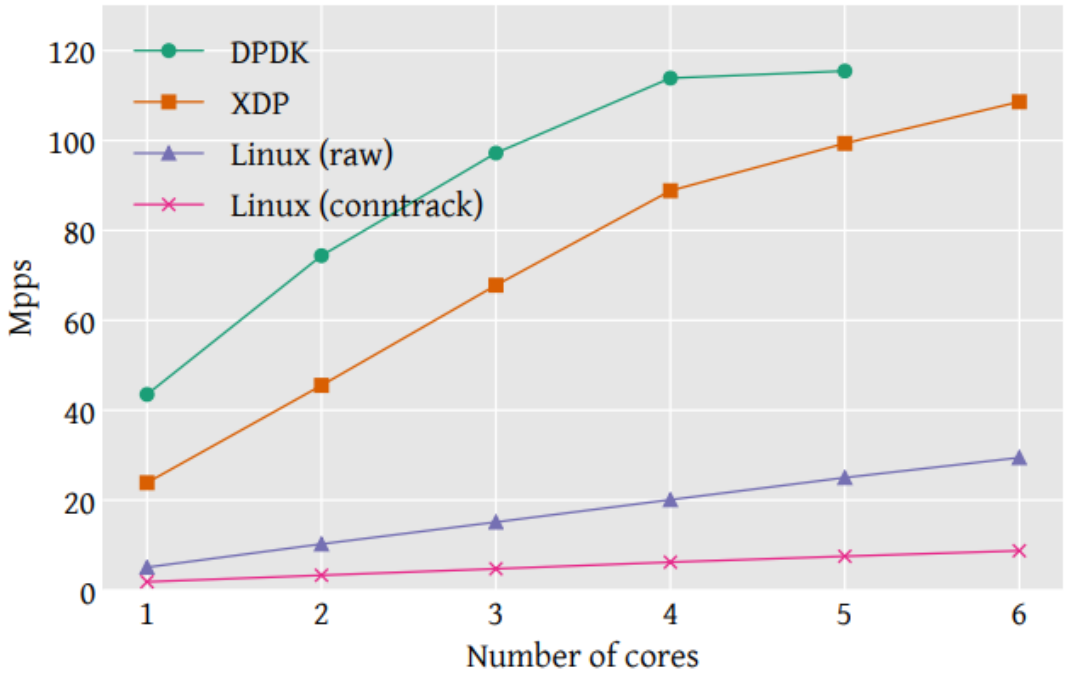
- Retains kernel security boundary.
- No special hardware requirements.
  - Only needs basic support like multiqueue NIC, TSO, etc.
  - Easier to integrate with existing NICs and drivers.
- Allows selective utilization of kernel stack (routing, TCP, etc).
- No expensive packet re-injection from user-space to kernel.
- Transparent to applications running on host.
- Dynamic runtime re-programming.
- No need to dedicate CPU cores to packet processing.

Limitations of XDP over DPDK?

- Performance!

Limitations of XDP over DPDK?

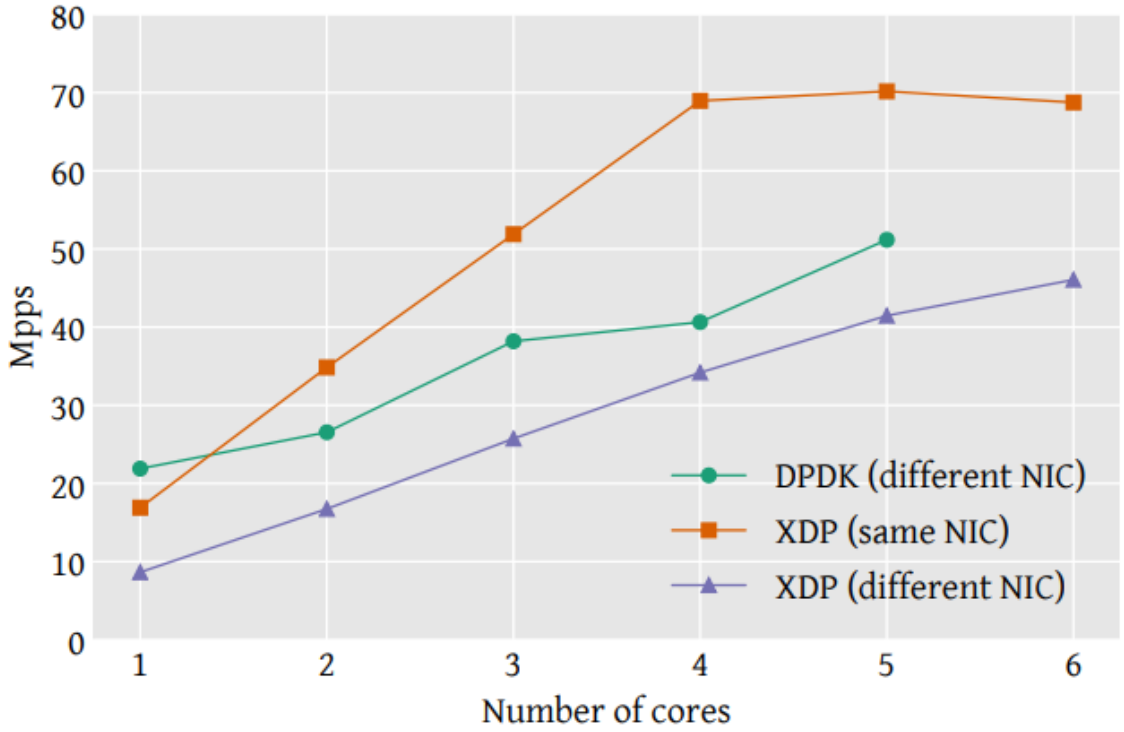
- Performance!



Packet drop throughput

Limitations of XDP over DPDK?

- Performance!



Packet forwarding throughput

Limitations of XDP over DPDK?

- Performance!

	Average		Maximum		$< 10\mu s$	
	100 pps	1 Mpps	100 pps	1 Mpps	100 pps	1 Mpps
XDP	$82\mu s$	$7\mu s$	$272\mu s$	$202\mu s$	0%	98.1%
DPDK	$2\mu s$	$3\mu s$	$161\mu s$	$189\mu s$	99.5%	99.0%

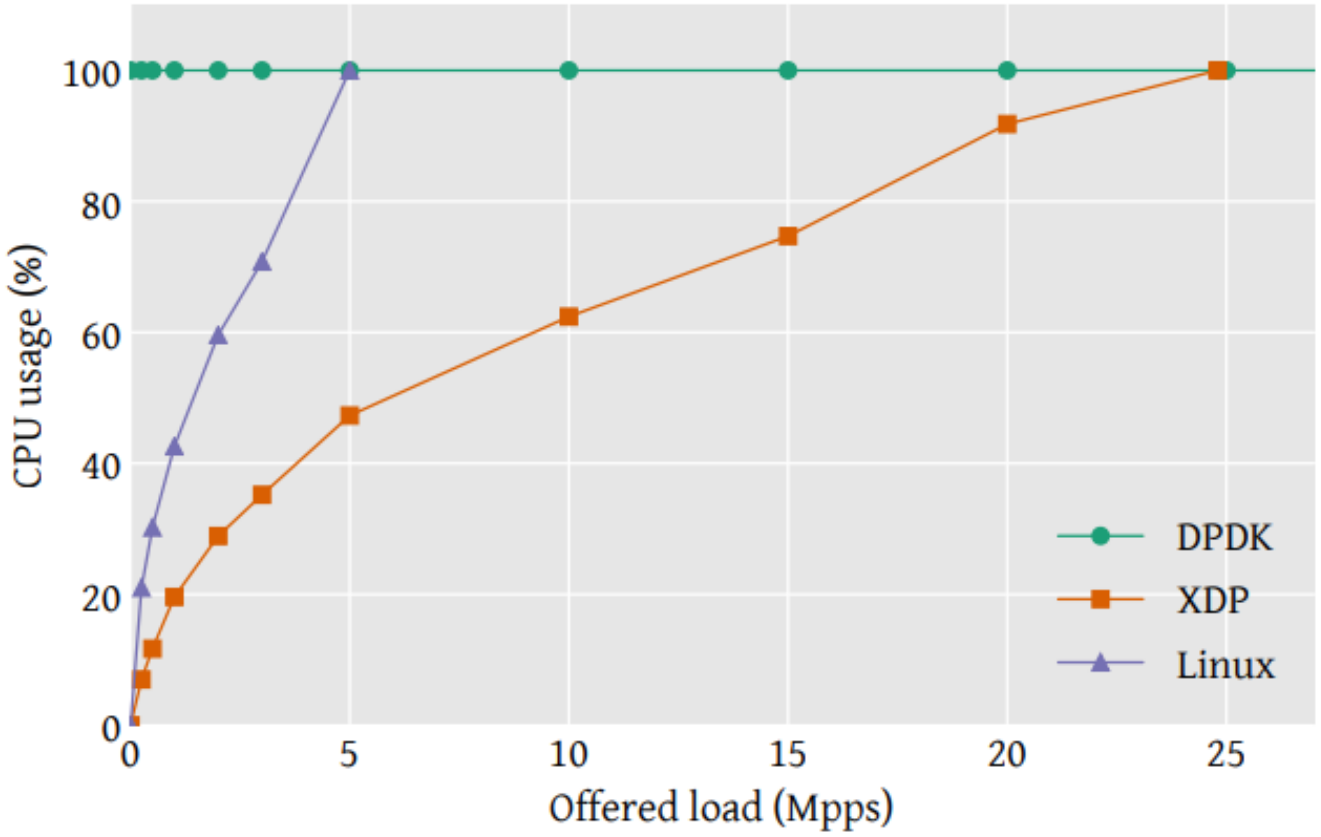
Latency

Limitations of XDP over DPDK?

- Performance!

- Reasons?
  - Interrupt vs polling
  - Overhead of generic device driver

CPU utilization



Relationship to Kernel-bypass (DPDK)

- If kernel network stack is a freeway,
  - kernel-bypass is a proposal for high-speed train infrastructure.
  - XDP is a proposal for adding carpool lanes on the freeway.

Contents from Herbert and Starovoitov XDP presentation.

Potential usecases of XDP

- Pre-stack processing (filtering, DOS mitigation).
- Forwarding and load balancing.
- GRO.
- Flow sampling and monitoring.
- Upper layer protocol (Layer 7) processing.

Cilium using XDP/eBPF for enforcing network policies and load balancing (as a replacement of sidecar proxies).

Some contents from Herbert and Starovoitov XDP presentation.

Key components of XDP

- XDP driver hook
- eBPF virtual machine
- BPF maps
- eBPF verifier

Future Directions

- TCP over XDP
- Zero-copy to user-space
- Performance optimziations
- QoS support
- Debugging support
- ….

Your thoughts?