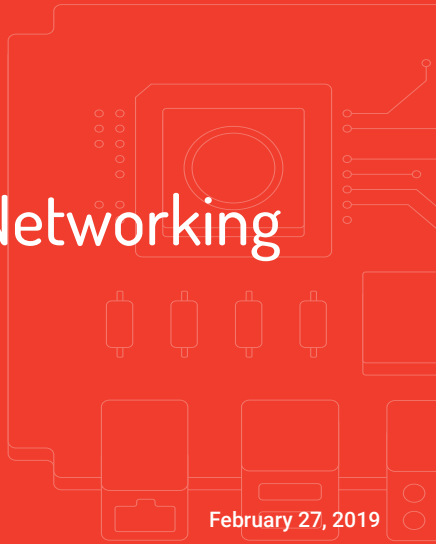
Fast Path for Embedded Networking

Luka Perkov



sartura

February 27, 2019

Sartura

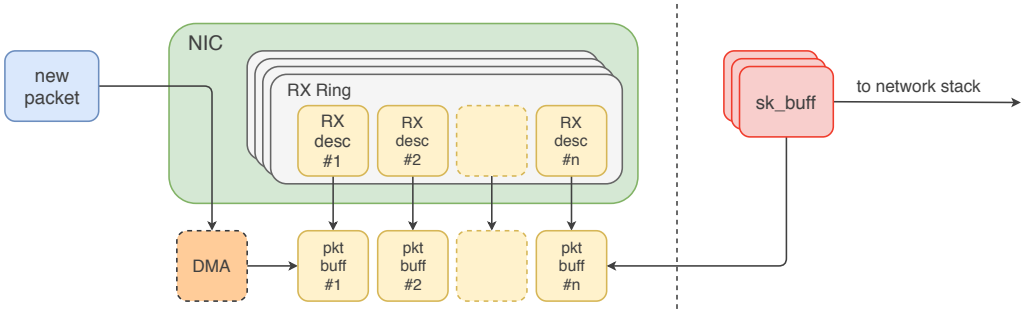
- Delivering solutions based on Linux, OpenWrt and Yocto
  - Focused on software in network edge and CPEs
- Continuous participation in Open Source projects
- Gathering Linux enthusiasts and Open Source supporters

State of fast path networking

- The goal is to leverage the power of off-the-shelf networking hardware
- Insufficient performances of the generic Linux stack
- Emergence of kernel bypass technologies
- Recent in-kernel advancements significantly improve the Linux stack

Why is Linux networking slow?

- Allocation of internal buffer structures consumes bus cycles
- Generic use -> buffers contain metadata for a vast number of protocols
- This complexity slows down overall processing speed
- Userspace executing resource-costly system calls (context switching)
- Good for up to 1 Gbit/s workloads
- Bad for specialized workloads of modern network cards



NIC and kernel packet buffers

Kernel bypass

- Linux kernel performance issues lead to kernel bypass technologies
- Special pathways for direct communication between hardware and userspace
- Main kernel bypass technologies:
  - DPDK
  - netmap
  - Snabb

Kernel bypass – disadvantages

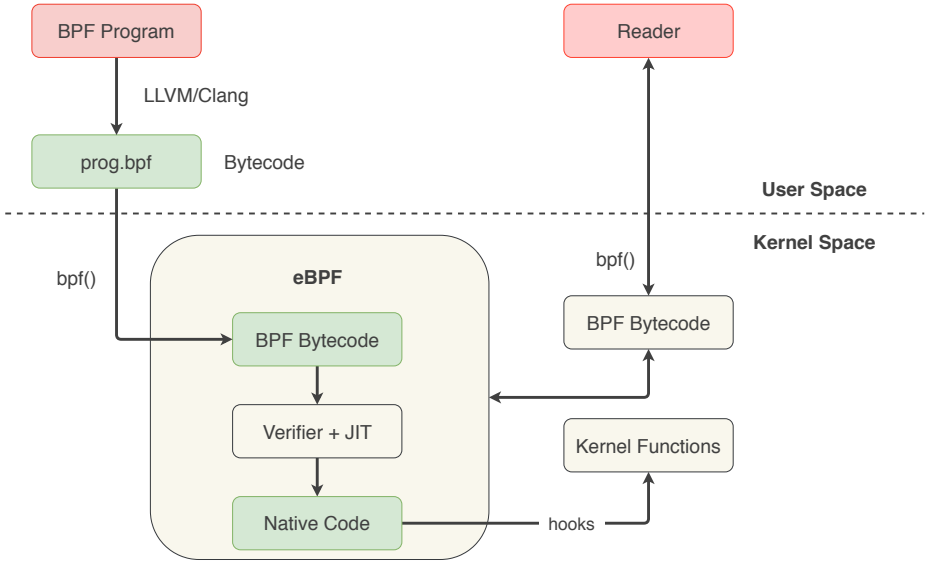
- Often limited to specific hardware
- Parallel and out-of-kernel networking stacks
- Kernel objects remain inaccessible
- Separate APIs, process of writing applications often difficult
- Security concerns

Fast path in harmony with the kernel

- Recent developments of in-kernel technologies: eBPF and XDP
  - In-kernel team's response to kernel bypass technologies
- Easily integrated, programmable
- Redefining performance and security of Linux kernel networking stack

extended Berkeley Packet Filter (eBPF)

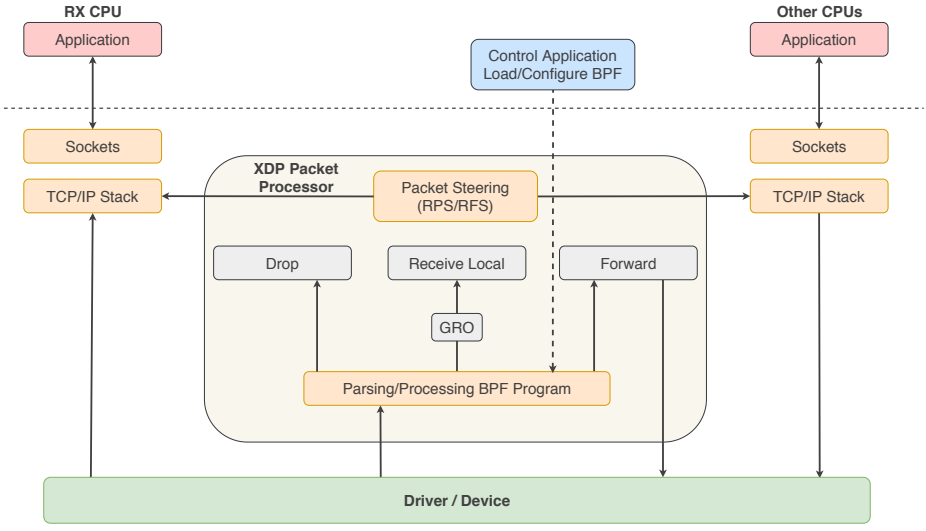
- 11 64-bit registers, JIT compiler, tail calls, BPF verifier
- Strict memory access control
- Highly programmable -> proliferation of eBPF hooks for various purposes:
  - Kernel debugging and performance analysis
  - Attaching eBPF programs to sockets, tunnels, tc subsystem
  - Attaching eBPF programs at earliest networking driver stage (XDP)



eBPF programs

eXpress DataPath (XDP)

- Generic framework for high-performance packet processing
- Integrated in the kernel, shares kernel's security model
- Runs eBPF programs at the lowest level of networking stack
- Eliminates buffer and packet metadata allocation
- New functionality can be added on the fly and without kernel modifications



XDP packet processing

XDP vs kernel bypass

- XDP augments and cofunctions with TCP/IP stack
- No dedicated CPUs with XDP
- Raw packets do not need to be re-injected into the kernel from 3rd party userspace applications
- Userspace networking - no possibility to make security decisions once packets leave the kernel
  - In-kernel BPF code is much more restricted

AF_XDP

- New socket for getting packets into userspace
- Redirecting ingress frames to XDP-enabled network devices
- Enables redirecting frames to buffers in userspace applications (e.g. DPDK)
- Zero copy movement of packet data between userspace and kernel

XDP use cases

- DDoS protection (CloudFlare)
- DDoS scrubber
- Load balancer (Facebook, Cilium)

XDP driver support

Vendor	Broadcom	Cavium	Intel	Mellanox	Netronome	Qlogic	Solarflare	Others
Driver	bnxt	thunderx	ixgbe	mlx4	nfp	qede	sfc	veth
			ixgbevf	mlx5				virtio_net
			i40e					tun

TABLE 1 XDP drivers (as of kernel 4.18)

Sartura & XDP

- Extending and adapting eBPF and XDP to embedded networking
- Industry collaboration to support Marvell's mvneta and mvpp drivers
- Packet drop rate tests
  - Tests for packet drop rates achieved using iptables ruleset, eBPF program and eBPF program compiled wit JIT compiler
  - Tested on Marvell's ESPRESSObin board (ARMADA 88F3700 SoC)
  - 3 test runs on a constant of 1,379,231 pps sent

Test run	iptables	XDP	XDP+JIT
1	184,037	527,483	1,041,677
2	183,155	526,852	1,041,160
3	183,573	527,301	1,040,545
Average	**183,588**	**527,212**	**1,041,127**

- XDP without JIT = 5x iptables drop rate
- XDP with JIT = 10x iptables drop rate
  - Near theoretical 1Gbit link limit of 1.48 Mpps

Conclusion

- eBPF and eBPF subsystems are a powerful shift for Linux networking
- Use-specific optimizations
- Reusing existing kernel infrastructure
- XDP programs executed at earliest networking driver stage
- Vast performance improvements over standard Linux kernel stack
- Sartura actively involved in private and public fast path technology projects

Fast Path for Embedded Networking

luka.perkov@sartura.hr · www.sartura.hr