XDP lightning talk

XDP - eXpress Data Path

Jesper Dangaard Brouer,

Principal Engineer, Red Hat Inc.

Bornhack, Denmark, Aug 2018

Intro: What is XDP?

Really, don't everybody know what XDP is by now?!

- XDP = eXpress Data Path

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 kernel
  - via BPF: makes early network stack run-time programmable
  - Cooperates with kernel

Fast and programmable networking

You have access to these new super powers!

You as Linux users are in control

- Via uploading a BPF program
- You get early access to raw packets
  - Both read and modify
- Take early action
  - Perfect for DDoS protection
  - Facebook use it for Load-Balancing
- The limit is your imagination
  - Within BPF and hook limitation
  - … and kernel community is open to support new use-cases

Intro: XDP: data-plane and control-plane

Overall design

Data-plane: inside kernel, split into:

- Kernel-core: Fabric in charge of moving packets quickly
- In-kernel BPF program:
  - Policy logic decide action
  - Read/write access to packet

Control-plane: Userspace

- Userspace load BPF program
- Can control program via changing BPF maps
- Everything goes through bpf system call

Intro: XDP actions and cooperation

What are the basic building blocks I can use?

BPF program return an action or verdict

- XDP_DROP, XDP_PASS, XDP_TX, XDP_ABORTED, XDP_REDIRECT

How to cooperate with network stack

- Pop/push or modify headers: Change RX-handler kernel use
  - e.g. handle protocol unknown to running kernel
- Can propagate 32Bytes meta-data from XDP stage to network stack
  - TC (clsbpf) hook can use meta-data, e.g. set SKB mark

Intro: Why developers should love BPF

How BPF avoids creating a new kernel ABI for every new user-invented policy decision?

BPF is sandboxed code running inside kernel (XDP only loaded by root)

- A given kernel BPF hook just define:
  - possible actions and limit helpers (that can lookup or change kernel state)

Users get programmable policies (within these limits)

- Userspace "control-plane" API tied to userspace app (not kernel API)
  - likely via modifying a BPF-map
- No longer need a kernel ABI
  - like sysctl/procfs/ioctls etc.

Getting started with XDP

How do you find some example code...

Use LLVM / clang compiler

- Write code in restricted C
  - BPF have some limitations to guarantee safely

Look at examples

- Kernel source dir: samples/bpf/
- My github repo
  - https://github.com/netoptimizer/prototype-kernel/
  - Directory: kernel/samples/bpf/

End slide

... Questions?

plus.google.com/+JesperDangaardBrouer

facebook.com/brouer

linkedin.com/in/brouer

twitter.com/JesperBrouer

youtube.com/channel/UCSypIUCgtI42z63soRMONng

Thanks to all contributors

XDP + BPF combined effort of many people

- Alexei Starovoitov
- Daniel Borkmann
- Brenden Blanco
- Tom Herbert
- John Fastabend
- Martin KaFai Lau
- Jakub Kicinski
- Jason Wang
- Andy Gospodarek
- Thomas Graf
- Michael Chan (bnxt_en)
- Saeed Mahameed (mlx5)
- Tariq Toukan (mlx4)
- Björn Töpel (i40e + AF_XDP)
- Magnus Karlsson (AF_XDP)
- Yuval Mintz (qede)
- Sunil Goutham (thunderx)
- Jason Wang (VM)
- Michael S. Tsirkin (ptr_ring)
- Edward Cree