XDP: The Future of Networks

David S. Miller, Red Hat Inc., Seoul 2017

Overview

History of eBPF and XDP

Why is it important.

“Fake News” about eBPF and XDP

Ongoing improvements and future developments

Workflow of XDP programming

What is eBPF?

Van Jacobson and Steven McCane, 1992, invented classic BPF

Built for very simple packet filtering. You use this in ‘tcpdump’.

The language was very simple.

For a fully functional programming environment we need something more.

Extended, or Enhanced, BPF (eBPF)

An extension of classical BPF

Full instruction set, 64-bit registers and data, etc.

Compilation environments exist to produce eBPF code

Can write programs in C and other languages

eBPF also fundamentally supports “maps” or data structures

eBPF, a Safe Execution Environment

No looping structures are allowed.

Pointer bounds checking is done at program load time.

Arbitrary memory accesses are not allowed.

Access to kernel objects is provided via carefully designed “helpers”

Well then… what is XDP?

XDP is one of many applications of eBPF.

XDP runs eBPF programs on packets when they arrive at networking card.

The result of XDP programs trigger “actions” on the packet:

DROP, TX, PASS, REDIRECT, etc.

XDP can modify packet contents and push/pull headers.

What makes XDP so incredibly fast?

No dynamically allocated packet metadata.

Memory allocation is largest transactional overhead of packet processing.

XDP eliminates this overhead entirely.

Networking stack does no packet parsing and state creation before XDP.

Full overhead of packet processing is the XDP program itself.

Application Spaces for XDP

DDoS protection, both static and dynamic (think ACLs and dynamic rules)

Load balancing.

Switching for building SDN fabrics.

Custom statistics.

Sophisticated traffic sampling via perf events.

Advantages of Using XDP

Inside the kernel and fully integrated with rest of networking stack.

We can choose to give or not give XDP access to objects and tables.

Using helpers and eBPF maps.

Programs execute always in finite time.

Strictly bounded execution time and environment.

Fake News

Some people say certain things about XDP and eBPF.

But they are simply not true.

I will tell the real story and give you the truth.

Is XDP Just a Fad?

No, it is not.

XDP is a long-term architectural solution to many problem spaces.

- High Performance
- Full Programmability
- Kernel Integration and Safety

Is XDP Unsafe?

No, XDP is completely safe. In fact, it is no less safe than userspace.

The eBPF verifier protects us from rogue eBPF programs.

Kernel VM and user process code protects us from rogue userland programs.

There is no difference.

Therefore, saying “userspace is safer” makes no sense.

Is XDP Less Flexible than DPDK?

This is not true, DPDK being in userspace is it’s greatest weakness.

XDP’s full kernel integration means that kernel objects are accessible.

Why is this so important? → Containers.

DPDK has no container story whatsoever.

Is XDP a Replacement for Netfilter, tc, etc.?

The answer here is: Yes, for some things it can be.

XDP is designed for high performance packet processing.

In exchange for this performance, it cannot do everything.

So there are overlapping areas between XDP and existing facilities.

But there are things it cannot do.

XDP Ongoing Development

Introspection, Introspection, Introspection…

Better tools.

Debugging.

More generic support for switching.

Socket cgroup eBPF for socket option tuning.

Arduino and XDP.

Workflow is very similar, and maybe the tools will be at some point too.

Arduino has well defined entry points, and “helpers”, just like eBPF/XDP

You compile and “push” programs onto the Arduino device

The is exactly the same as XDP program development.

Mailing List

xdp-newbies@vger.kernel.org

Send “subscribe xdp-newbies” to majordomo@vger.kernel.org

Plain ASCII text only. Thank you!

Please Come to Netdev 2.2!

Linux Networking Developer conference here in Seoul!

November 8th, 9th, and 10th

https://www.netdevconf.org/2.2/

Meet other developers and see what is happening in the netdev community!

Thank you.

조재흥

NIPA

Alexei Starovoitov, Daniel Borkmann

Thomas Graf, Jesper Brouer, Lawrence Brakmo, John Fastabend

Linus Torvalds