XDP MythBusters

David S. Miller

Overview

What is eBPF and XDP

Why is it an important long term solution to a problem space

Frequently communicated myths and inaccuracies

Things that are true now but will not be true for long

An epiphany on eBPF/XDP workflow

What Is eBPF?

Van Jacobson and Steven McCane back in 1992 saw that at least some part of the kernel needed to be programmable

Result was classical BPF, built specifically for simple packet filtering

Full programmability requires something closer to a real CPU instruction set

Result was an extension to classical BPF, called eBPF

Full compilers exist from C and other languages into eBPF

eBPF also has “maps” or formal data structures (no loops!)

What Is XDP?

One of many applications of eBPF

Runs eBPF programs at the moment the driver receives a packet

XDP programs trigger actions using return codes

XDP_DROP, XDP_TX, XDP_PASS, etc...

XDP programs can modify packet contents

XDP programs can push and pull headers

Why is XDP Fast?

No dynamically allocated metadata

Memory allocation is a significant transactional cost of packet processing

So we just eliminate it

XDP is given only the start and end pointers of the packet area

XDP deals only with linear packet data

XDP has no implicit dependency on state

XDP Problem Spaces

Various forms of DDoS protection

Block by source address and other more sophisticated matches

Load balancing via the XDP_TX return code

Custom statistics

Sophisticated traffic sampling via perf events

High speed trading platforms...

Advantages Of This Approach

It is inside the kernel, therefore fully integrated with the rest of the networking

We can specifically choose to give XDP programs access to objects and tables

Via helpers and eBPF maps

XDP programs always execute in finite time, no looping

Strictly bounded execution time

eBPF/XDP Myth List

This section requires audience participation

If the slide asks a question, you must all answer after I read the question

The appropriate answer is always “Nooooooooooooo……”

Let’s do a test run, OK?

Can XDP slice bread?

“Nooooooooooooooooooooooooo.....”

Is XDP just a fad?

XDP Is A Long Term Architectural Solution

It is specifically designed to properly weigh several different needs

High performance

Full programmability

Kernel integration

Safety

Is XDP unsafe?

XDP Is No Less Safe Than Userspace

The eBPF verifier protects us from rogue eBPF programs

Kernel code implementing user processes and VM protection protects us from rogue userland programs

THERE IS NO DIFFERENCE

So if you argue “userland is safer” be honest with yourself that you just feel more comfortable with userland doing things

And that this is a purely emotional position rather than a factual one

Is XDP less flexible than DPDK?

DPDK Lives Behind the Wall of Userspace

DPDK’s greatest weakness is exactly that it is not integrated into the kernel

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

DPDK’s container story ➜

Is XDP a replacement for netfilter, tc, etc.?

XDP Targets Specific Kinds of Use Cases

XDP is not a Jack of all Trades:

It provides high speed packet operations

Flexible policy decision making

And for this power there is a cost

There are things it cannot do

Is there some overlap between netfilter, tc, and XDP?

The appropriate answer is: YES

Things Which Are Changing, Moving Forward

Introspection (fetching installed XDP programs)

Debugging (dwarf2 is too complicated, looking at CTF)

Once we have debugging, attaching debugging info to maps and perf events

Tracing XDP programs using perf events

More consistent and friendly tooling

Development Using Arduino As A Model

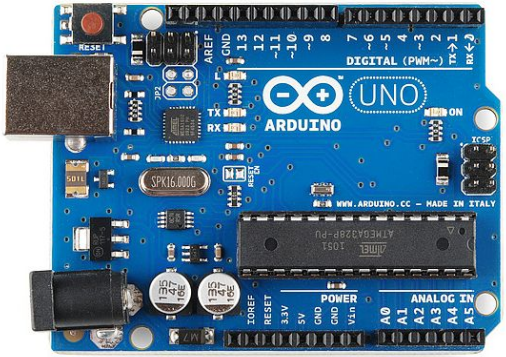
One day I had an epiphany...

Arduino has defined entry points, helper functions, etc.

You compile and “push” programs into the execution environment to run

This is exactly the same as writing XDP programs

The workflow is identical



Thank You

Linus Torvalds

Van Jacobson

Alexei Starovoitov and Daniel Borkmann

Thomas Graf and Tom Herbert

Intel, Mellanox, and Netronome driver developers

Jamal Hadi Salim