History of XDP and the Future of Networking

David S. Miller
Red Hat Inc.

Berkeley Packet Filter (BPF)

Invented by Van Jacobson and Steven McCane in 1992

A simple machine language and virtual machine for filtering

Allows for kernel side filtering of packets

Userspace capture tools can see only the packets they want

All others are skipped before kernel copies them to userspace

Let us call this “Classical BPF”

What Does Classical BPF Look Like?

Example:

```
     ldh [12]
     jne #0x800, drop
     ldb [23]
     jneq #6, drop
     ret #-1
drop:
     ret #0
```

나는 바보입니다.....

I nearly made a **HUGE** mistake.

A new piece of technology seemed unnecessary and pointless to me.

It almost was rejected completely.

But thankfully, it’s author persevered and did not quit.

This has become today’s most exciting development in networking.

Enhanced BPF (eBPF)

Invented by Alexei Starovoitov

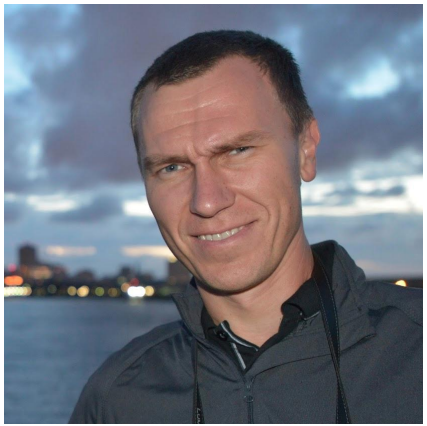
A major extension of Classical BPF

Full 64-bit instructions, registers, and data

Explicit data structure abstraction called “maps”

Can write programs in C and other languages

These programs get compiled into eBPF code



eBPF 대박이에요....

eBPF is a fast, safe, and structured execution environment

No looping structures are allowed

Pointer bounds checking is done at program load time

Programs can only access specific pieces of memory

Access to kernel objects is provided via “helper” functions

eXpress Data Path (XDP)

XDP is one of many uses of eBPF

Runs eBPF program on network packets

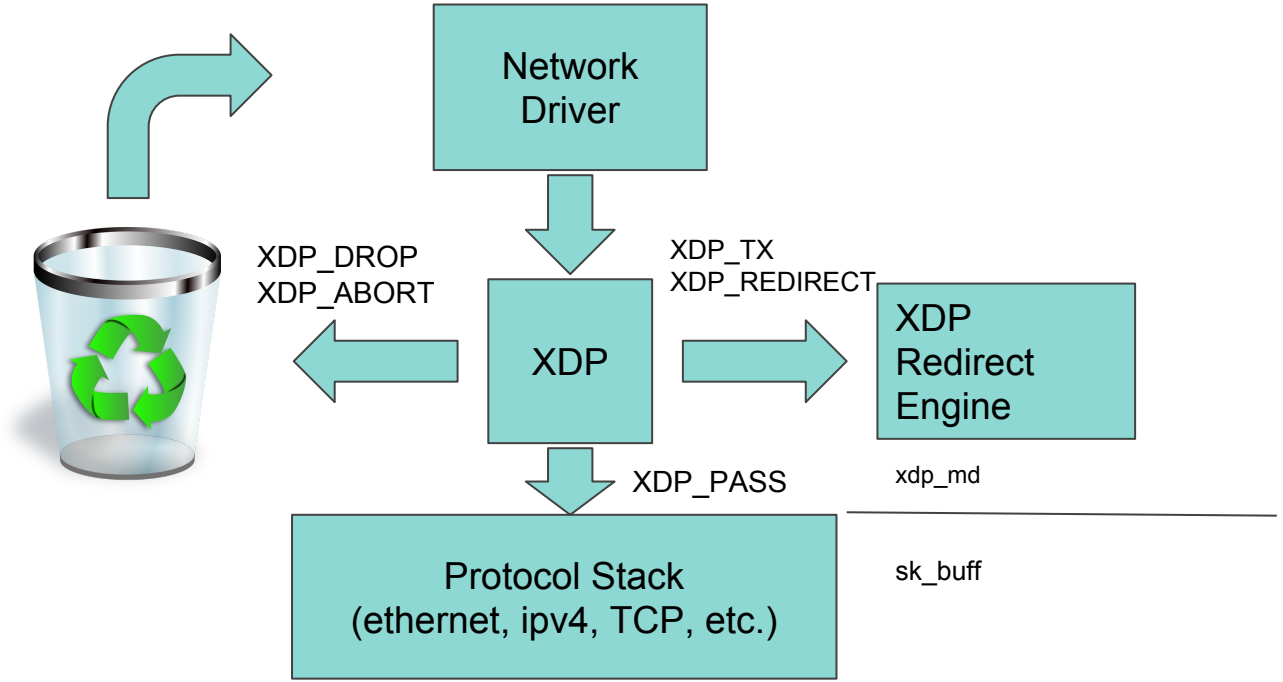
Happens exactly when they are received

XDP program triggers an “action” on the packet

DROP, PASS, TX, REDIRECT, ABORT, etc.

Can modify packet and push or pull headers

XDP Control Flow



XDP 보기

```
SEC(“xdp1”)
int xdp_ex1(struct xdp_md *ctx)
{
     void *data_end = (void *)(long)ctx->data_end;
     void *data = (void *)(long)ctx->data;
     struct ethhdr *eth = data;
     struct iphdr *iph;

     if (data + sizeof(*eth) > data_end)
          return XDP_DROP;
     if (eth->h_proto != htons(ETH_P_IP))
          return XDP_DROP;
 …
```

XDP 보기 (continued)

```
…
    iph = data + sizeof(*eth);
    if (iph + 1 > data_end)
         return XDP_DROP;

    if (iph->protocol != IPPROTO_TCP)
         return XDP_DROP;

    Return XDP_PASS;
}
```

XDP 빨리빨리...

Why is XDP so fast?

- No dynamically allocated packet metadata
  - Largest overhead of packet processing
- No processing or state creation before XDP
- XDP is the first entity to see a packet

Application Spaces for XDP

DDoS Protection, both static and dynamic

Load Balancing

Switching and building SDN fabrics

Custom statistics

Sophisticated traffic sampling via perf events

Advantages of Using XDP

Inside the kernel

Fully integrated with the networking stack

XDP’s access to kernel objects is strictly controlled

Programs execute in finite time

Strictly bounded execution time and environment

The Truth is Out There

Some people say certain things about eBPF and XDP

But they are simply not true

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

Is XDP Just a Fad?

No, it is not.

Long term architectural solution to a problem space:

- High performance
- Full programmability
- Kernel Integration and Safety

Is XDP Unsafe?

XDP is completely safe. 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, being in userspace is DPDK'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.

Does XDP Replace Netfilter, tc, etc.?

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

XDP is designed for high performance packet processing.

In exchange for this performance, it cannot do everything.

There are overlapping areas between XDP and other facilitites.

And this is a good thing.

XDP Ongoing Development

Introspection (what XDP program is running, what maps does it use?)

Better tools

Libraries and code sharing

Debugging (symbol tables and data structure layout)

Redirection and generic switching

Exposing hardware features or “hints” to XDP programs

Mailing List

xdp-newbies@vger.kernel.org

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

Plain ASCII text only, no mime encodings or attachments please!

Thank you!

감사합니다!

조재흥

NIPA

Van Jacobson, Alexei Starovoitov

Jesper Brouer, Thomas Graf, Daniel Borkmann

Linus Torvalds