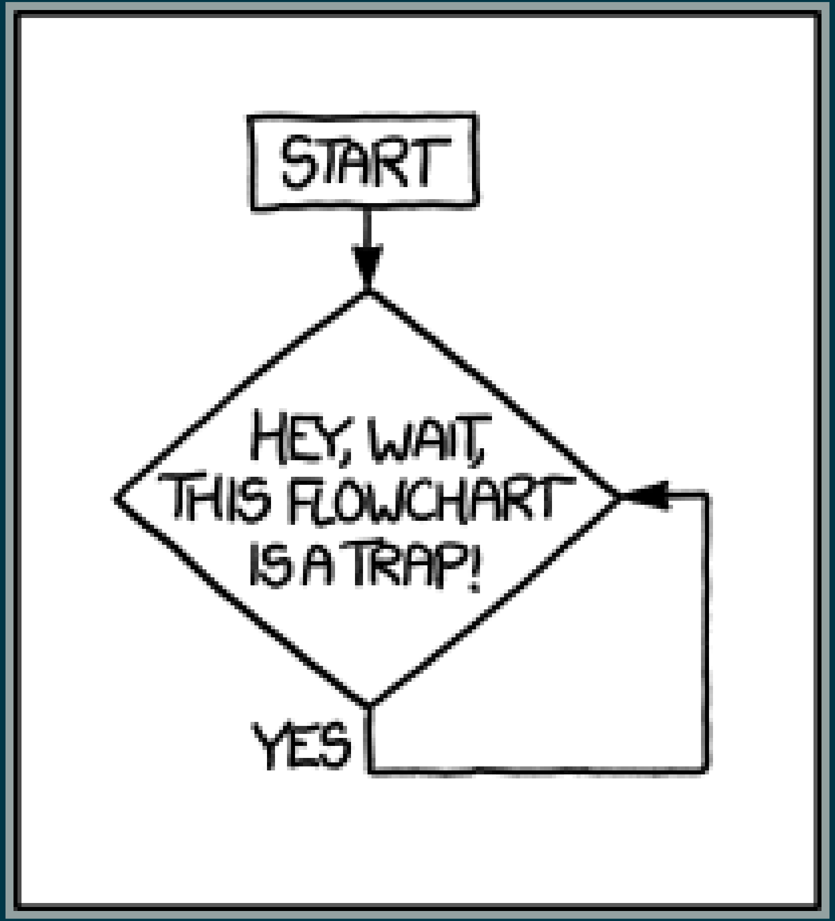
XDP + BPF_PROG_TEST_RUN = A programmable traffic generator!

Toke Høiland-Jørgensen

Principal Kernel Engineer, Red Hat

Lund Linux Con
April 2022

Reminder: How does XDP work?



https://xkcd.com/1195

Hopefully you paid attention during the two previous talks :)

What is BPF_PROG_TEST_RUN?

```
struct { /* anonymous struct used by BPF_PROG_TEST_RUN command */
        __u32           prog_fd;
        __u32           retval;
        __u32           data_size_in;   /* input: len of data_in */
        __u32           data_size_out;  /* input/output: len of data_out
                                         *   returns ENOSPC if data_out
                                         *   is too small.
                                         */
        __aligned_u64   data_in;
        __aligned_u64   data_out;
        __u32           repeat;
        __u32           duration;
        __u32           ctx_size_in;    /* input: len of ctx_in */
        __u32           ctx_size_out;   /* input/output: len of ctx_out
                                         *   returns ENOSPC if ctx_out
                                         *   is too small.
                                         */
        __aligned_u64   ctx_in;
        __aligned_u64   ctx_out;
        __u32           flags;
        __u32           cpu;
} test;
```

Introducing BPF_F_TEST_XDP_LIVE_FRAMES

Introducing BPF_F_TEST_XDP_LIVE_FRAMES

Instead of returning the XDP program result to userspace, act on it!

- If `XDP_PASS`, inject the packet into the stack (at `netif_receive_skb()`)
- If `XDP_REDIRECT` or `XDP_TX`, send packet out an interface (using `ndo_xdp_xmit()`)
- Optimise for batching using the `repeat` parameter (pages are recycled)
  - Note! Page recycling -> not the same data every time!

Comes with documentation! https://docs.kernel.org/bpf/bpf_prog_run.html

What can we do with this?

- Packet injection into the kernel (using `XDP_PASS`)
- Testing `XDP_REDIRECT` infrastructure (like maps)
- Sending packets - XDP-powered traffic generator!

The XDP traffic generator

Live demo time!

The XDP traffic generator

In case the live demo didn’t work:

```
# ./xdp-trafficgen udp ens3f1 # single core
Transmitting on ens3f1 (ifindex 6)
[..]
XDP_REDIRECT    11150720 pkts (   8919659 pps)       696920 KiB (  4567 Mbits/s)

# ./xdp-trafficgen udp ens3f1 -t 6 # 6 cores
Transmitting on ens3f1 (ifindex 6)
[..]
XDP_REDIRECT    65123603 pkts (  52095122 pps)      4070225 KiB ( 26673 Mbits/s)

# ./xdp-trafficgen udp ens3f1 -t 6 -d 100 # spraying over 100 dst-ports
Transmitting on ens3f1 (ifindex 6)
[..]
XDP_REDIRECT     8226576 pkts (  32896120 pps)       514161 KiB ( 16843 Mbits/s)
```

How does it work?

- Prepare packet buffer in userspace
- Pass it to `BPF_PROG_TEST_RUN`
- Return `XDP_REDIRECT` with the right interface
  - For dynamic ports, update the port+checksum first

The simplest case is literally just:

```
SEC("xdp")
int xdp_redirect_notouch(struct xdp_md *ctx)
{
        return xdp_stats_record_action(ctx, bpf_redirect(config.ifindex_out, 0));
}
```

Can we do TCP as well?

Yes!

- Set up connection from userspace
- Install XDP program on interface to intercept replies
  - Process ACKs in XDP program, update state and drop packets
- Use `BPF_PROG_TEST_RUN` to send out TCP packets
  - Return `XDP_DROP` when running up against the CWND

```
Connected to fe80::ee0d:9aff:fedb:11cd port 1234 from fe80::ee0d:9aff:fed8:f5d3 port 39500
[...]
Period of 1.000081s ending at 1652215664.276825
XDP_DROP          3249504 pkts (     23878 pps)     4760015 KiB (   287 Mbits/s)
XDP_PASS                0 pkts (         0 pps)           0 KiB (     0 Mbits/s)
XDP_TX          516392331 pkts (   6217106 pps)   756434078 KiB ( 74605 Mbits/s) <--- retransmissions!
XDP_REDIRECT    545096864 pkts (   6534151 pps)   798481734 KiB ( 78410 Mbits/s)
```

Raw throughput (single core): 78.4 Gbps! Goodput: ~4 Gbps!

End: Questions?

Why not try out XDP live packet mode yourself? What else can we use it for?

XDP-tools: github.com/xdp-project/xdp-tools