XDP-workshop agenda/ideas

Please add your own topic to the XDP-workshop in this document.

When is conference

XDP Workshop - Netdev Conf 0x15
- When : Tue, July 13, 17:00 – 19:00 (CEST)
- Where : https://www.airmeet.com/e/6d8af870-d2c0-11eb-9d4c-a1fc389c9f34
- Session link: https://netdevconf.info/0x15/session.html?XDP-General-Workshop

Topic: XDP multi-buff

Persons involved: lorenzo.bianconi@redhat.com,
Why XDP multi-buff:

- Motivation: Lift XDP limitations on MTU and jumbo-frame configs
- Longer-term: Allow **veth** to receive/send GRO SKB's with XDP-prog attached
- Long desc in design document (which is need to be updated to reflect upstream discussions)

add-xdp-on-driver.pdf (slide 39)

Sub-topic#1: How to get total packet len?

The current way to get XDP packet length is the difference between `data` and `data_end`.

```
pkt_len = ctx->data_end - ctx->data;
```

Or copy-paste from samples/bpf:

```
static __always_inline int handle_ipv4(struct xdp_md *xdp)
{
	void *data_end = (void *)(long)xdp->data_end;
	void *data = (void *)(long)xdp->data;
	int pckt_size = data_end - data;
	[...]
```

Proposed eBPF helper:
https://github.com/LorenzoBianconi/bpf-next/commit/6cd9de460eb5ee5fdd009fbdc25da39592e2c79c

Sub-Topic#2: Backwards compatibility for XDP multi-buff

See Toke's email: https://lore.kernel.org/netdev/8735srxglb.fsf@toke.dk/

Sub-Topic#3: API to access payload-data in packet fragments?

This seems to be blocking the patchset.

Why is it controversial upstream?

- The XDP model BPF-prog has Direct Access (DA) to packet data payload.
- Direct Access (DA) assumes physical contiguous memory (for verifier to prove memory access is safe).

Topic: XDP-hints

Persons involved: brouer@redhat.com, Alexander Lobakin
Why XDP-hints:

- NICs support a range of HW offload “hints” that isn’t available to XDP
    - HW-hints e.g.: checksum offloading, vlan offloading, timestamps, mark, flow identifier, RSS-hash
- GOAL: Create flexible layer between NIC drivers and XDP+netstack
    - NIC hardware and drivers are still inflexible (regarding struct offset storing)
    - BTF provides flexibility to let NIC driver describe offsets it uses
    - Drivers only standardize on struct *member names*, offset is flexible per NIC
    - Netstack also benefits when converting xdp_frame to SKB

Mailing-list: xdp-hints@xdp-project.net

- https://lists.xdp-project.net/postorius/lists/xdp-hints.xdp-project.net/
- https://lists.xdp-project.net/xdp-hints/

General consensus upstream: **BTF should be used for describing metadata area(s)**.
Many opinions on how exactly to use BTF:

1. Jesper idea: Drivers export multiple BTF-IDs with simple-layouts
    - Drivers internally have complex C-struct, but stamp ctx with BTF-ID which describe what fields are valid.
2. John Fastabend: Export one BTF-ID with a complex-layout containing flags member telling consumers which fields are valid.
    - These flags need to be UAPI exported and same across all drivers

Notes on things to cover:

- Remember to wrap your head around: That metadata area "grows" via minus offset as `ctx->data_meta` points to the area before `ctx->data`.
- Design should cover AF_XDP using regular userspace C-code (no auto BTF rewrites like BPF-progs)
- BTF struct member NAMES becomes UAPI, e.g. ‘rxhash32’ must have the same semantic meaning (and size) across drivers.
- XDP-hints both on RX and TX
    - Most see XDP-hints as a RX feature
    - TX side is also needed: e.g. TX-csum-req, TX-vlan-offload, LaunchTime

Different places/steps where XDP-hints are used or converted:

- XDP-hints **RX-step** happens (in driver) BEFORE calling XDP “RX”-hook
  - This gives XDP-prog access to the XDP-hints
- XDP-hints **TX-step** happens AFTER the (non-existing) XDP “TX”-hook.
  - This allows XDP TX-prog to adjust/correct HW-TX-hints features
    - (Special interest in LaunchTime)
  - **TX-step** transfer XDP-hints into the driver-specific TX-descriptor
    - See how LaunchTime have diff bits avail per NIC chip
  - (Up-for-discussion as XDP “TX”-hook doesn’t exist yet!)
- XDP-hints xdp_frame convert **to-SKB-step**
  - Transfers XDP-hints into SKB fields
- AF_XDP general
  - Use 1-bit in AF_XDP descriptor to indicate XDP-hints (with BTF)
    - No room for BTF-ID in descriptor
    - BTF-ID is stored in-frame as `ctx->data` minus 4 bytes (32-bit)
- AF_XDP **receive** XDP-hints
  - Userspace have to code own decode of BTF-offset and keep track of offset per BTF-ID to read out struct members that are relevant fpr RX-prog
- AF_XDP **transmit** XDP-hints
  - Ask NIC what TX-BTF format it supports
  - Format XDP-hints metadata correctly
  - Drivers **TX-step** (above) does exactly same for AF_XDP frames

- TX-completion: Could we make data avail after DMA TX-completion event
  - What about data that is available after transmission, such as HW TX timestamp? Should it go into the hints and be retrieved when the frame is back to the completion queue?

Topic: XDP features detection

There’s no easy way to detect what XDP features the currently running kernel supports, and perhaps more importantly, what XDP features a given network interface supports. This makes writing portable solutions hard to impossible.

Use cases:

- Portable data plane solution.
- Suricata wants to detect if a user's config (e.g. XDP cpumap or AF_XDP feature) is available before applying config.

Topic: QoS and Queuing

Persons involved: toke@redhat.com, lorenzo.bianconi@redhat.com, Freysteinn.Alfredsson@kau.se

The problem we're trying to solve is that XDP currently have no way to queue or reorder packets: when you redirect a packet to another device, it is immediately pushed to hardware (except for a bulking queue of 16 packets), and if there's no room in the hardware queues, the packet is just dropped.

This is obviously not a good situation: it means we can't do any kind of traffic policies, classful queueing, etc. And rate transitions are not going to work well: if you redirect from (say) a 100G interface into a 10G interface you'll just drop 90% of the traffic on the floor without having any say over which 90% that is.

So what we're trying to do is introduce a queueing layer to XDP, like we have qdiscs for the kernel stack. We can't just reuse the qdisc layer, because that is tied to the skb data structure, and besides it has way too high overhead for XDP. So we're instead trying to build an XDP-specific layer which is light-weight but still flexible and fully programmable (using BPF). The idea is to build this on top of recent(ish) academic work on programmable queueing primitives, specifically the PIFO data structure[0] and its follow-on Eiffel software realisation[1].

Our idea for realising this for XDP is to create a new map type that realises the PIFO abstraction, which can then be used as an XDP_REDIRECT target. I.e., XDP programs will be able to enqueue packets into a PIFO by redirecting them. The tricky part then becomes, how do we get the packets back out? Here we will probably need a new hook that runs in the driver when it is ready to transmit, where a BPF program can instruct the driver which PIFO to dequeue packets from. We have some ideas here, but it's still very much up in the air. We're targeting LPC in September for presenting something a bit more concrete, so for this workshop, the idea is just to introduce people to the idea and hear if anyone has any immediate feedback.

[0] http://web.mit.edu/pifo/
[1] https://www.usenix.org/conference/nsdi19/presentation/saeed

Topic: Issue TX-queues less-than CPUs

Persons involved: **All NIC vendors**

Issue: The most common problem is on a system with many CPUs, where often the number of NIC queues has to be extended to match at least the number of CPUs.

Inconsistent NIC driver behavior, **when NIC queues cannot match number of CPUs**:

- Some NIC drivers: **Deny** loading XDP-prog
- Some NIC drivers: **Drop** XDP redirected packets silently
- Some NIC drivers: **Report** drops in dmesg
- Some NIC drivers: **Warn** in dmesg and drop packets from CPUs with high smp-ID
- Some NIC drivers: **Crash**

E.g. upstream work on sfc driver by ihuguet@redhat.com

Topic: Issue NIC counters inconsistent for XDP

Persons involved: **All NIC vendors**

We need consistency in the counters across NIC drivers and virtual devices.

General idea: Always increment normal netdev stats on RX-packets, even for XDP_DROP packets. Reason is that XDP_DROP is a policy choice on the (local) machine. (The normal netdev stats are exported to different tools e.g. snmpd, which will show up in many monitoring graphs, which will be misleading if XDP_DROPs are not included).

Normal regular **netdev** counters should be updated:

- Any **RX-packet** even on **`xdp_drop`**: reported in RX stats
- **`xdp_tx`** and **ndo_xdp_xmit()**: reported in TX stats

ethtool counters:

- Dedicated counters for xdp verdicts + some hard to catch error counters

See add-xdp-on-driver.pdf (slide 38) on XDP-stats and ethtool XDP verdict names used in driver `mvneta` (board espressobin).

See output from:

- `root@espresso-bin:~# ethtool -S eth0 | grep xdp`

Recommended XDP **ethtool** verdict names:

ethtool --stats name	Description
rx_xdp_redirect	(self describing)
rx_xdp_pass	(self describing)
rx_xdp_drop	Incl. xdp_aborted + unknown verdict to simplify driver code. Drivers can choose to impl. xdp_aborted (incl unknown fallthrough). Argument: Sysadm can debug verdicts with xdpdump or tracepoint
rx_xdp_tx	Fast XDP RX-to-TX bouncing via verdict `XDP_TX`
rx_xdp_tx_errors	Helps to catch driver queue overruns (otherwise hard to catch)

tx_xdp_xmit	Pkts from other drivers via ndo_xdp_xmit
tx_xdp_xmit_errors	Driver level errors after ndo_xdp_xmit (otherwise hard to catch)

Discussion: Should drivers implement ethtool rx_xdp_aborted (incl fallthrough) ?

- (note down participants opinions here)

Already updated drivers:

- mvneta
- mvpp2
- ti
- mlx5
- veth
- socionext
- ...

Topic: Connection tracking

Most of the current data plane use cases require connection tracking.
There are two options to implement it:

1. An eBPF library. It seems Cilium went this route. Advantage: no kernel changes needed. Disadvantage: the state is not shared with the slow path and doesn't play in-concert with network stack (and existing tools).
2. A connection tracking helper. This requires bpf helpers in modules, which is currently not possible. Alexei summarized his requirements for such infrastructure: https://lore.kernel.org/bpf/20200130215330.f3unziderf5rlipf@ast-mbp/

Implementation details:

- The conntrack helper should be designed to work with both XDP and TC-BPF.
- A conntrack helper working in-concert with netstack can fallback to netstack on e.g. fragmented IP-packets. (Thus, allowing XDP to focus on fast-path with a smaller code base).

Topic: Debugging XDP program problems

<proposed topic>
Share your Best Known Methods (BKM) to debug and / or figure out what's going wrong when you're trying to develop or use an XDP program on multiple vendors' drivers.

Tool: xdpdump (developed by `echaudro@redhat.com`)

- Part of https://github.com/xdp-project/xdp-tools (packaged as 'xdp-tools' fedora+RHEL)
- tcpdump like tool for capturing packets at the XDP layer.
- For debugging XDP programs that are already loaded on an interface.

- Packets can be dumped/inspected before on **entry to XDP** program, or **after at exit** from an XDP program. Furthermore, at exit the XDP action is also captured. This means that even packets that are dropped at the XDP layer can be captured via this tool.