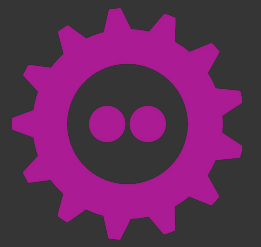


XDP and page_pool allocator

...let's go fast!

Ilias Apalodimas
Linaro

Lorenzo Bianconi
Red Hat

Linaro

Red Hat

About us:

- Ilias:
  - Serving as a co-maintainer for page_pool API - ilias.apalodimas@linaro.org
  - Added XDP and page_pool support to netsec driver
- Lorenzo:
  - Software engineer @ Red Hat, driver maintainer - lorenzo@kernel.org
  - Added XDP and page_pool support to mvneta driver

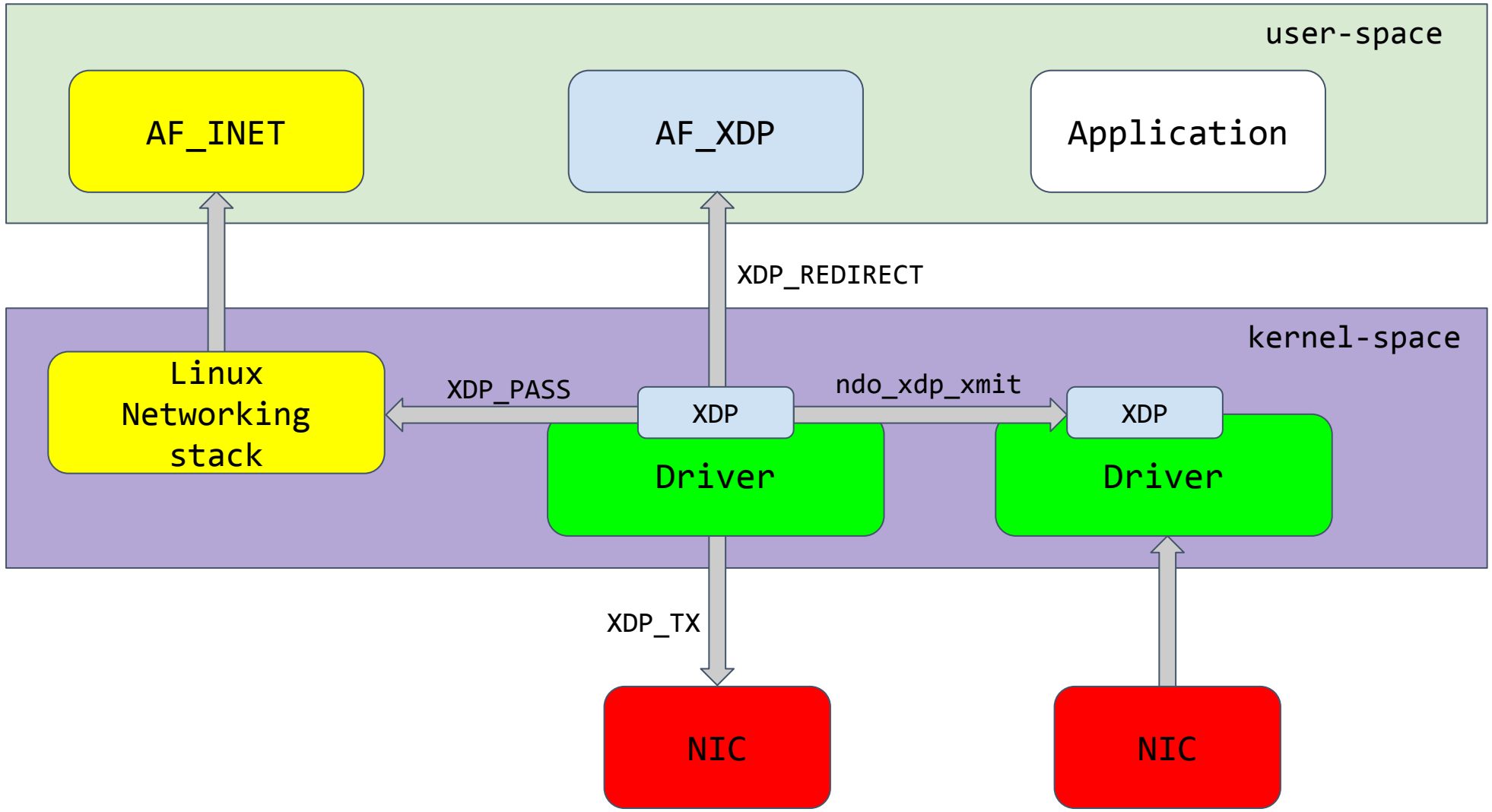
Linaro Red Hat

What is XDP?

XDP (eXpress Data Path) is a Linux in-kernel fast path:

- XDP can be considered as a software offload layer for the kernel networking stack
- Driver level (RX) hooks at DMA layer before skb allocation
  - Just after `dma_sync_*_for_cpu()`
- XDP designed for speed:
  - Operates @ L2-L3 while the networking stack works @ L2-L7
  - Skip unnecessary processing in the networking stack (route lookup, netfilter hooks, etc.)
- Not a bypass but in-kernel fast path:
  - Leverages existing kernel infrastructure
  - Programmable flexibility via eBPF sandboxing
- XDP use-cases:
  - Anti-DDoS (Facebook, CloudFlare)
  - L4 Load Balancer (Katran - Facebook)
  - etc.

Linaro Red Hat



Linaro
Red Hat

XDP requirements

XDP memory model:

- XDP frame in physical contiguous memory
  - BPF Direct-Access for validating correctness
  - Read and Write access to the DMA buffers
  - Disable jumbo frames loading a BPF program
  - No paged frames support, data cannot be split across pages
- Linear DMA pages must provide space for metadata
  - XDP headroom to push/pop header through `bpf_xdp_adjust_head()`
  - `XDP_PASS`: need to reserve tailroom for `skb_shared_info` to rely on `build_skb()`
- Cannot allocate page fragments to support it (e.g. through `napi_alloc_skb()`)
- Rx buffers must be recycled to get high speed!

Linaro Red Hat

page_pool allocator

- Optimized for one packet per page
  - Supports split-page model (usually driver is in charge of recycling)
- Native buffer recycling
  - in-irq cache and ptr_ring cache
  - Currently supported for `XDP_DROP` but `XDP_PASS` is coming
- Allocates order^n pages (usually order-0 = 4K page)
- Fast, usually runs in NAPI context, no extra locking overhead
- DMA management can be done via the page_pool API
  - DMA-mapping capability (keeps page mapped)
  - DMA-sync for cache non-coherent devices

Linaro Red Hat

Caveats

- Although page_pool is faster due to native recycling for XDP, it's slower for skb
- Bigger memory footprint (linear to number of descriptors used), unless page splitting is implemented
- Working on native skb recycling, which should eliminate the skb use case penalty
    - The current mvneta driver was already allocating a page per packet. The recycling patches boost performance by ~25% on < 512b packets

Linaro Red Hat

“...talk is cheap. Show me the code”

- Intel and Mellanox XDP implementations are complex (naturally since it’s a complex hardware)
- mvneta (marvell 1Gbit) and netsec (socionext 1Gbit) can serve as a simplified guideline on how to add XDP support
- We need all XDP verdicts covered to accept a driver
  - `XDP_DROP`
  - `XDP_TX`
  - `XDP_PASS`
  - `XDP_REDIRECT, ndo_xdp_xmit()`

Linaro Red Hat

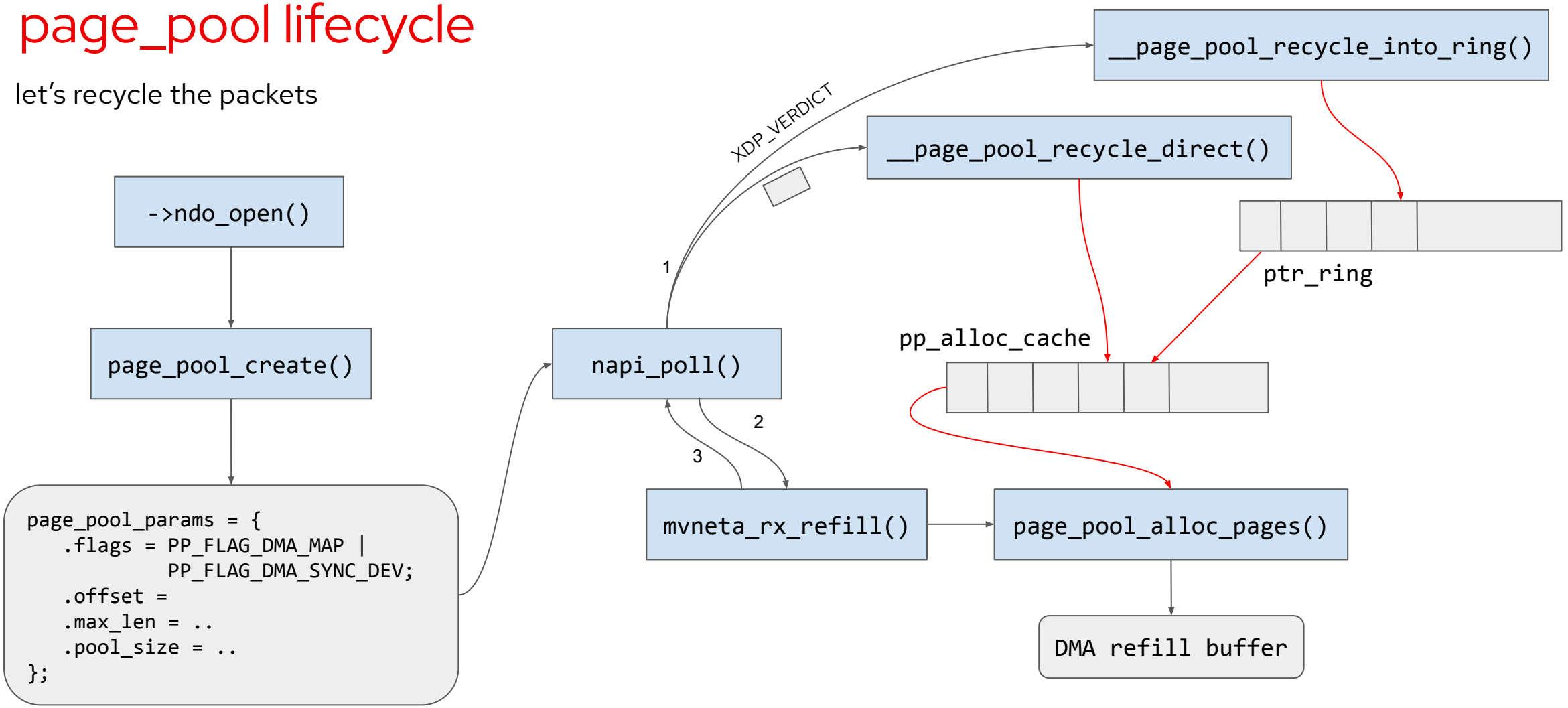
Marvell ESPRESSObin - mvneta

SoC	Marvell Armada 3700LP (88F3720) dual core ARM Cortex A53 processor up to 1.2GHz
System Memory	1 GB DDR3 or optional 2GB DDR3
Storage	1x SATA interface 1x micro SD card slot with footprint for an optional 4GB EMMC
Network Connectivity	1x Topaz Networking Switch 2x GbE Ethernet LAN 1x Ethernet WAN 1x MiniPCIe slot for Wireless/BLE peripherals
USB	1x USB 3.0 1x USB 2.0 1x micro USB port
Expansion	2x 46-pin GPIO headers for accessories and shields with I2C, GPIOs, PWM, UART, SPI, MMC, etc.
Misc	Reset button, JTAG interface
Power supply	12V DC jack or 5V via micro USB port
Power consumption	Less than 1W thermal dissipation at 1 GHz

Linaro Red Hat

page_pool lifecycle

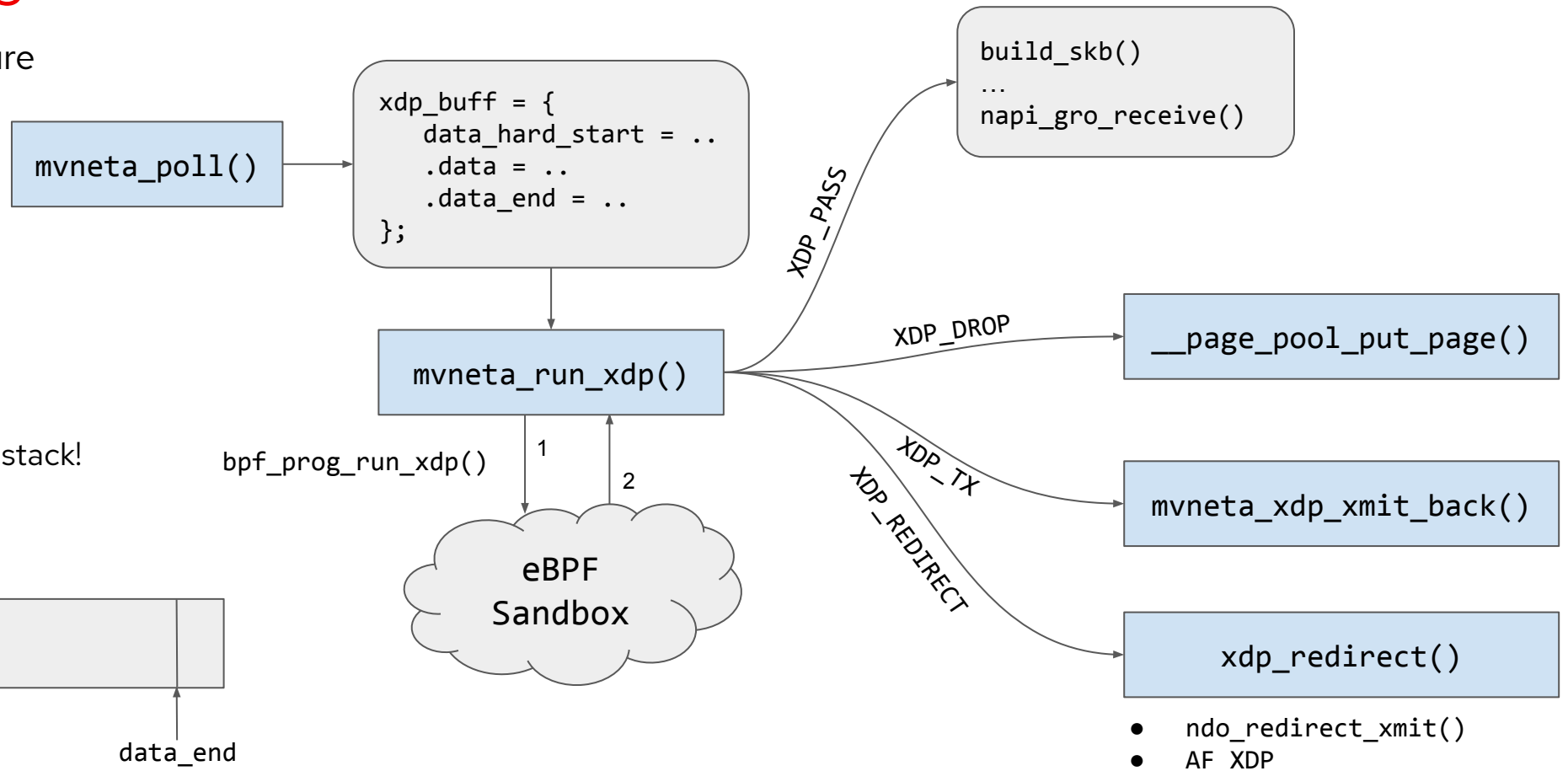
let's recycle the packets



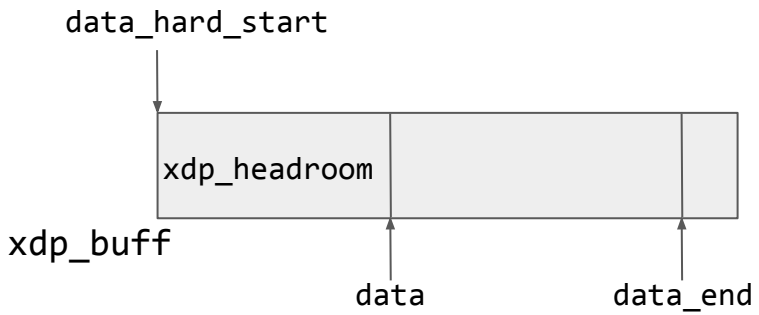
Linaro Red Hat

XDP verdicts

mvneta XDP architecture



`xdp_buff` is on the stack!

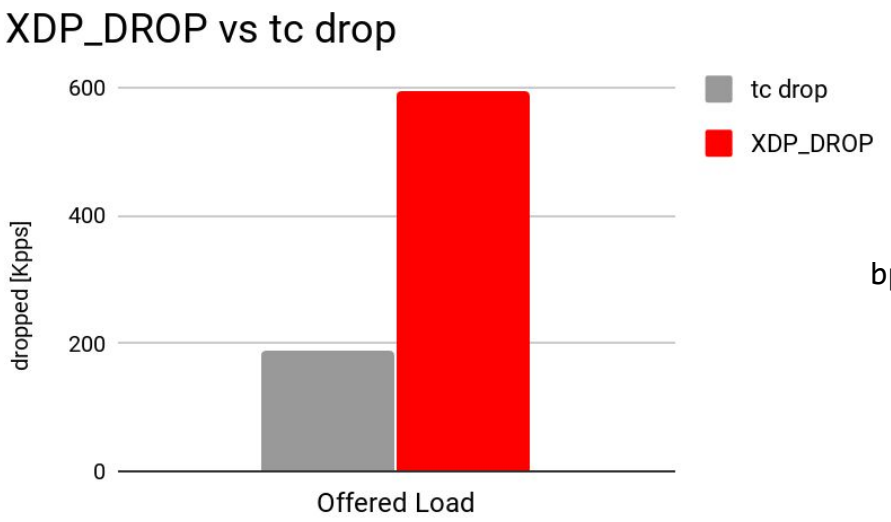


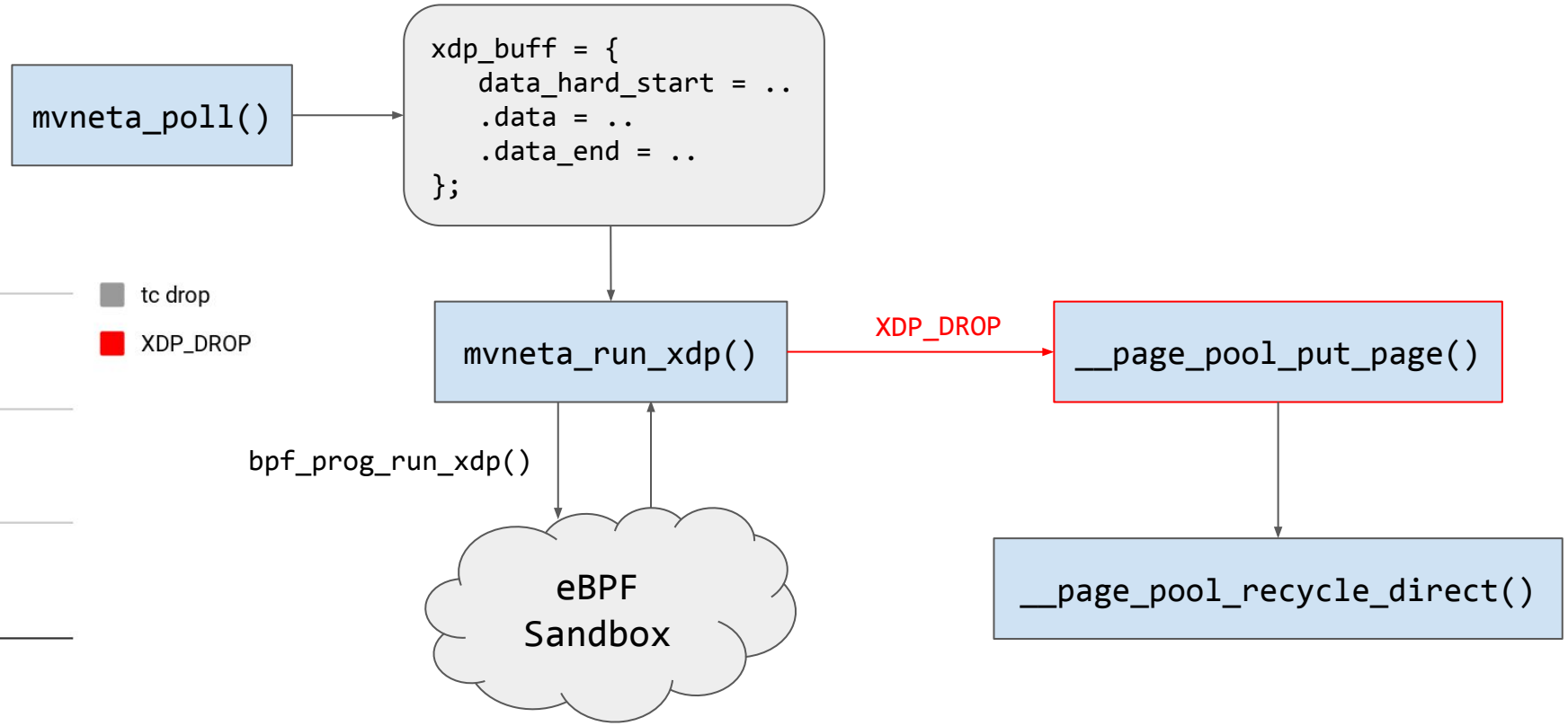
- `ndo_redirect_xmit()`
- `AF_XDP`
- etc.

Linaro Red Hat

XDP_DROP

let's drop as fast as we can





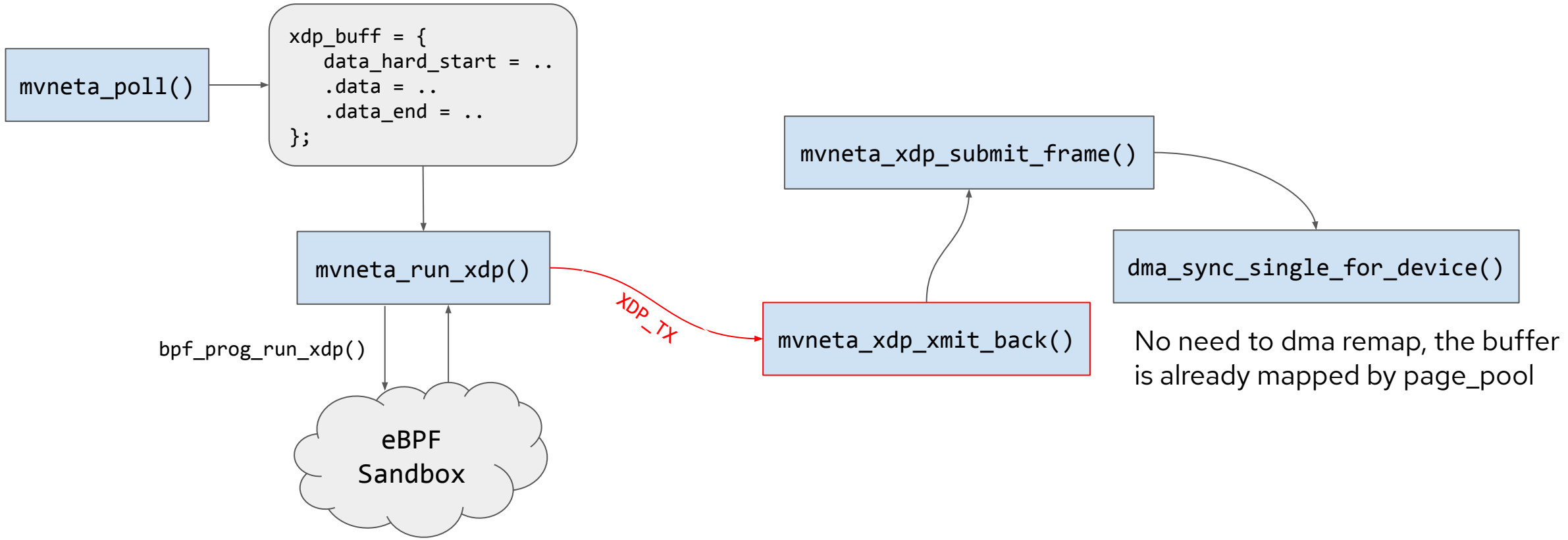
- packet size: 64B
- `$tc qdisc add dev eth0 clsact`

  `$tc filter add dev eth0 ingress matchall action gact drop`

Linaro Red Hat

XDP_TX

let's send the packet back

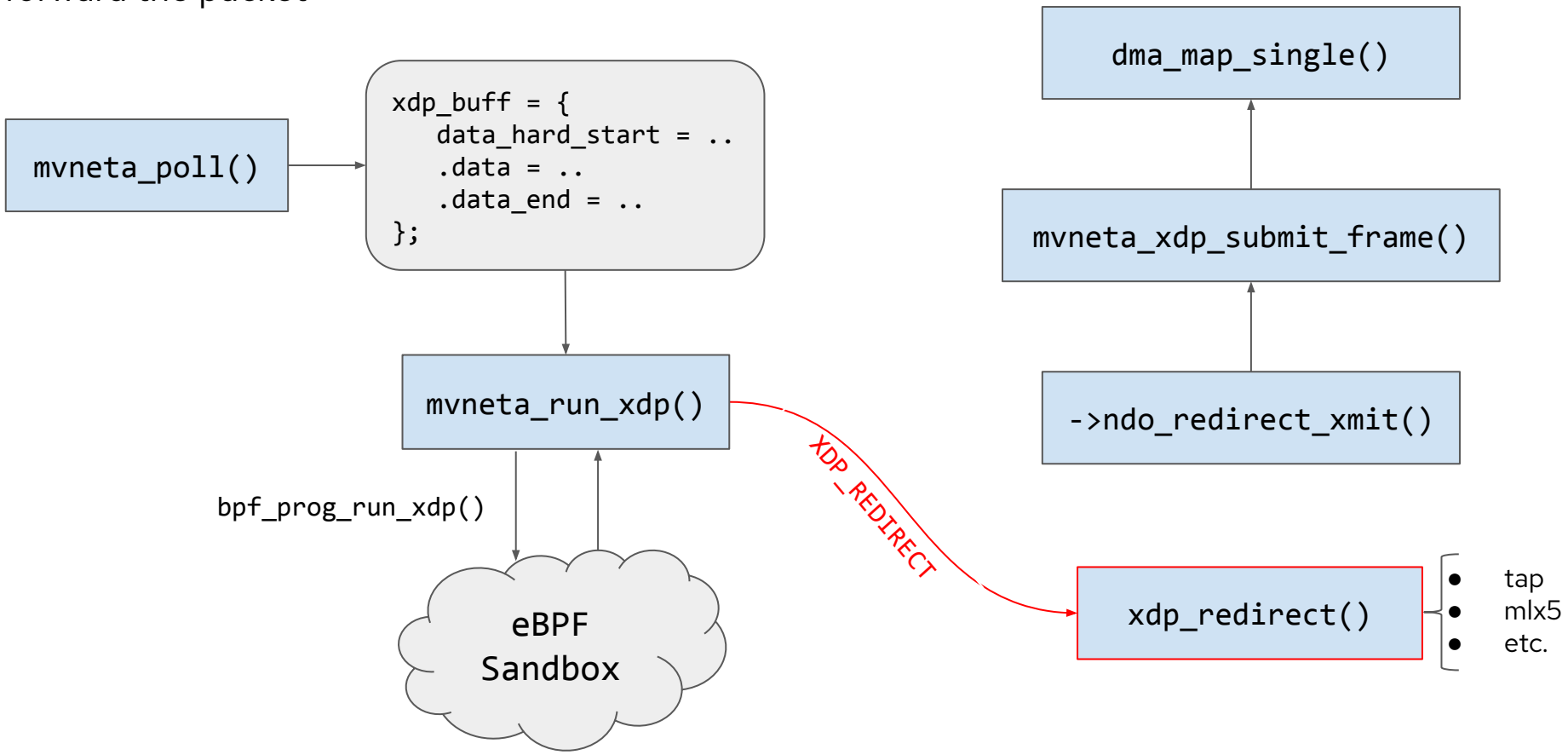


No need to dma remap, the buffer is already mapped by page_pool

Linaro Red Hat

XDP_REDIRECT

let's forward the packet

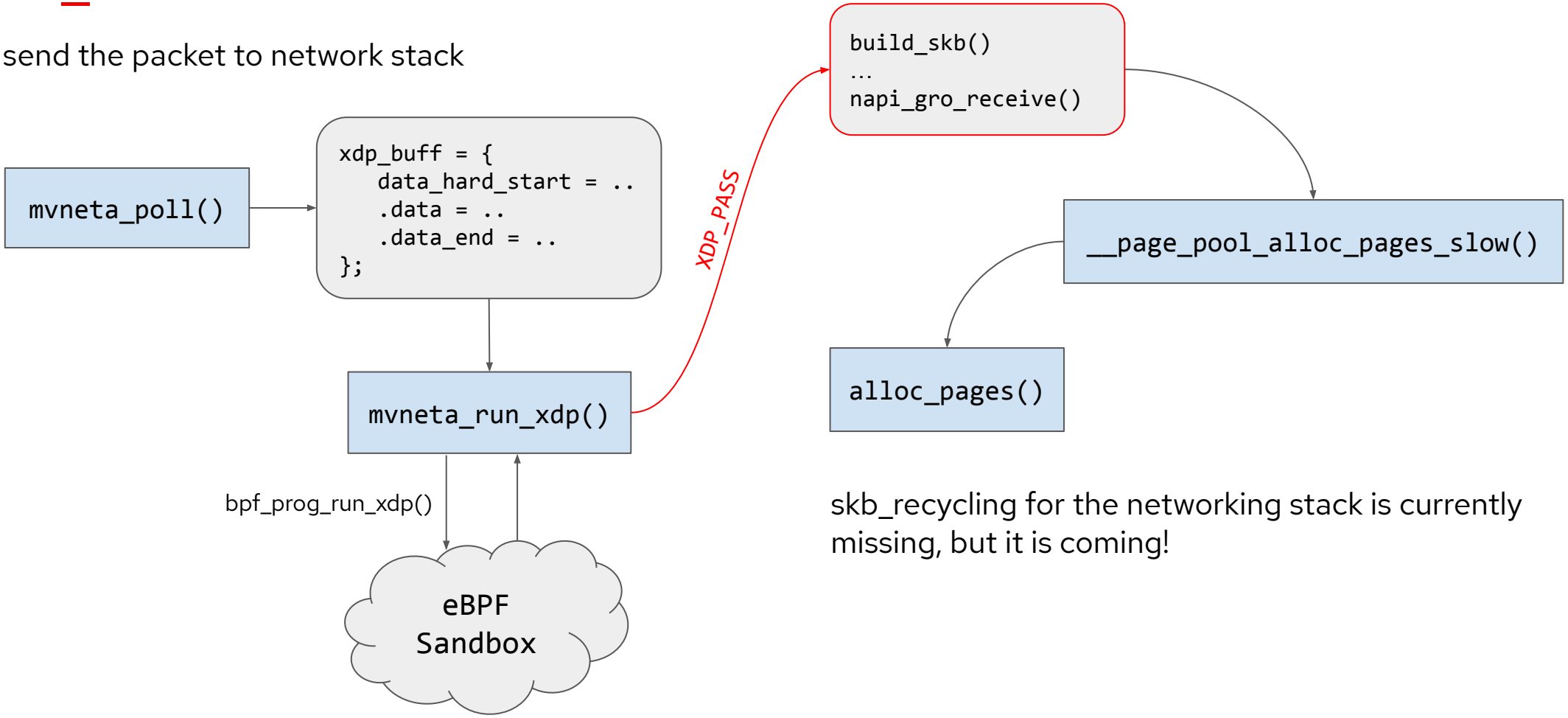


Buffer needs mapping before sending

Linaro Red Hat

XDP_PASS

let's send the packet to network stack



skb_recycling for the networking stack is currently missing, but it is coming!

Linaro Red Hat

Conclusions

- XDP can be considered as a software offload layer for the kernel networking stack
- XDP memory model
    - Contiguous memory area
- page_pool allocator
    - DMA buffer recycling
- mvneta XDP architecture
    - `XDP_DROP`
    - `XDP_TX`
    - `XDP_REDIRECT`
    - `XDP_PASS`
- Future work
    - **skb recycling for `XDP_PASS`**
    - mvneta: XDP support for hw buffer manager (e.g. ClearFog)
    - mvneta: XDP roadmap
    - mvneta: native `AF_XDP` support

Linaro Red Hat

Q&A:

Linaro Red Hat

Thank you

- https://github.com/xdp-project
- Contact:
  - ilias.apalodimas@linaro.org
  - lorenzo@kernel.org

Linaro Red Hat