# XDP ACCELERATION USING HW-BASED HINTS

PJ Waskiewicz, Anjali Singhai Jain
Networking Division

Nov 2017

# Agenda

- XDP software model
- HW hints in XDP programs
- Initial Performance Results
- Metadata layout considerations
- Programming HW hints
- Dynamically requesting hints with eBPF
- Wrap-up/Questions

# XDP Software Model

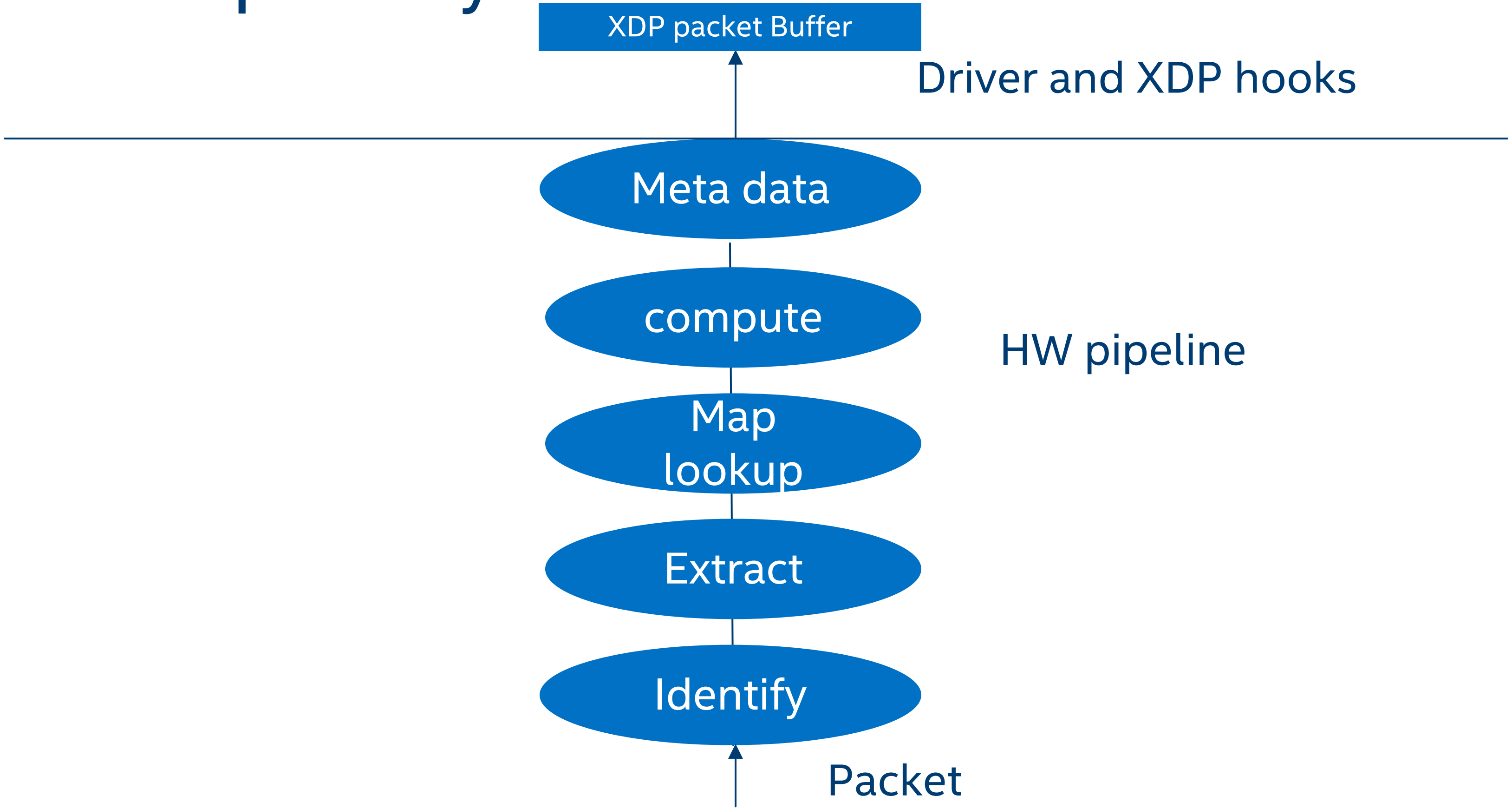
- XDP programs are continuing to evolve and are becoming more complex

- Each XDP program does packet parsing to:
  - Identify the packet type and extract packet header information
  - Based on the use-case then the XDP program
    - may monitor incoming traffic on the network
    - manipulate packets
    - compute hash or xsums for modified packets
    - make packet forwarding decisions based on some map lookups (DROP/PASS/TX/REDIRECT/etc.)

# Our Goal

- What can present-day HW do to help
  - Accelerate what is being done in XDP programs in terms of packet processing
  - Offset some of the CPU cycles used for packet processing
- Keep it consistent with XDP philosophy
  - Avoid kernel changes as much as possible
  - Keep it HW agnostic as much as possible
  - Best effort acceleration
  - A frame work that can change with changing needs of packet processing
- Expose the flexibility provided by programmable packet processing pipeline to adapt to XDP program needs
- Help design the next generation Hardware to take it a notch up!

# HW Capability

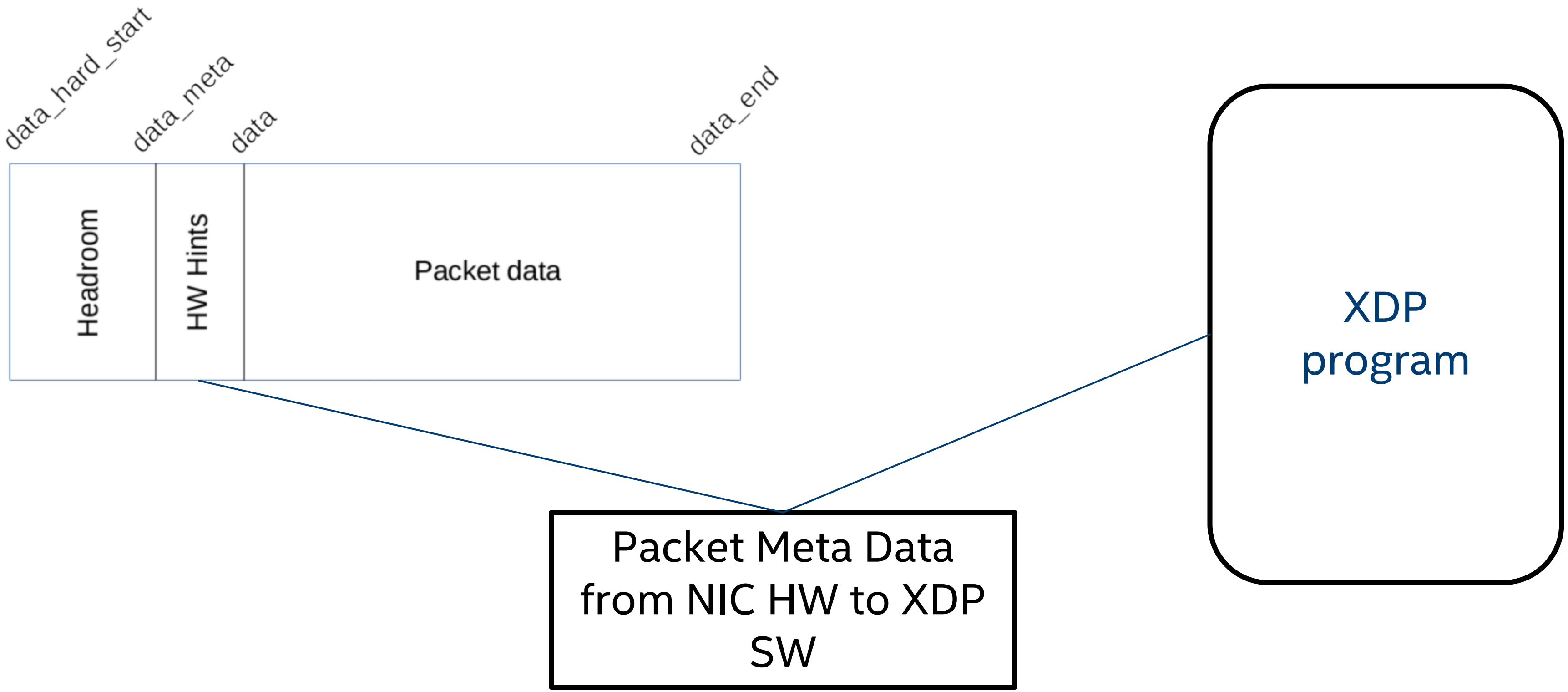


# Two problems to solve

- How do you dynamically program the Hardware to get the XDP program the right kind of packet parsing help?

- How to pass the packet parsing/map lookup hints that the HW provides with every packet into the XDP program so that it can benefit from it?

# HW hints flow



# Performance improvements

- Internal testing yielded promising results
- Test setup:

Target: Intel Xeon E5-2697v2 (Ivy Bridge)

Kernel: 4.14.0-rc1+ (net-next)

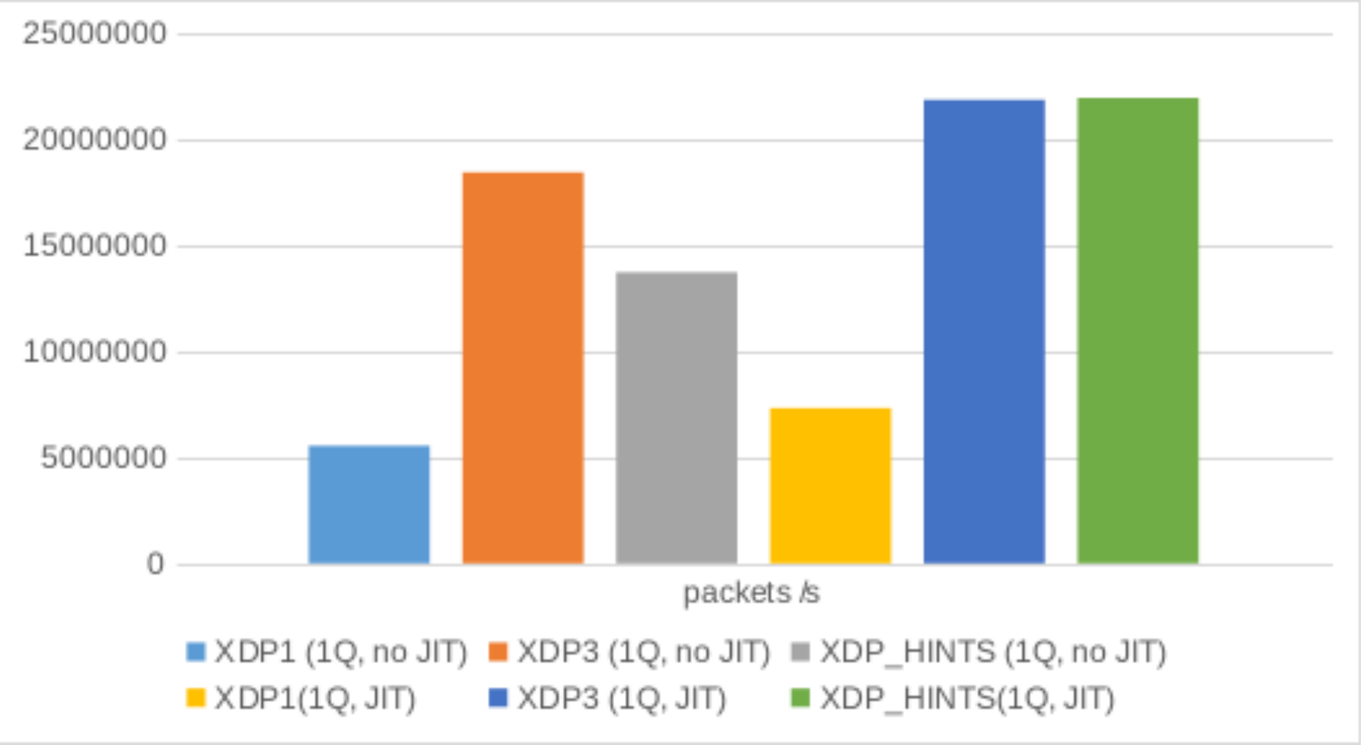
Network device: XXV710, 25GbE NIC, driver version 2.1.14-k

Configuration: Single Rx queue, pinned interrupt

XDP3: Zero packet parsing (best case scenario)

XDP_HINTS: Uses ptype provided by driver, no packet parsing

# Performance improvements, cont.



# Performance improvements, next steps

- Continued testing on newer Xeon systems
  - Try to observe any DDIO improvements
- Try minimizing memcpy()'s into XDP buffer headroom
  - At least measure impact of memcpy() versus direct DMA
- Test with larger, more complex XDP programs
  - Test with encap/decap, encryption, forwarding, etc.

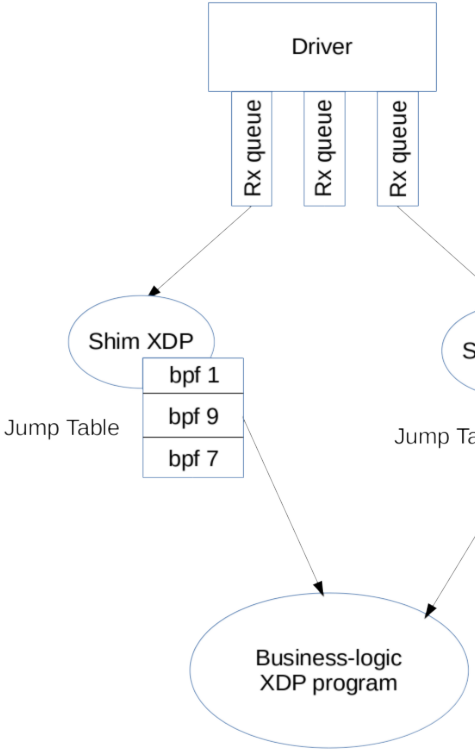
# Metadata layouts – what to do?

- Approach 1: Common layout independent of underlying HW
  - Requires community agreement on common structures
  - Would be in the UAPI

- Approach 2: Vendor libraries in eBPF libraries
  - Requires XDP/eBPF programs to detect underlying hardware

- Approach 3: Chained XDP programs
  - Lightweight “shim” would contain vendor-specific logic
  - Tail-call larger program with parsed metadata to run rest of logic

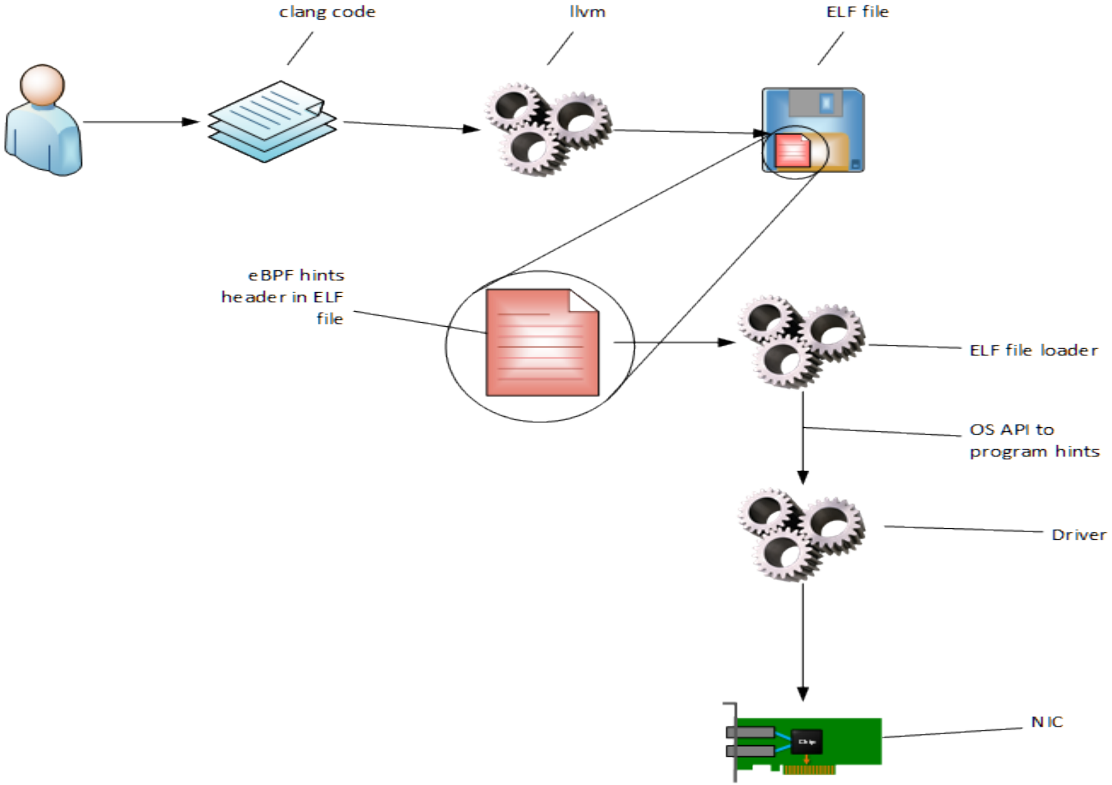
# Chaining XDP programs



# Programming HW hints

- Existing hardware flow programming available via tc
  - tc flower
  - tc u32
- Difficult to match filters programmed via tc and which HW hints to use in XDP programs
- Any new match actions and/or fields for tc flower need kernel changes to implement
- Defining HW hints to program via eBPF sections can be dynamic and not require kernel changes to extend

# eBPF hint programming flow



# HW Hints

Parsing Hints

| Type of HW hint | Size | Description |
|---|---|---|
| Packet Type | U16 | A unique numeric value that identifies an ordered chain of headers that were discovered by the HW in a given packet. |
| Header offset | U16 | Location of the start of a particular header in a given packet. Example start of innermost L3 header. |
| Extracted Field value | variable | Example Inner most IPv6 address |

Map Offload

| | | |
|---|---|---|
| Match | U32 | Match a packet on certain fields and the values,  provide a SW marker as a hint if the packet matches the rule |

Packet Processing Hints

| | | |
|---|---|---|
| Checksum | U32 | A total packet Checksum |
| Packet Hash | U32 | Hash value calculated over specified fields and a given key for a given packet type |
| Ingress Timestamp | U64 | Packet timestamp as it arrives |

# ELF Special Headers to request HW hints

```
struct bpf_hw_hints_def SEC("hw hints") rx_offset = {
        .type = PACKET_OFFSET_INNER_L4,
        .size = sizeof(__u16),
        };
```

```
struct bpf_hw_hints_def SEC("hw hints") rx_ptype = {
        .type = PTYPE,
        .size = sizeof(__u16),
        };  /* PTYPE values should be agreed upon between the SW and
the HW providing the hints, the driver may have to do the translation
between the two */
```

```
struct bpf_hw_hints_def SEC("hw hints") rx_match = {
        .type = PACKET_MATCH,
        .fields = {PTYPE, INNER_L3_SRC, INNER_L4_SRC},
        .mask = { 0xff, 0.0.ff.ff, 0xffff},
        .value = { 0x10,  10.10.20.2,  65},
        .result = 25 /* This hints adds a match rule into Hw, which creates a SW defined result when Hw
finds a match */
        .size = sizeof(__u32),
        };
```

# Programming Flow

- The ELF sections that carry Hw programming hints need to be passed over to the driver in some form so that it can program the HW accordingly.
- Introduce some new helper ndo_offload_xdp_hints() that the driver can call to extract what the XDP program can use as hints and program the HW accordingly.
- The driver hides all the HW programming details, the hints format is generic for any HW.
- A given HW may or may not be able to provide all the hints.
- It’s a best effort mechanism to offload what the HW can support.

# Wrap-up, next steps

- Performance results using HW hints are promising
  - Still need to test on newer hardware
  - Still need to test with more complex XDP programs
- Prototyping of eBPF-based HW hint programming needs to be completed
  - Will provide RFC patches to community to bless direction
- Need feedback from community on how to make ready for merging
  - Need agreement on actual metadata layout in xdp_buff headroom
  - Need agreement if eBPF-based HW hint programming is right direction

# Questions?

intel®

# Backup

# Using HW hints: motivation

- Existing NIC hardware has packet processing capabilities and can provide this data in some form to the software.

- SmartNICs and programmable NICs will have capabilities to provide even more packet meta data to software

- Why not utilize these hints from the NIC Hardware already available as meta data to the NIC SW driver and make it available to XDP?

- Utilizing these hints will help the XDP programs to accelerate packet processing and take faster decisions based on the business logic for a given use-case