XDP Infrastructure Development

"Internal" NetConf presentation

Jesper Dangaard Brouer
Principal Engineer, Red Hat

Date: October 2016
Venue: NetConf, Tokyo, Japan

Introduction

- This presentation is only relevant to
  - Infrastructure developers of XDP

Speaking bluntly about XDP

- Basically a driver RX-code-path benchmark tool
  - eBPF, only thing that makes it usable for real use-cases
    - (DDoS use-case is very real!)
- XDP focus: solving driver RX bottleneck
  - E.g: Mlx5 driver, RX drop inside driver (single CPU)
    - 6.3Mpps at NetDev 1.1 (Feb 2016)
    - 12.0Mpps Jesper's PoC hacks
    - 16.5Mpps with XDP and changed MM-model (net-next 86994156c73)
      - (no-cache prefetch, more optimizations coming, expect 23Mpps)

XDP is motivation for NIC vendors

- XDP is motivating drivers developers to:
  - Change memory model to writable-pages
  - Force later allocations of SKBs in drivers
  - Fix RX bottleneck in drivers

Secret to XDP performance(1): mem

- Current XDP features **secret to performance**:
- They avoid calling memory layer
  - Local driver page recycle tricks
  - Upcoming multi-port TX
    - Cannot hide behind local driver recycling
    - Need more generic solution (like page_pool proposal)

Secret to XDP performance(2): I-cache

- XDP *benchmarks* does not stress I-cache
  - Hiding behind:
    - Very small benchmark bpf programs
    - Bench does not show intermixed traffic
- Once XDP programs get bigger
  - Running into I-cache misses
    - eBPF progs with tail calls
- Solution: Work on packet-array vector
  - Don’t intermix traffic XDP/netstack traffic

XDP missing documentation

- Attempt to doc XDP project: https://prototype-kernel.readthedocs.io
  - Not the official doc:
    - Need to be accepted (and reviewed) into kernel tree
- See documentation as a collaboration tool
  - I'm not controlling project, just the secretary
  - Capture design spec
    - from summary of upstream discussions
- Basic requirements link

XDP userspace interface

- Bad semantics:
  - Blind load and override current XDP program
  - Leads to hard-to-debug issues for userspace
- XDP only query option is "bool"
  - Userspace don't know who's xdp_prog is running.
- Tools: like tc and iptables
  - Allow root to override/del
    - but have visibility to view current state
    - tc even have add/del/change/replace semantics

Improving XDP userspace interface?

- Programs can netlink monitor for "link" changes
  - Curr issue: replace will just show a xdp "true"
- Simple solution: static global ID counter
  - Attaching xdp_prog inc and return as id
  - netlink update contains ID
    - Give ability to identify/debug issues
- Could add/del/change/replace semantics be useful?
  - Acronym CRUD (Create, Read, Update and Delete)

XDP features and versions?

- How to handle: Capabilities negotiation?
  - Both driver and userspace tool need to
    - have concept of features/capabilities
- How do we handle adding new features?
  - and new versions of features?
- Current upstream solution assume:
  - that XDP_DROP and XDP_TX is always supported
  - XDP_DROP could be useful separately
    - and significantly easier to implement (e.g. e1000)
  - XDP_TX difficult on HW with a single TXq
    - Mess with netstack interaction e.g. BQL and fairness

Missing: push/pop headers

- Implementation is missing ability to:
  - Modify packet length
    - useful for push/pop of headers (tunnel, VLAN etc.)
  - Need to know/define headroom size
    - Simple option use: NET_SKB_PAD
      - but what about XDP prog portability?
        - Cannot see HEADROOM as "input" to XDP anywhere?
        - (likely want this compile time for eBPF)

Other API issues

- VLAN issue, only discussed, never implemented
  - AFAIKR VLAN ids should always be inlined in packet
    - Implies disabling HW features, when loading XDP
      - (Hint: not implemented…)

XDP prog per RX-queue

- Why a XDP program per RX-queue
  - Flexibility, do not monopolize entire NIC
- Performance issue:
  - XDP change memory model of RX queue
    - packet per page (trading memory for speed).
  - Cause perf regressions, for normal stack delivery
    - (1) bottleneck in page allocator (fixable via page_pool)
    - (2) skb→truesize increase, affecting TCP window
  - Thus, limit XDP to some RX queues, allow
    - Only affect traffic that really needs XDP

XDP: HW provided protocol offsets

- Most HW provide protocol offset in descriptor
  - E.g. Willy Tarreau "NDIV" solution use it
  - Pass useful L2/L3/L4 offsets to application
    - save it from parsing packets
    - (Presented in 2014 link)
- Would it be useful for XDP?
  - Find most efficient way to pass info to eBPF

XDP: Multi port forwarding

- Need help designing this!?

- Proposal: XDP port abstraction
  - Using ifindex is limiting and Linux centric
  - Port index table, allow port types to be intermixed
    - concept rather simple:
      - provide ingress port, return egress index
    - do we need to express more?
  - Extend with broadcast to all index’es in group
    - L2 bridge need a “flood” operation

XDP: Network Device Operation for "raw" xmit

- For multi-port TX
  - net_device extend with NDO for "raw" page transmit
    - Please: Bulking from day-0
      - Even if solving: lockless access to remote queue
  - Exposing TX queue number or not?
    - Likely best to hide TXq behind API
  - vhost_net
    - Can we “raw” transmit to a guest OS?
      - V1: Copy packet
      - V2: RX Zero-copy via dedicated RXq + page_pool

XDP: Generic hook?

- Generic XDP hook
  - Only if performance is good enough
  - IMHO: Should allow kernel itself to use XDP
- Who want to use a generic hook?
  - Nf-tables?

End of slide show

- Did I miss something?
- Any other XDP related topics?