eBPF & Switch Abstractions

Nick Viljoen <nick.viljoen@netronome.com>
Contents

- Background
- The Multi-Host NIC Abstraction
- The Switchdev Based Multi-Host NIC Abstraction
- Currently upstream (as of 2 weeks ago)
  - Boot
  - Setting Switchdev Mode
  - Loading Qdiscs
- Next Steps (currently being upstreamed)
  - Generalising Qdisc Offload
  - Adding clsact Qdiscs (u32, cls_bpf)
- Future Work
Background: HW, BPF JIT, NIC as a Switch
Background: HW

- Many core, fully programmable network processor
  - 48-96 preprocessing cores
  - 54-120 programmable cores, 8 threads per core, MIMT
  - up to 4 PCIe
  - Up to 40 ports supported
  - ~17MB of on chip memory
  - 2-24GB of DRAM
  - Distributed & Transactional memory architecture

- Low power
  - ~ 10-35W (dependent on chip + frequency)
Background: HW Datapath

Processing Islands

- **PCIe**
- **Phy Ports**
- **Pre processing**
- **CTM**
- **CLS**
- **Reorder**
- **Accelerators** (C, P)
- **IMU/EMU** (LU, AT)

Yellow square = Flow Processing Core (FPC)
Background: eBPF Offload Architecture

- Program is written in standard manner
- LLVM compiled as normal
- The nfp’s jit is called like any other architectures jit
- This converts the BPF bytecode to NFP machine code
- Translation reuses the verifier infrastructure in kernel
- Defining the FPC datapath code using BPF
Background: NIC as a Switch

Multi-Host

```
H

PHY/MAC

H
```

“Multi-Homed”

```
Host

PHY/MAC

PHY/MAC

PHY/MAC

PHY/MAC
```
Multi-Host NIC Architecture
Multi-Host NIC Abstraction: Today

Many great things, but a few holes

No Visibility for other hosts

No Linux Controlled QoS

No concept of where offloads occur in datapath, e.g. XDP offload

© 2018 NETRONOME SYSTEMS, INC.
Multi-Host NIC Abstraction: Switchdev Based

Host 0

Host 1

Host 2

Host 3

L2 Switch

Attach point for QoS

Configurable from host 0

Potentially Visible Stats

Clear concept of all the logical ports present

PHY/MAC
Current Work: Switchdev and Simple Qdisc Offload
Multi-Host NIC Abstraction: Incorporating Qdiscs

Using offloaded qdiscs for QoS allows improved throughputs and latency.
3 Key Steps

- Initialisation (.probe)
  - App Initialisation
  - vNIC Allocation
- Entering Switchdev Mode (.devlink_eswitch_set)
  - Spawning Representors
- Qdisc Setup (.ndo_tc_setup)
  - Attaching Qdisc to struct nfp_abm_link
Initialisation

Kernel
- ` pci_epf_core.c`
- `nfp_net_main.c`
- `nfp_abm_init`

Driver
- `nfp_net_pci_probe()`
- `nfp_mbox_cmd()`
- `nfp_net_pf_app_init`
- `nfp_net_pf_alloc_vnics`
- `nfp_abm_ctrl_qm_disable`
- `nfp_abm_ctrl_read_params`

App Abstraction
- `main.c`
- `nfp_abm_init`
- `struct nfp_abm`
- `nfp_abm_vnic_alloc`
- `struct nfp_abm_link`

- `ctrl.c`
- `nfp_abm_ctrl_qm_disable`
- `nfp_abm_ctrl_read_params`

- `app->type->init`
- `app->type->vnic_init`

- `probe`
- `nfp_net_pci_probe()`
Entering Switchdev Mode

Devlink
Devlink dev eswitch show
devlink dev eswitch set
devlink sb pool show
devlink sb pool set

devlink.c

nfp_devlink.c

nfp_shared_buf.c

App Abstraction

ctrl.c

main.c

nfp_abm_eswitch_mode_get
nfp_abm.spawn_repr
nfp_abm_ctrl igen_enable
nfp_mbox_cmd()

nfp_app.c

nfp_shared_buf pool_get
nfp_shared_buf pool_set

User Space

Kernel

Driver

© 2018 NETRONOME SYSTEMS, INC.
Qdisc Offload

User Space

Kernel

Driver

App

Abstraction

main.c

ctrl.c

nfp_app.c

app->type->setup_tc

nfp_abm_setup_tc_red

nfp_abm_setup_tc_mq

struct nfp_qdisc_red

struct nfp_abm

struct nfp_abm_link

struct nfp_repr

nfp_mbox_cmd()
Next Steps: Extending the Egress Representor Architecture
Items Covered

- Generalising Qdisc Offload
  - Structure changes
    - nfp_abm_link
    - nfp_qdisc
- The clsact Qdisc
  - Motivation
  - Architecture
Generalising Qdisc Offload: Structure Changes

Before

```
struct nfp_abm_link {
    struct nfp_abm *abm;
    struct nfp_net *vnic;
    unsigned int id;
    unsigned int queue_base;
    unsigned int total_queues;
    u32 parent;
    unsigned int num_qdiscs;
    struct nfp_red_qdisc *qdiscs;
};
```

```
struct nfp_red_qdisc {
    u32 handle;
    struct nfp_alink_stats stats;
    struct nfp_alink_xstats xstats;
};
```
Generalising Qdisc Offload: Structure Changes

After

```c
struct nfp_abm_link {
    struct nfp_abm *abm;
    struct nfp_net *vnic;
    unsigned int id;
    unsigned int queue_base;
    unsigned int total_queues;
    struct nfp_qdisc *root_qdisc;
    struct radix_tree_root qdiscs;
};

struct nfp_qdisc {
    struct net_device *netdev;
    enum nfp_qdisc_type type;
    /***/
    struct nfp_qdisc **children;
    /***/

    union {
        /* TC_SETUP_QDISC_Prio */
        struct {
            const u32 *map;
        } prio;
        /* TC_SETUP_QDISC_RED */
        struct {
            u32 threshold;
            struct nfp_alink_xstats stats;
            struct nfp_alink_xstats prev_stats;
        } red;
    }
};
```
clsact Qdisc: Motivation

clsact ensures ability to use GRED

GRED allows more granular QoS
clsact Qdisc: Architecture

User Space

- tc filter
  - tc filter add (u32)
  - tc filter replace
  - tc filter del
  - tc filter show

- sch_9qdisc
  - 9qdisc9add9(mq/red)
  - 9qdisc9replace
  - 9qdisc9del
  - 9qdisc9show

Kernel

- tc qdisc
  - tc qdisc add (mq/red)
  - tc qdisc replace
  - tc qdisc del
  - tc qdisc show
c

Driver

- nfp_app.c

  - main.c
  - ctrl.c
  
  - nfp_mbox_cmd()

  - struct nfp_abm
  - struct nfp_abm_link
  - struct nfp_repr
  - struct list_head dscp_map

  - cls_u32.c
  - sch_9ingress.c/sch_9red.c/sch_9mq.c

© 2018 NETRONOME SYSTEMS, INC.
Future Work: Multi-host BPF Offload
Items Covered

- Firmware and BPF JIT
  - Flow Processing Core Datapath
- cls_bpf and Switchdev
  - Architecture
- XDP for Multihost Systems
  - Problems
  - Proposed Abstraction
Firmware and JIT Changes

- Preclassifiers used to isolate flow processing cores per host
- Lookup in jump table based on entry port
- Returns the number of programs to jump to and their locations

Flow Processing Core
- eBPF Progs
- Dynamically Loaded Helpers

Preclassifier

CLS (Mem)
- Port
- Progs

Reorder

Dynamically Loaded Helpers

Tail calls
XDP on Multi-Host systems

- Challenges that have to be solved
  - XDP is an RX exclusive hook
  - Heterogenous architecture support is nascent
  - Security
- However more and more of the potential problems are falling away
  - e.g David Ahern’s recent work on exposing the FIB table
XDP on Multi-Host systems: Proposed Abstraction

XDP attached to ingress at each port

redirect() + FIB table access allows conventional + unconventional switching
Summary

- Proposing a fully flexible datapath for a multi-host NIC
- Achieved through a combination of switchdev, qdisc offload, cls_bpf and XDP
- Work in progress
  - Switchdev architecture and qdisc offload has been upstreamed
  - Next is simple clsact support
  - Followed by cls_bpf & XDP
- Provides potential for BPF defined pipelines in heterogenous architectures
  - See Jakub’s talk at the microconference for more!
Credit (Team)

- Jakub Kicinski
- Jiong Wang
- Quentin Monnet
- David Beckett
- Edwin Peer
- Johan Moraal
- Mary Pham
Thank you!

Discussion

Questions/Comments