p4-tc Workshop
A new traffic classifier for Linux

Netdev 0x16

p4-tc Workshop
A new traffic classifier for Linux

Workshop Agenda (~ 2.5 hrs)

- Test Framework (45 mins)
- Kernel Code Walk (30 mins)
- Introspection (10 mins)
- Compiler Support (10 mins)
- Driver Interface (40 mins)
- Other topics: Programmable parsers (15 mins)
- Conclusion 5-10 mins

How to Contribute to p4 tc

- Mailing list

https://lists.netdevconf.info/cgi-bin/mailman/listinfo/p4tc-discussions

- Github

Opensource working Group : Meets every 2 weeks

WG Notes (add link)

Status

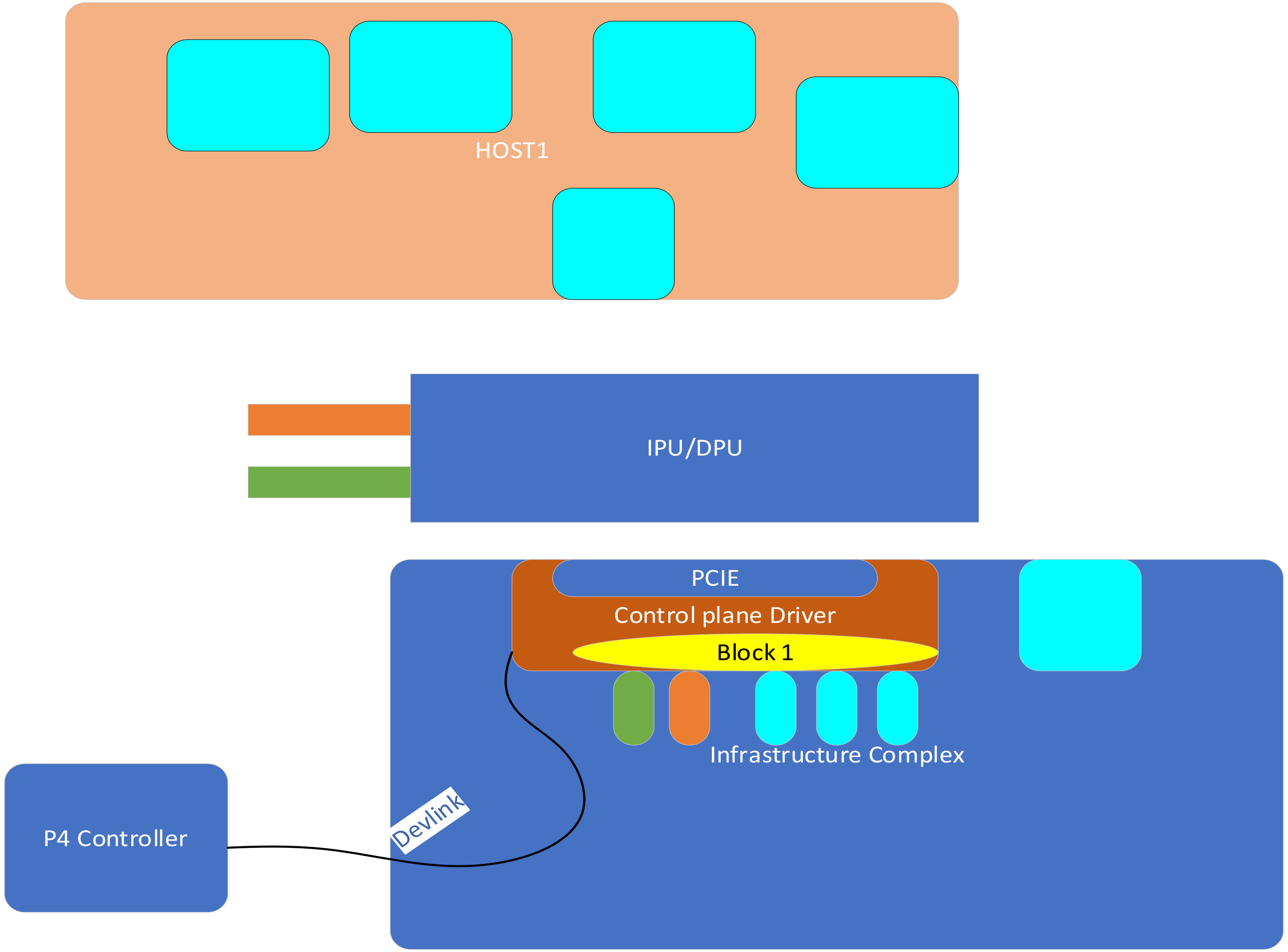
- Progress so far
  - SW model
  - Test framework
  - Compiler backend for generating p4tc scripts
  - Introspection
- Not started yet
  - Driver and offload hooks
  - Some opens on the parser

Driver Interfaces (p4 tc)

Anjali Singhai Jain

October 2022

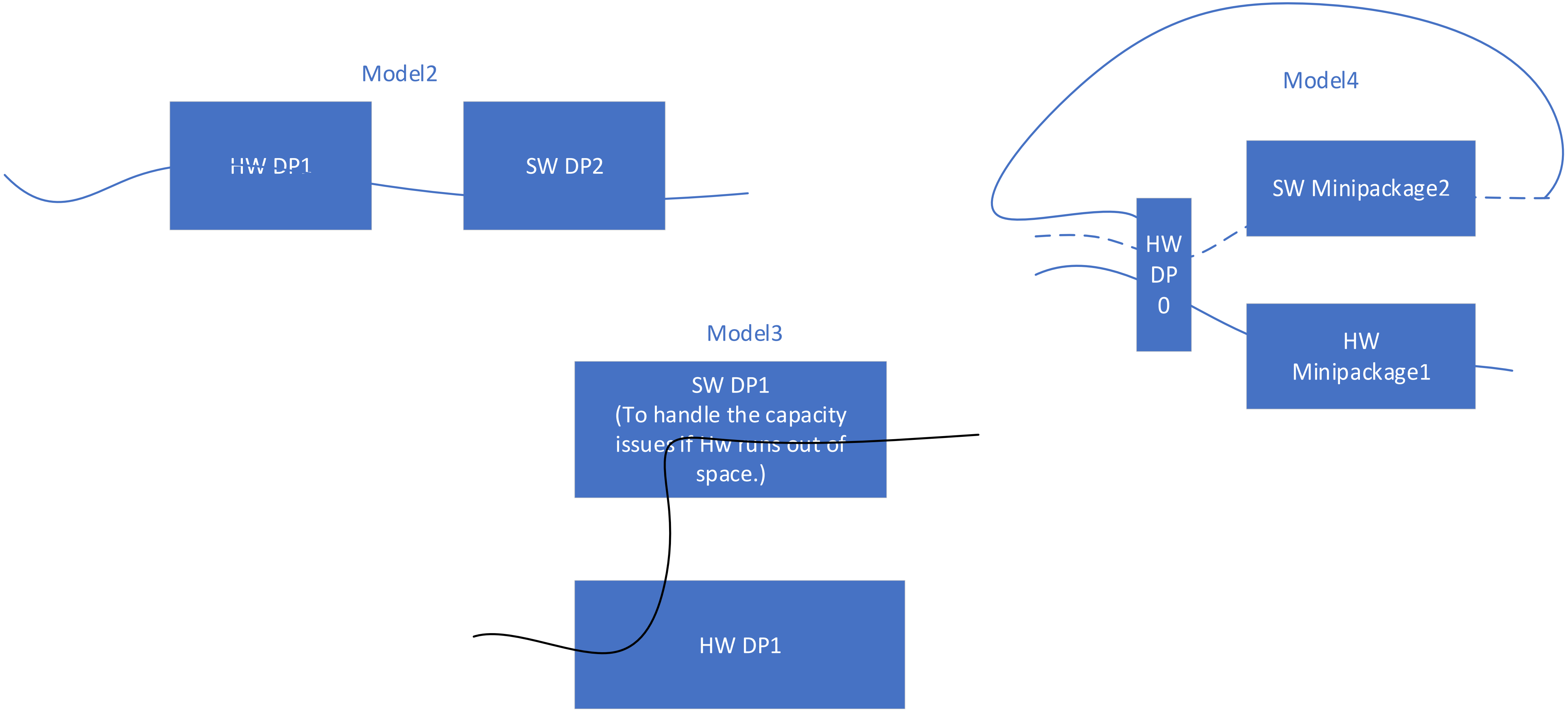
IPU/DPU Control Plane Topology



Use of SW and HW Dataplane (WIP)

- SW is used for emulating the HW Dataplane and is standalone to make the Ecosystem ready before HW shows up.
- SW is your fall back for anything that does not fit in HW, or SW is an extension for HW Dataplane. (Table ran out of capacity)
- Split the pipeline in HW to HW and SW flows (parser in HW decides)
  - Example HW does not handle fragmented packets
- A single packet gets processed in HW first and then in SW

Different SW/HW offload models



Model2

HW DP1

SW DP2

Model4

SW Minipackage2

HW DP 0

HW Minipackage1

Model3

SW DP1 (To handle the capacity issues if Hw runs out of space.)

HW DP1

Two modes of programming

- Slow path in SW on the Infrastructure complex.
  - Rule is programmed in reaction to the first packet of the flow missing in SW
  - Rule is programmed on the representor from where the packet was reported.
    - Identifies the P4 table, the key value, mask and action (action index or immediate action with data)
- No slow path in SW
  - Rule is programmed proactively for the policies etc
  - Since the rule is programmed in a table , it can apply to many packets from different source of packet

Example Flow:

- Control plane driver loads, creates switchdev device, port representors for external ports. Also the driver creates a devlink hook. Driver registers a callback for block creation/deletion
- Administrator Creates an ingress block using tc commands : Block1

(optional to create an egress block as well.) and adds a minimum of one netdev) (driver is notified of block creation.)

- Adds the rest of the representors to the ingress block1

- Install the p4 tc templates for SW
- Bind the p4 program in SW to tc block1

  ($ tc filter add block 1 ingress protocol any prio 1 p4 pname myprogram)
- myprogram will get tied to pcie device now.
- Remote P4 controller downloads the P4 package using devlink attached to the on the box.
- tc p4 create table entry
- the kernel will find a pcie device that is tied to block1

  (may be there are multiple programs tied to block1.)
- In essence once it finds the pcie device, it makes the ndo ops for adding filter rules (this could be any netdev for the device.)
- The driver gets the following info when a rule is added:
  - 1. The P4 program ID   2. table ID   3. Field ID, mask, value tuple 3.a priority   4. action ID and action data/index

    (Incase of index, action has to be pre-created.)