



Match Interface

OVS Fall Conference 2015

Hao Zheng, Intel

Legal Disclaimer

General Disclaimer:

© Copyright 2015 Intel Corporation. All rights reserved. Intel, the Intel logo, Intel Inside, the Intel Inside logo, Intel. Experience What's Inside are trademarks of Intel. Corporation in the U.S. and/or other countries. *Other names and brands may be claimed as the property of others.

Technology Disclaimer:

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at [intel.com].

Performance Disclaimers (include only the relevant ones):

Cost reduction scenarios described are intended as examples of how a given Intel- based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

Agenda



- Match Interface Overview
- Usage Scenarios
- Usage Models
- Example
- OVS Touch Points

- Summary



Match Interface

Generic Match/Action Pipeline Runtime:

- 'Match': Match conditions that this pipeline matches on
- 'Action': Actions that the pipeline can take. Exposed as black box functions
- 'Table': Tables that structure the pipeline, and their interconnections
- 'Config': Configuration of the pipeline (device, port and table attributes)
- 'Headers': Header structures to concisely describe packet protocols

Pipelines Can Be Defined in a Language (e.g. P4)

Match Interface

`get_headers(...)`

`get_header_graph(...)`

`get_tables(...)`

`get_table_graph(...)`

`get_actions(...)`

`create_table(...)`

`destroy_table(...)`

`set_entry(...)`

`get_entry(...)`

`clear_entry(...)`

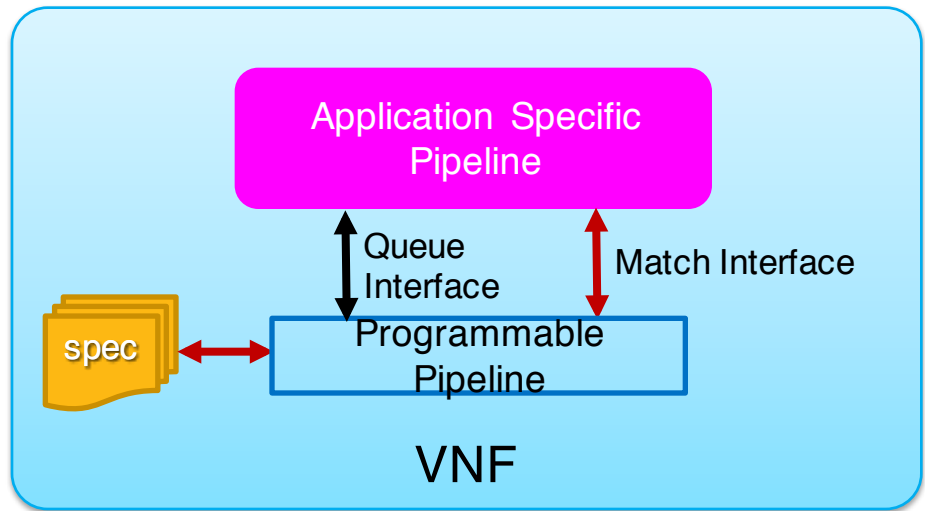
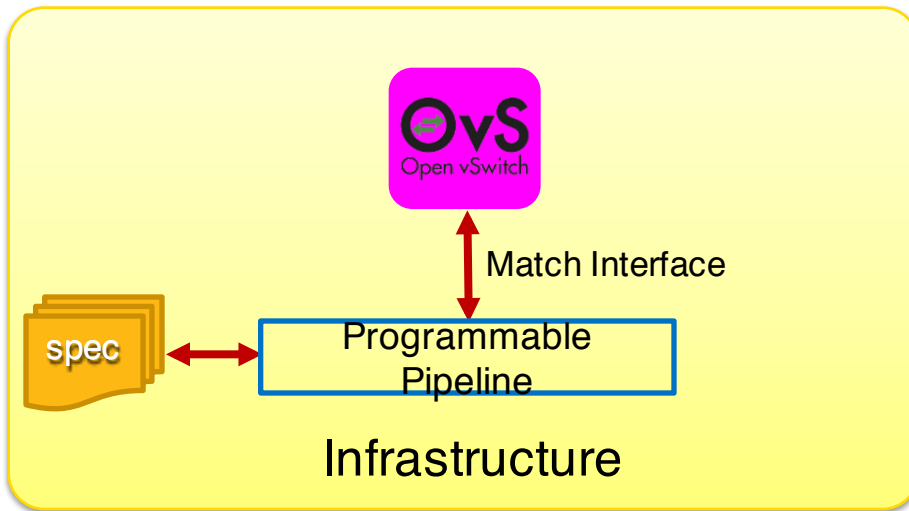
Usage Flow:

- Detect Capabilities: Get Header & Table Graph
- foreach table: Get params (Matches, Actions, Size)
- foreach action: Get parameters (action signature)
- Create Additional Tables (if needed)

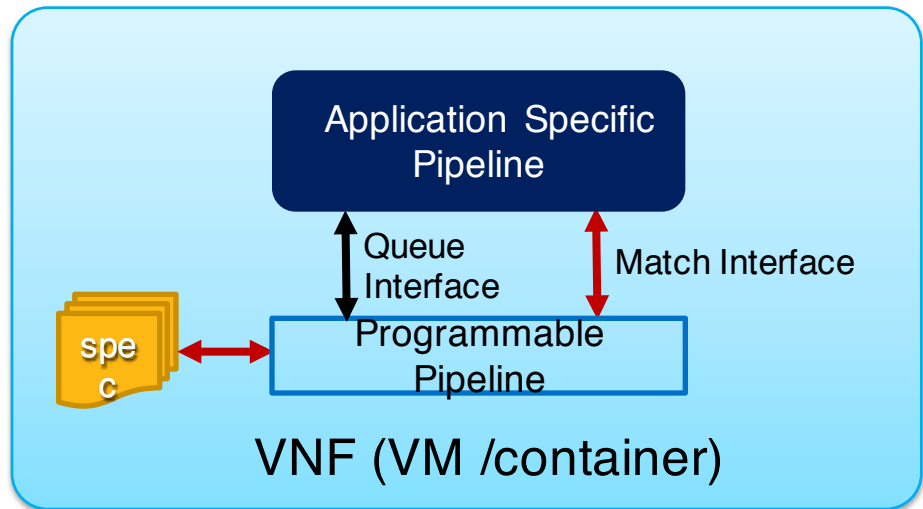
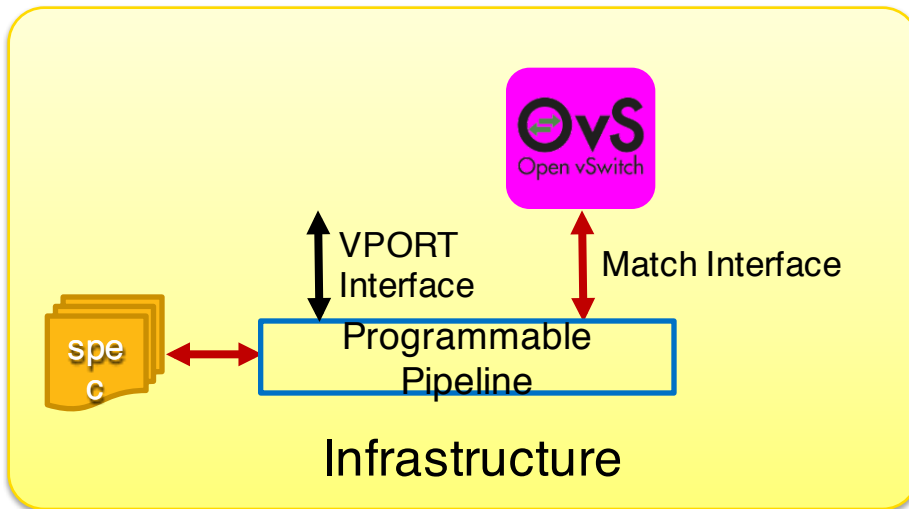
Populate With Entries:

- Set Entry (Add a Match-Action Rule)
- Get Entry (Read back stats & state)
- Clear Entry (Remove Match-Action Rule)

Match Usage Scenarios



Match Usage Models



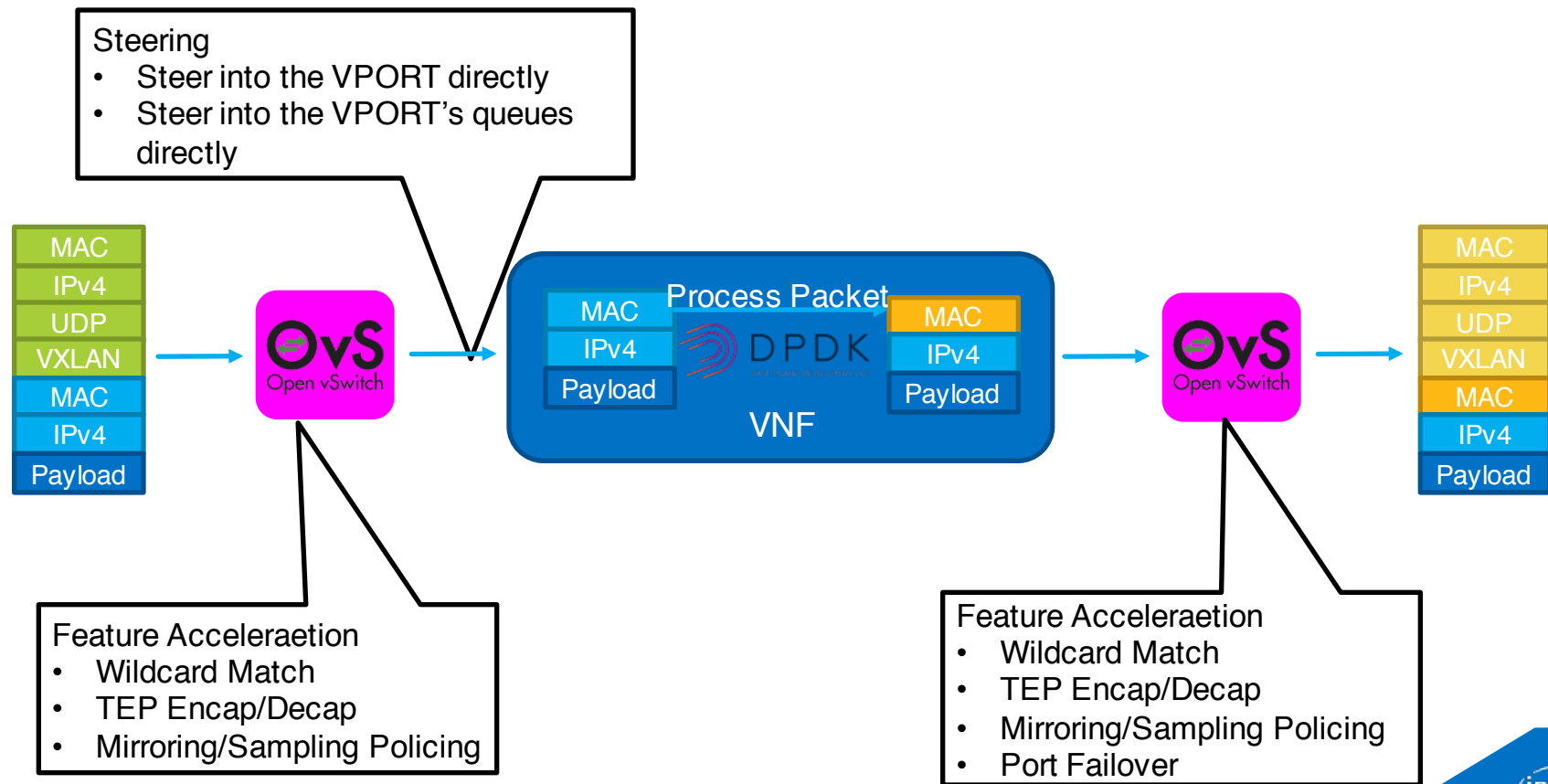
Feature Acceleration

Feature Acceleration

Pipeline Steering to/from VPORTs

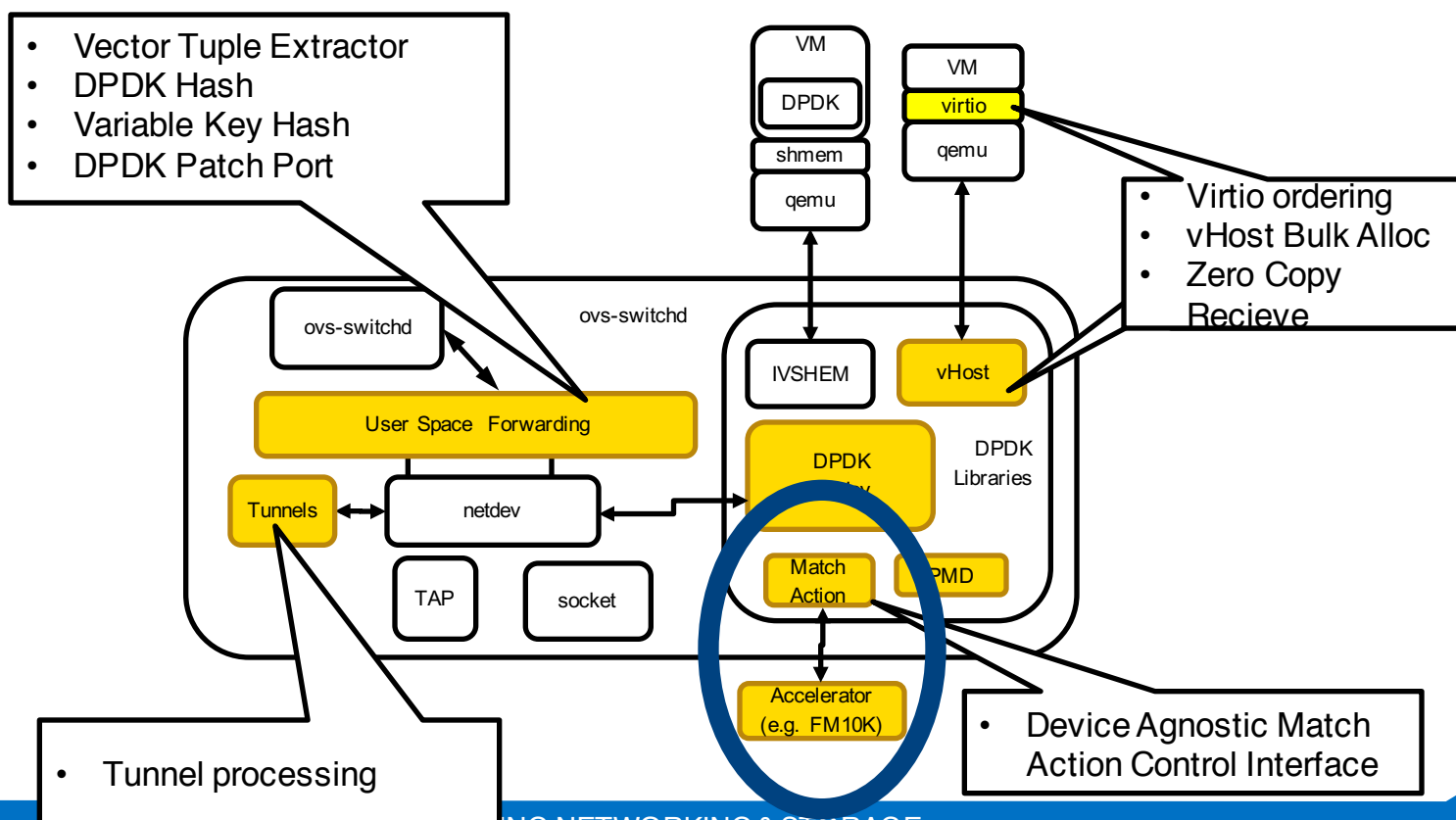
Pipeline Steering to/from Queues

Example

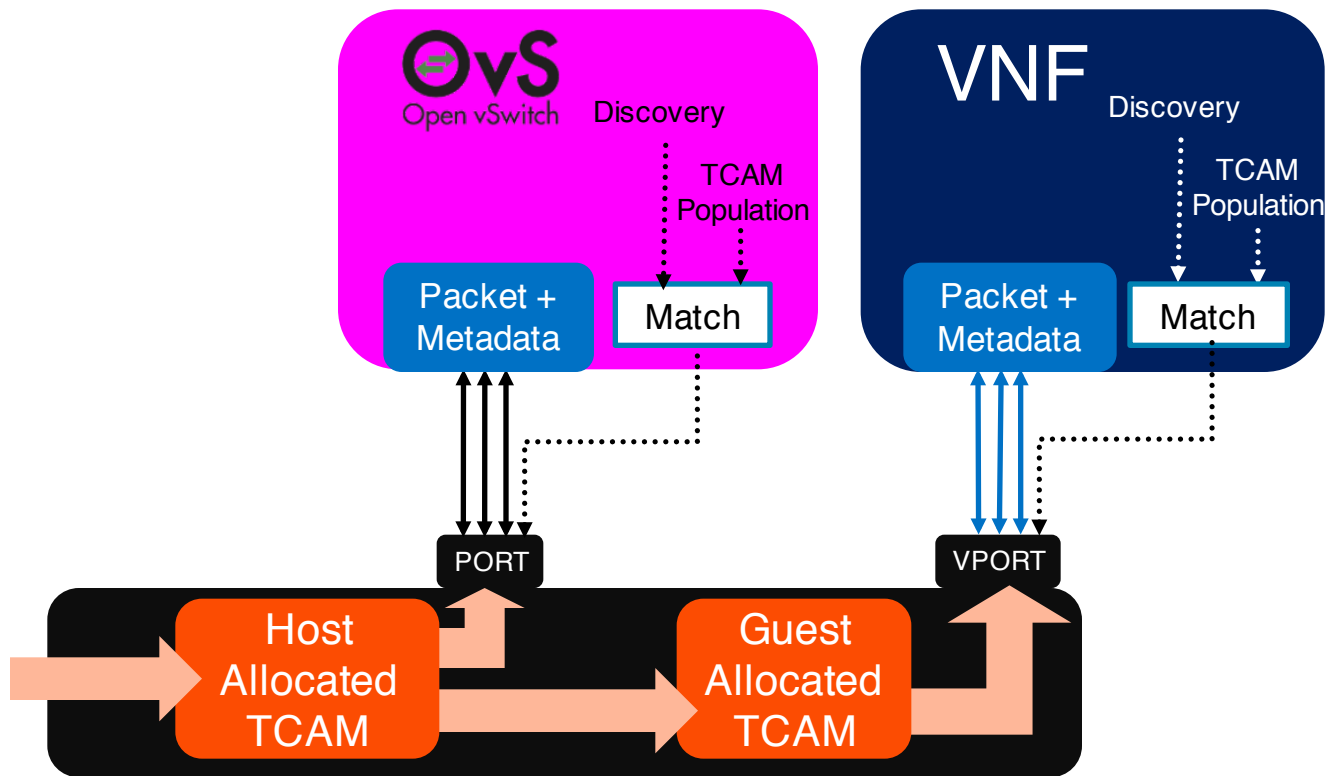


OpenvSwitch 2.x DPDK 2.x

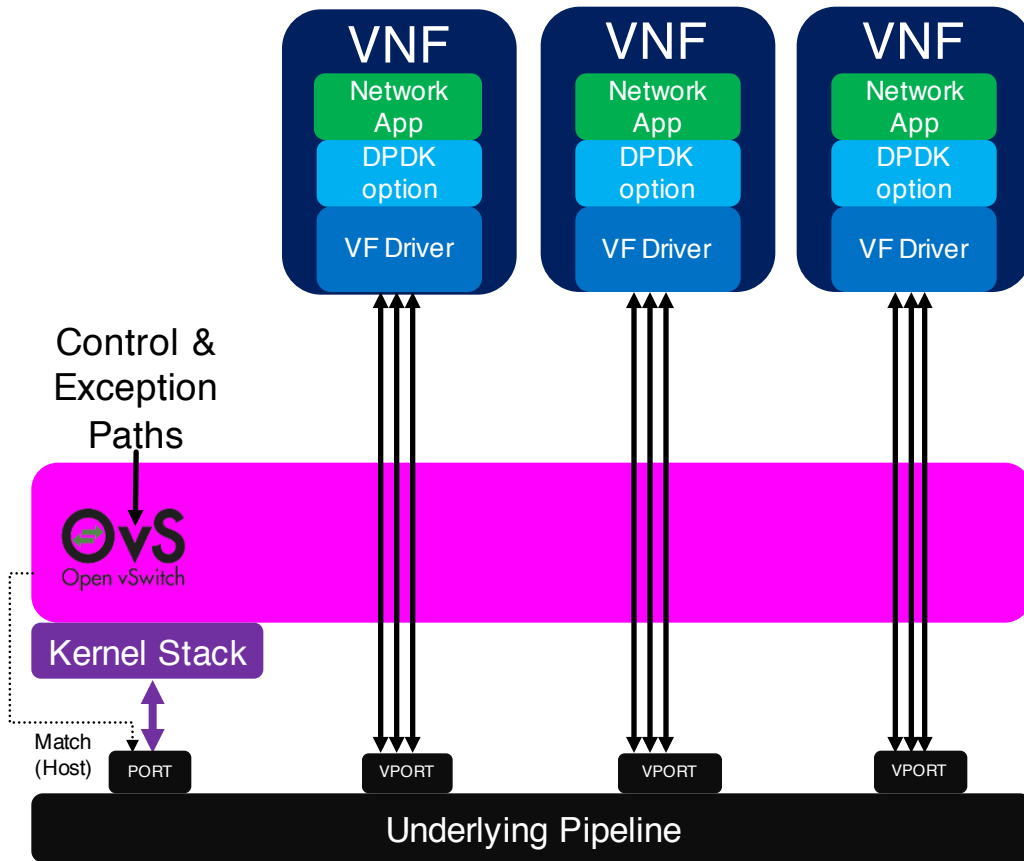
netdev-DPDK Performance Enhancements



Feature Acceleration: Pre-Classify TCAM



Pipeline Steering to/from VPORs



OVS Control Plane Configures Underlying Pipeline to Steer Packets in/out of VPORs

- Policies for copying rules into hardware
- Reading hardware state to keep software in sync
- Aging of rules
- Exceptions sent back into OVS

Example: OVS Controlled SR-IOV

Pipeline Steering to/from VPORs



```
[root@localhost ~]# ovs-vsctl show
5e34826d-1362-449e-9b48-27769288915a
  Bridge "br1"
    Port "br1"
      Interface "br1"
        type: internal
    Port "vf1"
      Interface "vf1"
        type: vf
        options: {pci="05:00.2", root="ens785"}
    Port vxlan
      Interface vxlan
        type: vxlan
        options: {key="100", remote_ip="60.0.0.1"}
  ...
```

Requirements for Pipeline Steering:

1. Underlying Pipeline applies all OVS rules on VNF's traffic
 - Exception path is back into OVS
2. Underlying Pipeline needs a direct connection to VPOR
 - So it can steer directly
3. OVS needs to represent an Underlying Pipeline's VPORs

Rule Copy Policy



Today: Supporting Rule Sets Generated by OpenStack and/or OpenDaylight

1. VM/container identified as being connected to underlying pipeline VPORT
2. OVS rules to/from that VPORT are pushed to underlying pipeline
3. If *any* of the applied functions for a particular flow cannot be supported, then
 - Revert to handling this flow within OVS
 - OVS receives the flow, and is able to forward traffic to/from any VPORT

Match Summary



Generic Match/Action Pipeline Runtime

- Pipelines can be built with a language (e.g. P4)
- Discovery Interface to Detecting the Pipeline
- Rule entry interface for populating tables

Looking to Upstream the Interface into Linux

- Prototype implementation can be found below

<https://github.com/match-interface>