P4 Compliant Control Plane Driver for Linux Kernel

Neha Singh, Intel Corporation
James Choi, Intel Corporation
Weiqian Dai, Intel Corporation

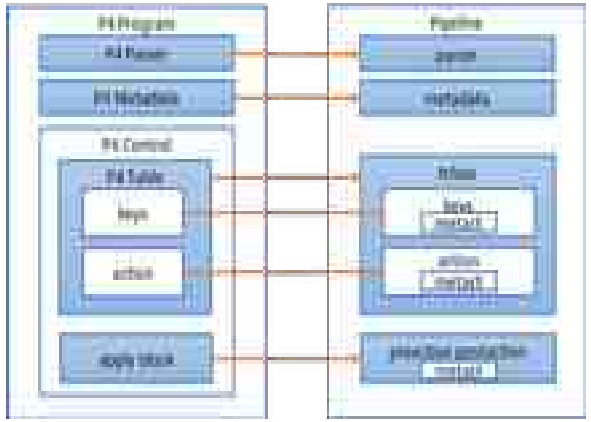
Motivation

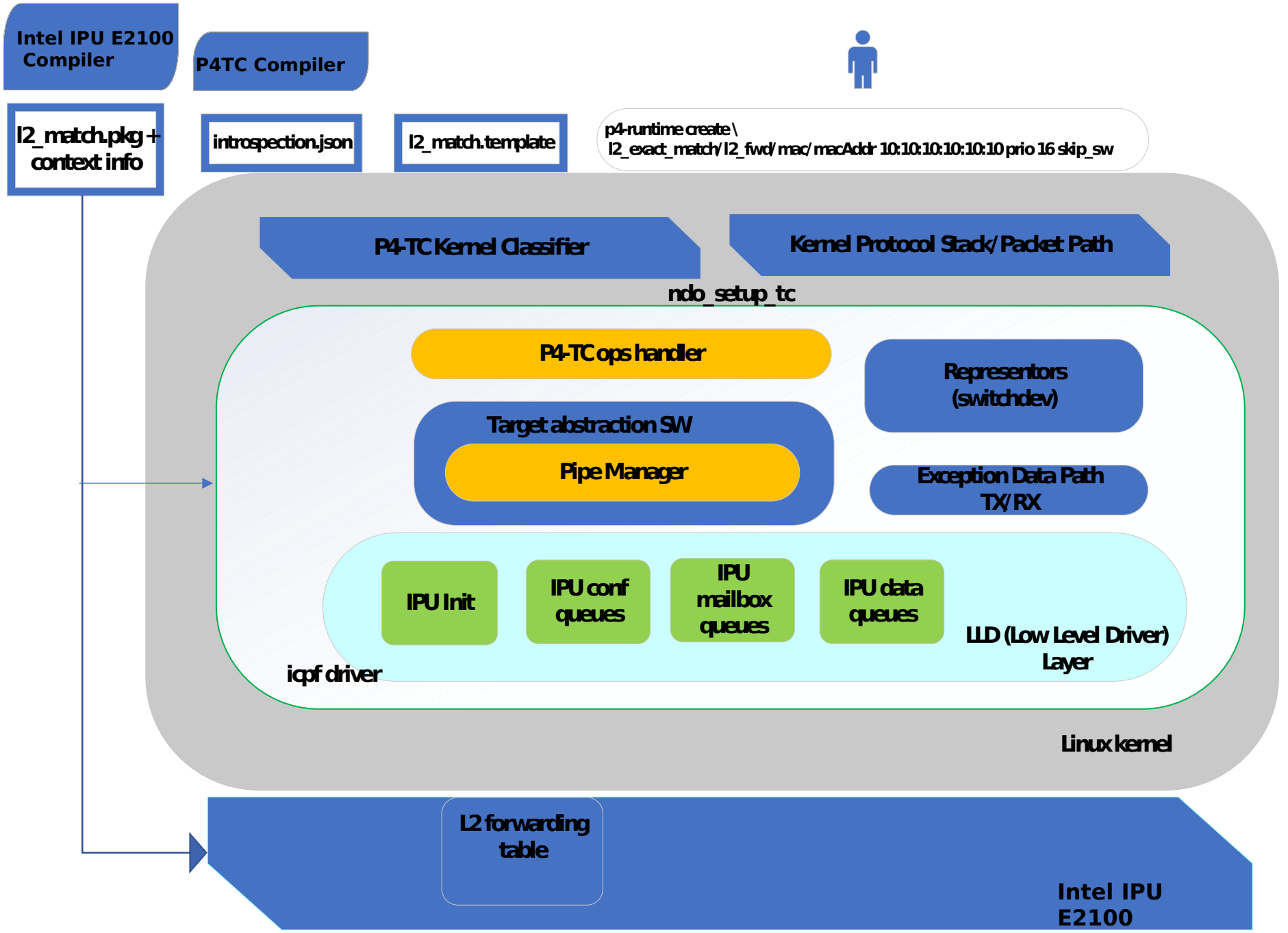
- Performance
- Save Host CPU
- Low latency
- P4 programmability for target device in kernel space.

Driver Architecture Overview

P4 Program:

```
action send(PortId_t vport) {
        send_to_port(vport);
    }
  action drop() {
        drop_packet();
    }

    /*
     * L2 forwarding table based on
destination MAC.
     */
    table l2_fwd {
        key = {
            hdr.ethernet.dstAddr:
exact;
        }
        actions = {
            drop;send;
        }
        const default_action = drop;
    }
    apply {
        if (hdr.ethernet.isValid())
{
            l2_fwd.apply();
        }
    }
}
```





Offload Interfaces – Runtime controls

P4 compatible runtime controls/CRUD operations requires knowledge of:

- Pipeline id: Indicative of p4 pipeline instance
- Table id: Indicative of table within given p4 pipeline instance
- Key and action: Attributes associated with a rule programming

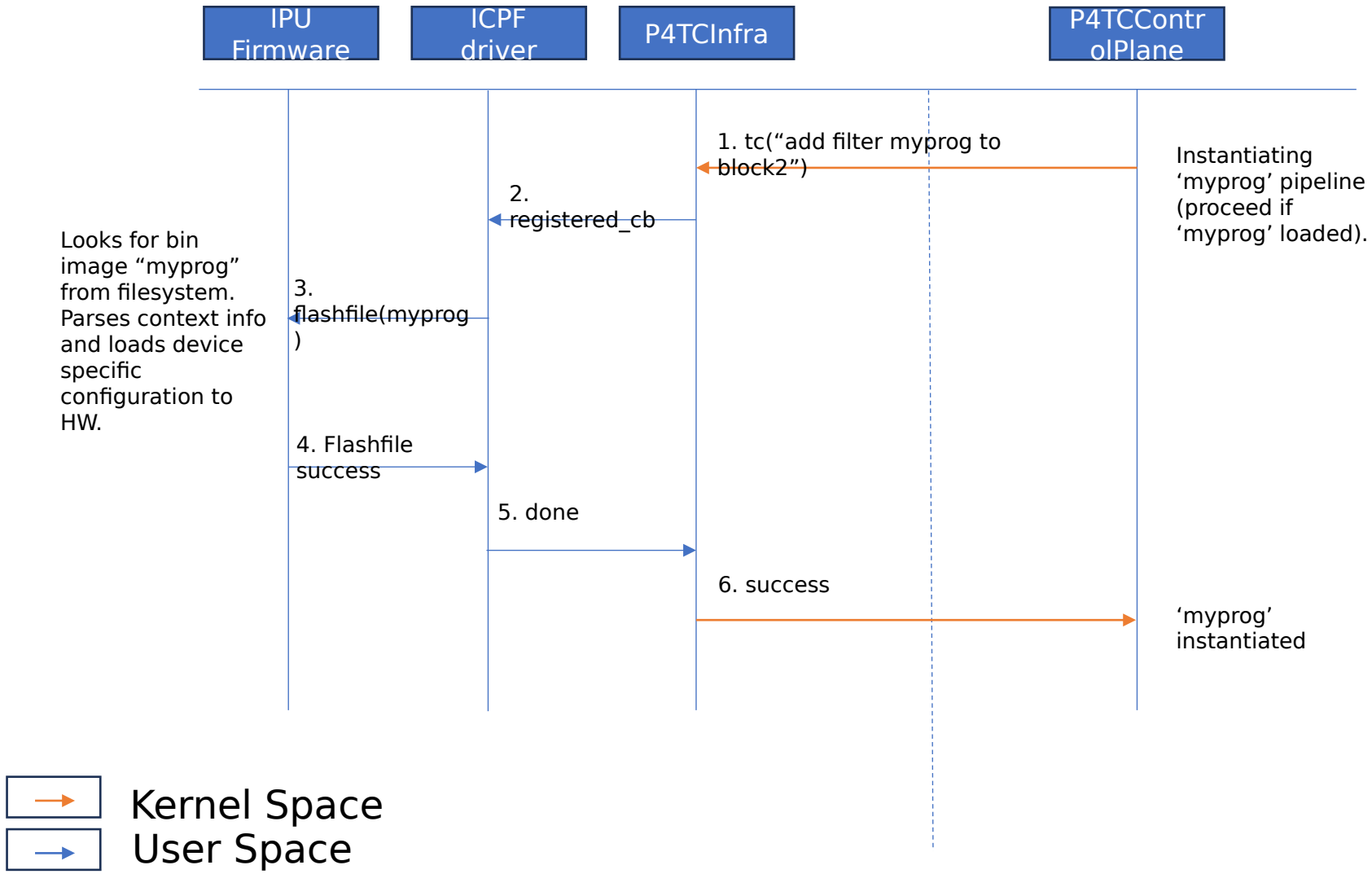
```
struct p4tc_offload_data {
        struct work_struct work;
        struct netlink_ext_ack *extack;
        struct p4tc_pipeline *pipeline;
        u32 pipe_id;
        u32 table_id;
        enum p4tc_offload_cmd cmd;
        enum p4tc_offload_obj obj;
        union {
                struct p4tc_offload_table_entry entry;
        };
};
```

```
int idpf_pipemgr_intf_ent_add(int dev_id, u32 pipe_id,
                              u32 mat_tbl_hdl,
                              struct tbl_match_spec
*match_spec,
                              u32 act_hdl,
                              struct tbl_action_spec
*action_spec);
```

HW P4 Package Association

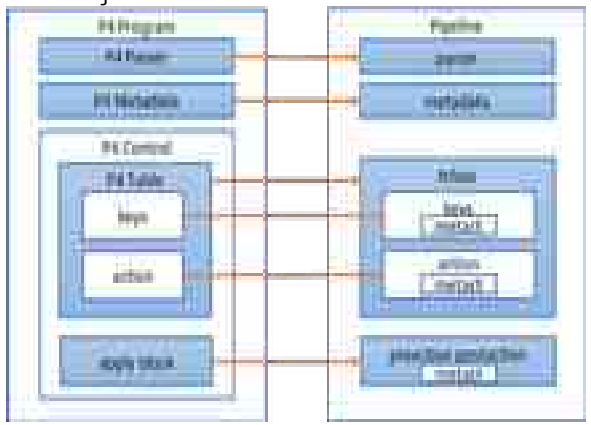
- Package file is a binary blob containing device specific information to associate your p4 dataplane initialization onto a P4 compliant device.
- This blob is loaded as part of tc filter add and is parsed by driver. Extracted contents will live in the driver for the life of P4 program.
- In our design, we propose one blob per p4 pipeline instance.

The sequence for loading context info and package for IPU 2100 (offload case)



P4 Program:

```
   action send(PortId_t vport) {
        send_to_port(vport);
    }
    action drop() {
        drop_packet();
    }

    /*
     * L2 forwarding table based on
destination MAC.
     */
    table l2_fwd {
        key = {
            hdr.ethernet.dstAddr:
exact;
        }
        actions = {
            drop;send;
        }
        const default_action = drop;
    }

    apply {
        if (hdr.ethernet.isValid()) {
            l2_fwd.apply();
        }
    }
```



Intel IPU E2100 Compiler

l2_match.pkg + context info

tc filter add dev eth0 protocol ip ingress flower skip_sw dstAddr 00:01:02:03:04:05 action drop

TC flower Kernel Classifier

Kernel Protocol Stack/Packet Path

ndo_setup_tc

tc flower ops handler

Representors (switchdev)

Devlink flash

Target abstraction SW

Pipe Manager

Exception Data Path TX/RX

IPU Init

IPU conf queues

IPU mailbox queues

IPU data queues

LLD (Low Level Driver) Layer

icpf driver

Linux kernel

L2 forwarding table

Intel IPU E2100

Supporting existing classifiers – tc flower

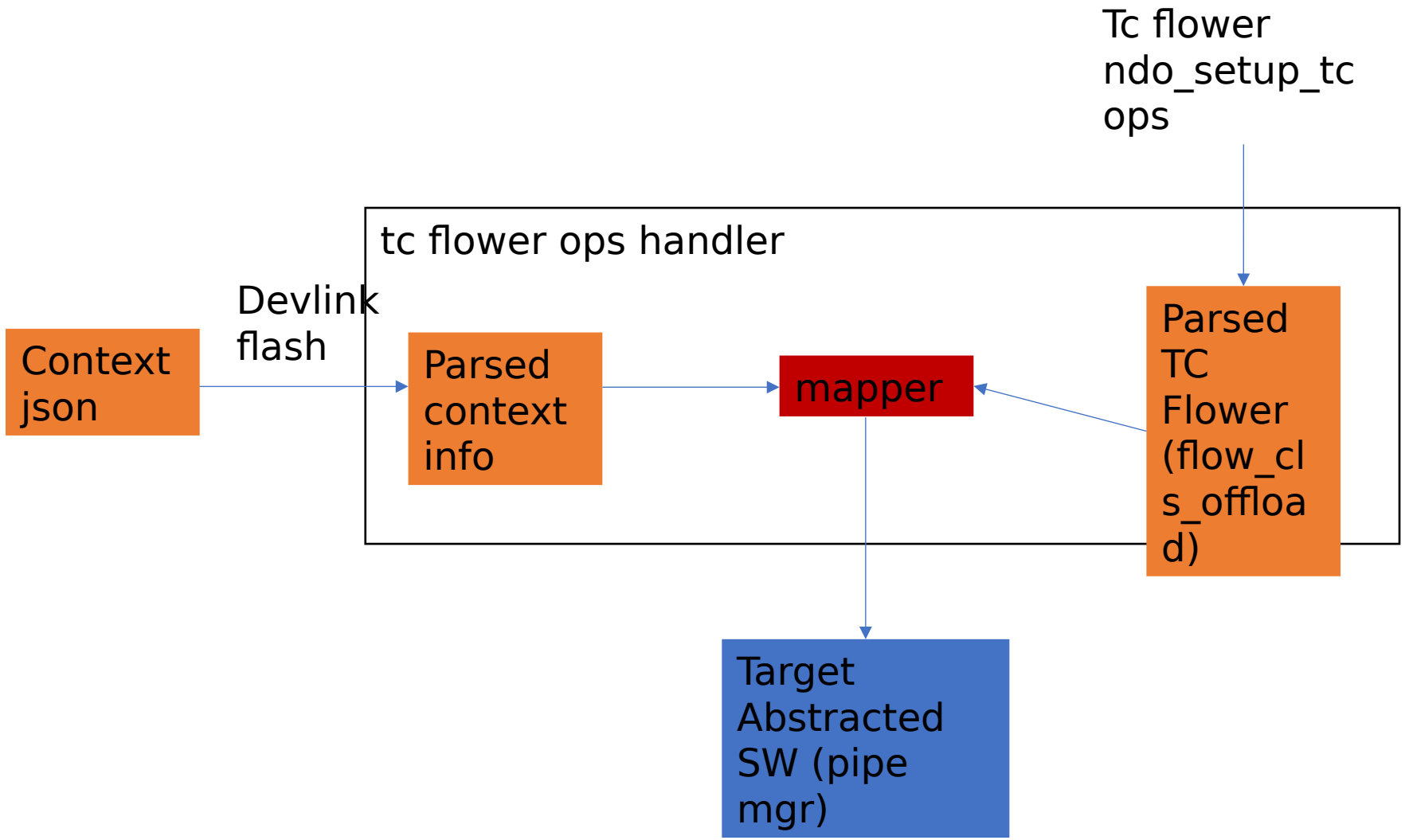
- Goal is to support tc_flower control plane applications to map to new P4 programmable devices with no changes to their existing code.
- We plan to introduce a library to support existing kernel classifier such as tc-flower.
- This library help map tc-flower rules to P4 table rules based on the annotations in the P4 programs running in HW.

Supporting existing classifiers – tc flower

Core functionalities supported by this library:

- Extracting key and action from tc flower rule add request
- Map extracted key and action to P4 compatible match and action attributes:
  - Model driven mapping to support mapping to any P4 pipeline in HW.
  - P4 specific params like pipeline_id, table_id are extracted and matched via binary blob loaded as part of pipeline instantiation.
- Configure P4 target device by calling appropriate P4-TC API’s.

TC flower operation handler – Architecture overview



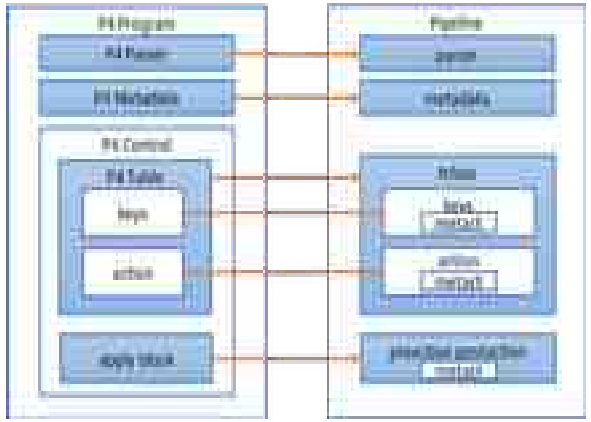
Design

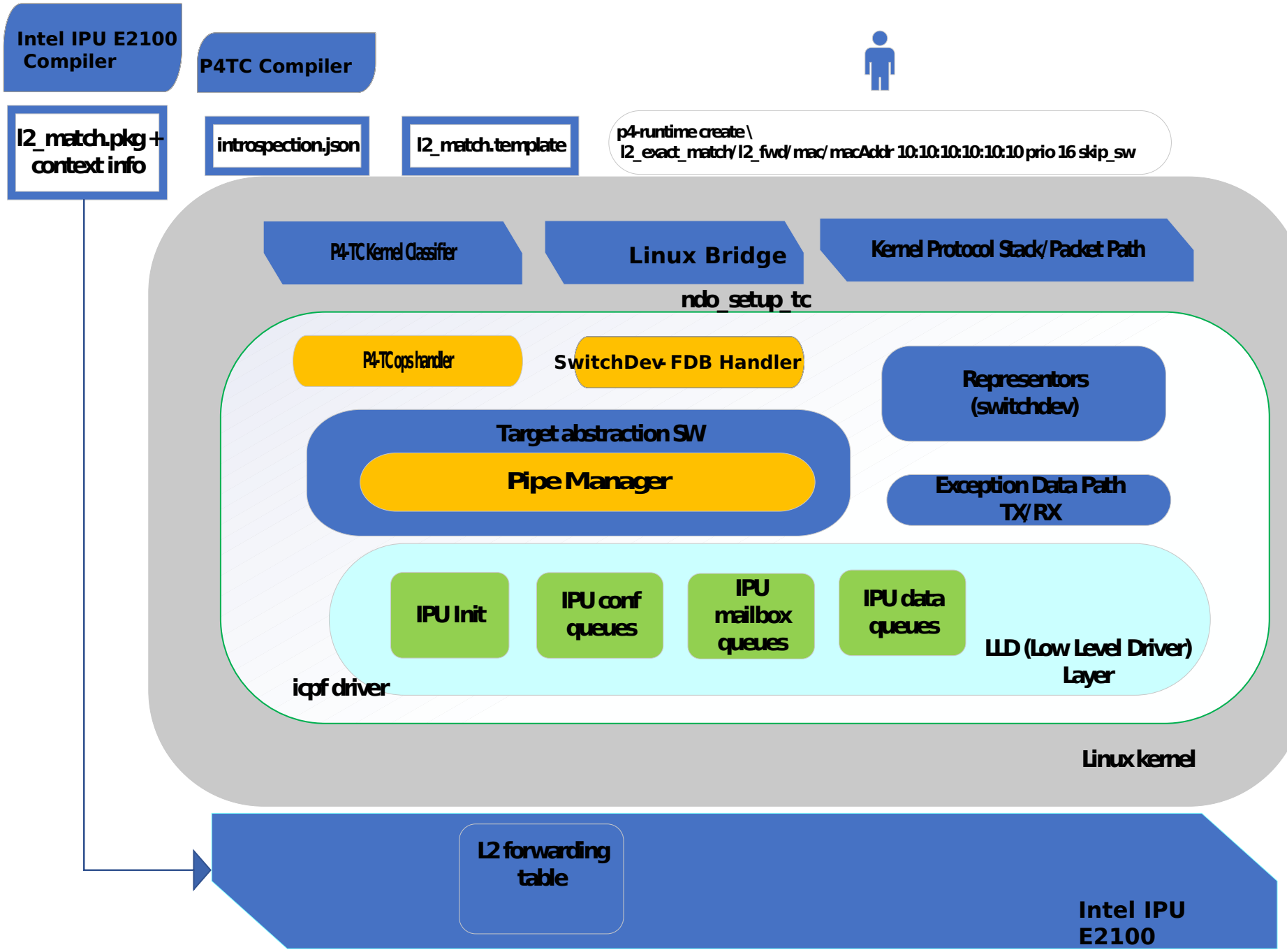
Snapshot of p4 program with annotations

```
table i_fwd {
        key = {
            hdrs.mac.da : exact
@tc_flower_type(FLOW_DISSECTOR_KEY_ETH_ADDR)
@tc_flower_field(dst);
            hdrs.mac.sa : exact
@tc_flower_type(FLOW_DISSECTOR_KEY_ETH_ADDR)
@tc_flower_field(src);
        }
        actions = {
            @tc_flower_action(FLOW_ACTION_DROP)
drop;

@tc_flower_action(FLOW_ACTION_REDIRECT) send;
        }
    }
```

P4 Program:

```
Ingress:

    table ingress_filter {
        key = {
            hdr.ethernet.srcAddr:
exact;
        }
        actions = {
            drop;bypass;
        }
        const default_action = drop;
    }
}

Egress:

    table egress_meter {
        key = {
            hdr.ethernet.srcAddr: exa
ct;
        }
        actions = {
            bypass; apply_meter;
        }
        const default_action =
bypass;
    }
}
```





SwitchDev FDB Model driven Map

Snapshot of p4 program with annotations

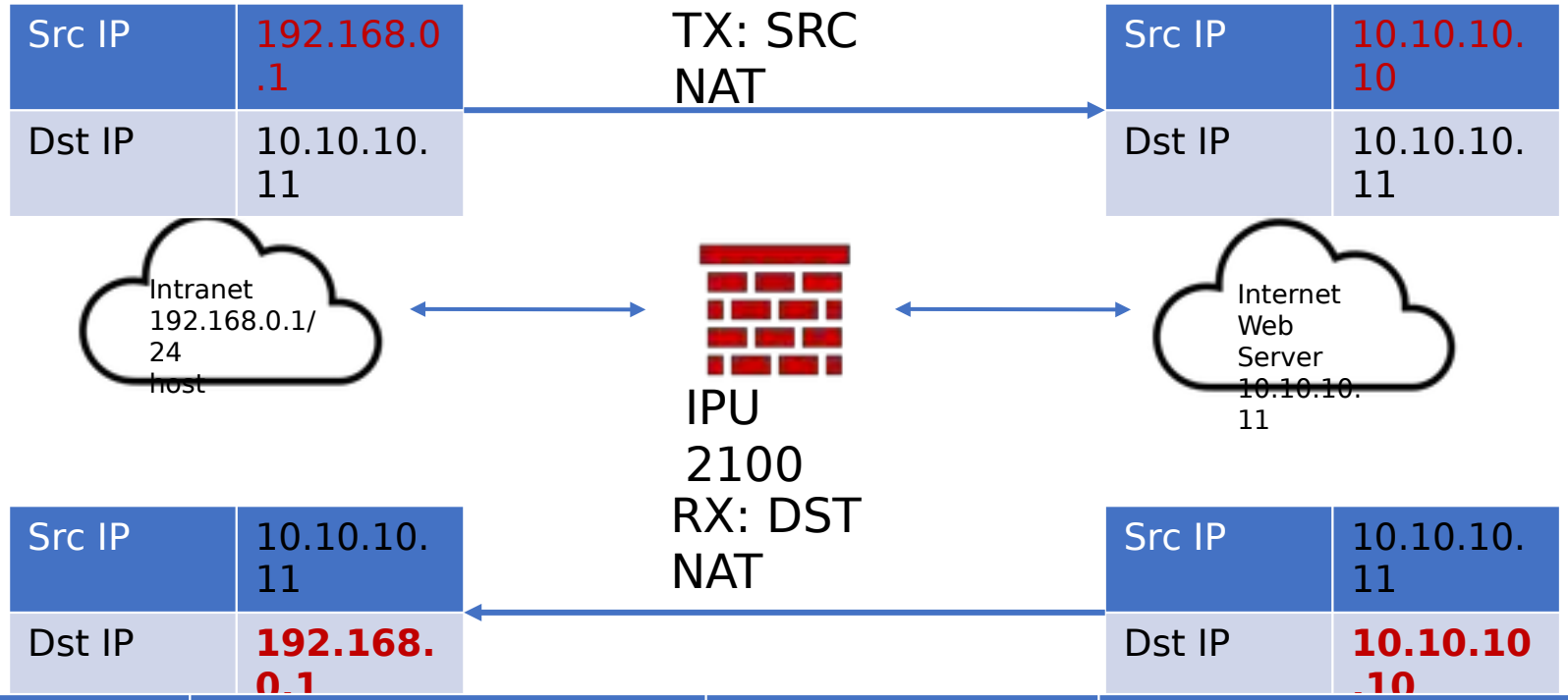
```
table l2_fwd {
        key = {
            hdr.ethernet.vid: exact

@switchdev_fdb_type(SWITCHDEV_FDB_ADD_TO_DEVICE)
    @switchdev_fdb_field(vid);
            hdr.ethernet.dst_addr: exact

@switchdev_fdb_type(SWITCHDEV_FDB_ADD_TO_DEVICE)
     @switchdev_fdb_field(addr);
        }
        actions = {

@switchdev_fdb_action(SWITCHDEV_FDB_ADD_TO_DEVICE)
send;
            @defaultonly NoAction;
        }
  }
```

P4-TC and FDB Co-existence

- Example: Ingress P4-TC for ACL -> FDB -> Egress P4-TC for Metering
- Control Plane Co-existence
  - Same Port Representers in both P4-TC pipelines and FDB instance.
  - Separate independent ingress and Egress P4-TC pipelines
  - P4-TC and Bridge control provisioning independently.
- Dataplane Co-existence
  - Independent pipeline integration
    - Ingress and Egress P4-TC pipelines process packets independently
    - No additional metadata carried in sk_buff between FDB and P4-TC pipelines
  - Integrated pipeline integration
    - Software pipeline acts as continuation of hardware P4 pipeline
    - Metadata from HW P4 pipeline mapped to SW P4 pipeline and carried from ingress P4 pipeline to egress P4 pipeline through FDB.
    - Evaluating options to fit with sk_buff.

P4-TC Offloaded NAT Demo

NAT DEMO: Source and Destination NAT Function Offload To IPU 2100



Category		Translated Item	
Source NAT	One to one mappings: Source address translation without port translation.	Source IP address	Allows the 1-to-1, static translation of a source IP address, but leaves the source port unchanged.
Destination NAT	One-to-one mapping: Destination address translation from public to private IP addresses.	Destination IP address	Use Static Destination public IP address translation to a private IP addresses.

Source Info: https://docs.paloaltonetworks.com/pan-os/9-1/pan-os-admin/networking/nat/configure-nat

NAT P4 Program(simple_nat_mod_demo.p4)

```
control my_control(inout parsed_headers_t hdrs,
                   inout user_metadata_t user_meta,
                   inout vendor_meta_t vmeta,
                   out user_rx_host_metadata_t user_rx_host_meta,
                   in user_tx_host_metadata_t user_tx_host_meta,
                   in pna_main_input_metadata_t istd,
                   inout pna_main_output_metadata_t ostd)

{

        action mod_src_ip(@tc_type ("ipv4") bit<32> new_src_ip)
        {
                hdrs.ipv4[vmeta.common.depth].src_ip = new_src_ip;
        }

        action mod_dst_ip(@tc_type ("ipv4") bit<32> new_dst_ip)
        {
                hdrs.ipv4[vmeta.common.depth].dst_ip = new_dst_ip;
        }

        action tx_mod_index(@tc_type("dev") PortId_t port, bit<24> index)
        {
                vmeta.common.mod_blob_ptr = (bit<24>)index;
                vmeta.common.mod_action = (bit<11>)MOD_SRC_IP;
                send_to_port(port);
        }

        action rx_mod_index(@tc_type("dev") PortId_t port, bit<24> index)
        {
                vmeta.common.mod_blob_ptr = (bit<24>)index;
                vmeta.common.mod_action = (bit<11>)MOD_DST_IP;
                send_to_port(port);
        }

        table mod_src_ip_table {
                key = {
                        vmeta.common.mod_blob_ptr : exact;
                }
                actions = {
                        mod_src_ip;
                }
        }

        table mod_dst_ip_table {
                key = {
                        vmeta.common.mod_blob_ptr : exact;
                }
                actions = {
                        mod_dst_ip;
                }
        }

        table e_fwd {
                key = {
                        hdrs.ipv4[vmeta.common.depth].src_ip : exact;
                }

                actions = {
                        tx_mod_index;
                }
        }
        table i_fwd {
                key = {
                        hdrs.ipv4[vmeta.common.depth].dst_ip : exact;
                }

                actions = {
                        rx_mod_index;

                }
        }

        apply {
                /*
                 * Perform exact match actions
                 */
                if (hdrs.ipv4[vmeta.common.depth].isValid() && TxPkt(istd)) {
                        /*
                         * Forward based on the sole src IP.
                         * Forward either to local vport or externally.
                         */
                        e_fwd.apply();
                } else if (hdrs.ipv4[vmeta.common.depth].isValid() && RxPkt(istd)) {
                        /*
                         * Forward based on the sole dst IP.
                         * Forward either to local vport or externally.
                         */
                        i_fwd.apply();
                }

                switch (vmeta.common.mod_action) {
                        MOD_SRC_IP: { mod_src_ip_table.apply(); }
                        MOD_DST_IP: { mod_dst_ip_table.apply(); }
                }
        }
}
```

Note: P4 program doesn’t need to explicitly specify checksum calculation. It’s Offloaded to IPU 2100.

Other authors

- Anjali Singhai, Intel Corporation
- Sridhar Samudrala, Intel Corporation
- Namrata Limaye, Intel Corporation
- Mahesh Shirshyad, AMD
- Vipin Jain, AMD