Netdev Conference 2022

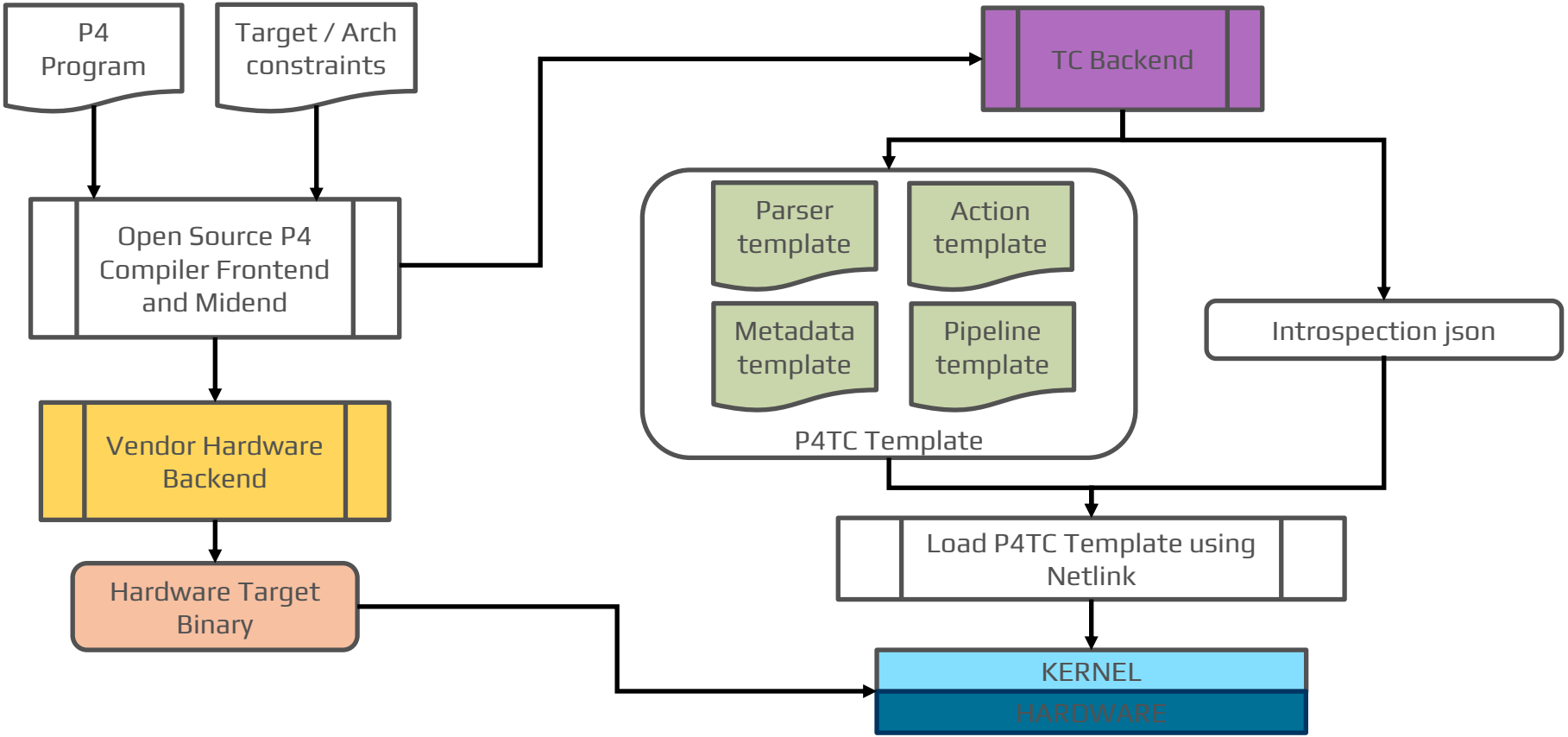
P4 Compiler Backend for P4TC

Sosutha Sethuramapandian – Cloud Software Development Engineer

Overview of P4 and P4 Architecture

- P4 is a high-level language for programming the data plane of network devices. P4 Programs can be compiled and loaded to target pipeline.
- Any P4 program comprises an architecture file and user program.
- Architecture file defines the structure and identifies programmable pipeline blocks (written in P4) – it serves as a contract between P4 program and the target.
- User program includes this architecture file and define packet processing pipeline for the business needs.
- Two standardized architectures provided by P4 community are
    - PNA (Portable NIC Architecture)
    - PSA (Portable Switch Architecture)

Compiler Workflow for P4TC Environment



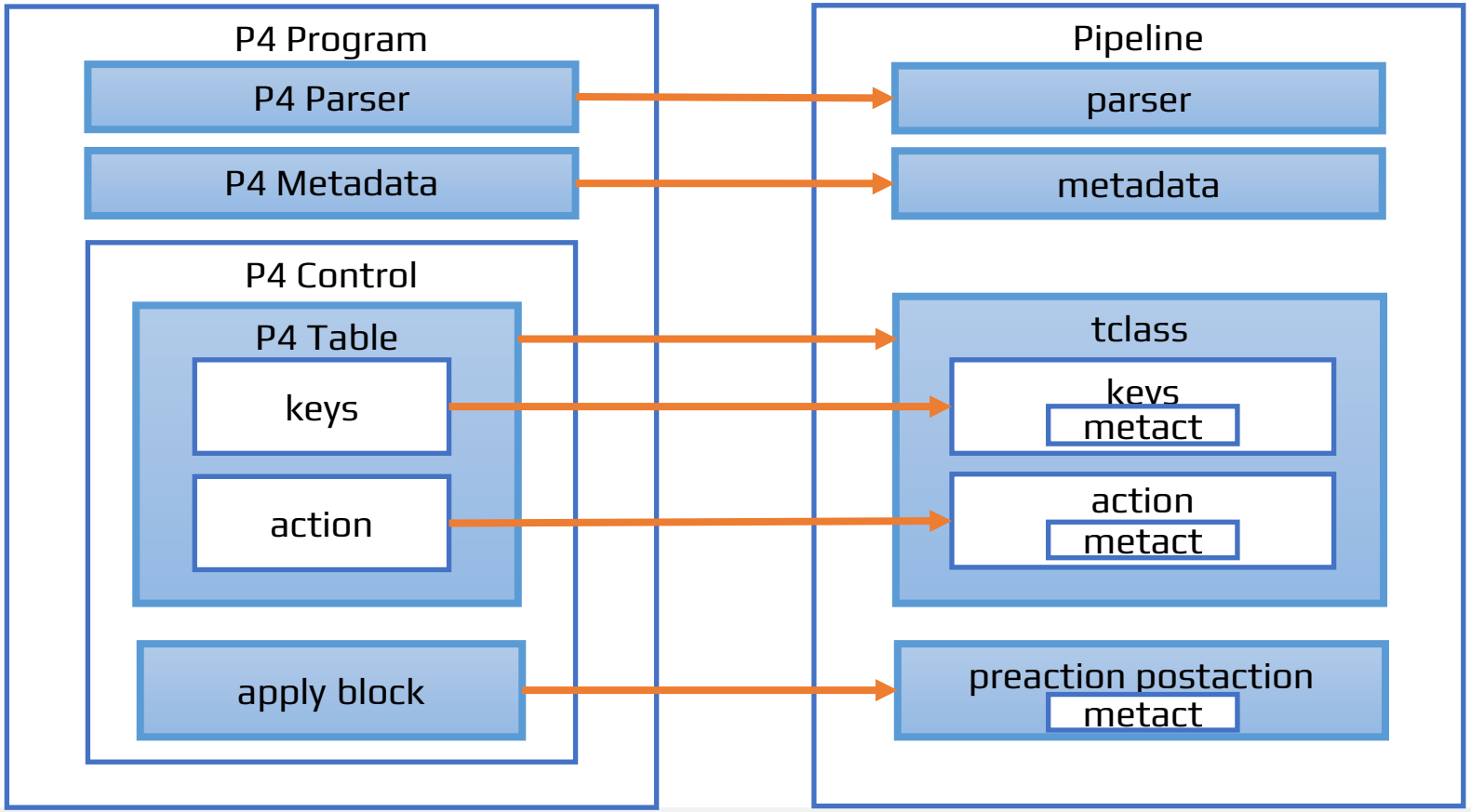
About P4TC Compiler

- Initial version of the P4TC compiler supports PNA architecture
- Uses open source P4 Compiler frontend and midend
- Target/Arch constraint file has target specific constraints which is additional input to the compiler apart from p4 program
- Generates P4TC Template script and introspection json for the given P4 program written using PNA architecture
- Plan to support both PNA and PSA architecture
  https://p4.org/p4-spec/docs/PNA-v0.5.0.html
  https://p4.org/p4-spec/docs/PSA-v1.1.0.html

P4TC Template Sections

- **Parser template**: To represent the parse graph given in the P4 and for parsing of the headers present in the packet
- **Metadata template**: To represent the metadata used in processing of the packet information
- **Action template**: To represent the action that need to be carried out on the parsed headers and metadata
- **Pipeline template**: To represent the overall pipeline control and the order of execution of the actions based on the given P4

Mapping of P4 Constructs to P4TC Template



P4TC Template Objects

- **tclass**: template object that has properties similar to p4 table
- **key**: template object that is mapped to table keys from p4
- **preactions** and **postactions**: Part of both tclass as well as pipeline template objects. They are used to represent the sequence or conditions under which actions (in case of tclass) or tables (in case of pipeline) need to be executed
- **metact**: metact action allows programmatic computation on header fields, keys, intrinsic, and user-defined metadata

Example p4 and p4tc snippets

```
action drop() {
   drop_packet();
}


action set_nhop(macAddr_t dstAddr, bit<9> port) {
   hdr.ethernet.srcAddr = hdr.ethernet.dstAddr;
   hdr.ethernet.dstAddr = dstAddr;
   hdr.ipv4.ttl = hdr.ipv4.ttl - 1;
   send_to_port(port);
}
```

```
# action template declaration
tc p4template create action/example/MainControlImpl/drop

# action template definition
tc p4template update action example/MainControlImpl/drop \
   metact \
   cmd act kernel.drop

# metadata creation
tc p4template create metadata/example/istd/direction mid 1 size 1

# action template declaration
tc p4template create action/example/MainControlImpl/set_nhop \
    param dstAddr type bit48 \
    param port type bit9

# action template definition
tc p4template update action example/MainControlImpl/set_nhop \
   metact \
   cmd set hdr/ethernet/srcAddr hdr/ethernet/dstAddr \
   cmd set hdr/ethernet/dstAddr param/dstAddr \
   cmd decr hdr/ipv4/ttl \
   cmd act kernel.mirred metadata/example/istd/direction redirect
dev param/port
```

```
table simple_match {

    key = {
      hdr.ipv4.dstAddr: exact;
      hdr.ipv4.srcAddr: exact;
    }
    actions = {
      set_nhop;
      drop;
    }
    size = 1024;
    default_action = drop;
  }

apply {
    if (istd.direction == NET_TO_HOST) {
      simple_match.apply();
    }
  }
}
```

```
# table creation
tc p4template create tclass/example/MainControlImpl/simple_match \
  tbcid 1 keysz 32 nummasks 8 tentries 1024 \

  keys id 1 default action metact cmd set key \
  hdrfield/example/MainParserImpl/hdr/ipv4/dstAddr \

  keys id 2 action metact cmd set key  \
  hdrfield/example/MainParserImpl/hdr/ipv4/srcAddr \

  postactions \

    action metact cmd beq results/hit true control pipe / jump 1 \
    cmd act results/action control ok \
    cmd act kernel.drop

# pipeline creation
tc p4template create pipeline/example pid 1 numtclasses 1 \
  preactions \
    action metact \
    cmd beq metadata/example/istd/direction 0 control pipe / control ok \
    cmd act tableapply simple_match control ok \
  postactions \
    action metact cmd act ok
```

Upstreaming the tc Backend to Open Source

- We aim to add tc backend as one of the open source backend along with dpdk, ebpf to the open source p4c repo https://github.com/p4lang/p4c

```
|--backends
|   |
|   |--bmv2
|   |--common
|   |--dpdk
|   |--ebpf
|   |--graphs
|   |--p4test
|   |--p4tools
|   |--tc (to be added)
|   |--ubpf
|--bazel
|--cmake
```

- Upstreaming will be done once the tc backend is stabilized with good number of testing and verification
- Plan is to upstream by Mar 31, 2023

Thank You