DPDK
DATA PLANE DEVELOPMENT KIT

DPDK Summit

DPDK based Fast Path Programing using P4

-Praveen Desu & Ram Subramoniam

Agenda

- **P4 Overview**
- **P4 & DPDK**
- **DEMO – P4 to DPDK programing**

P4 Overview

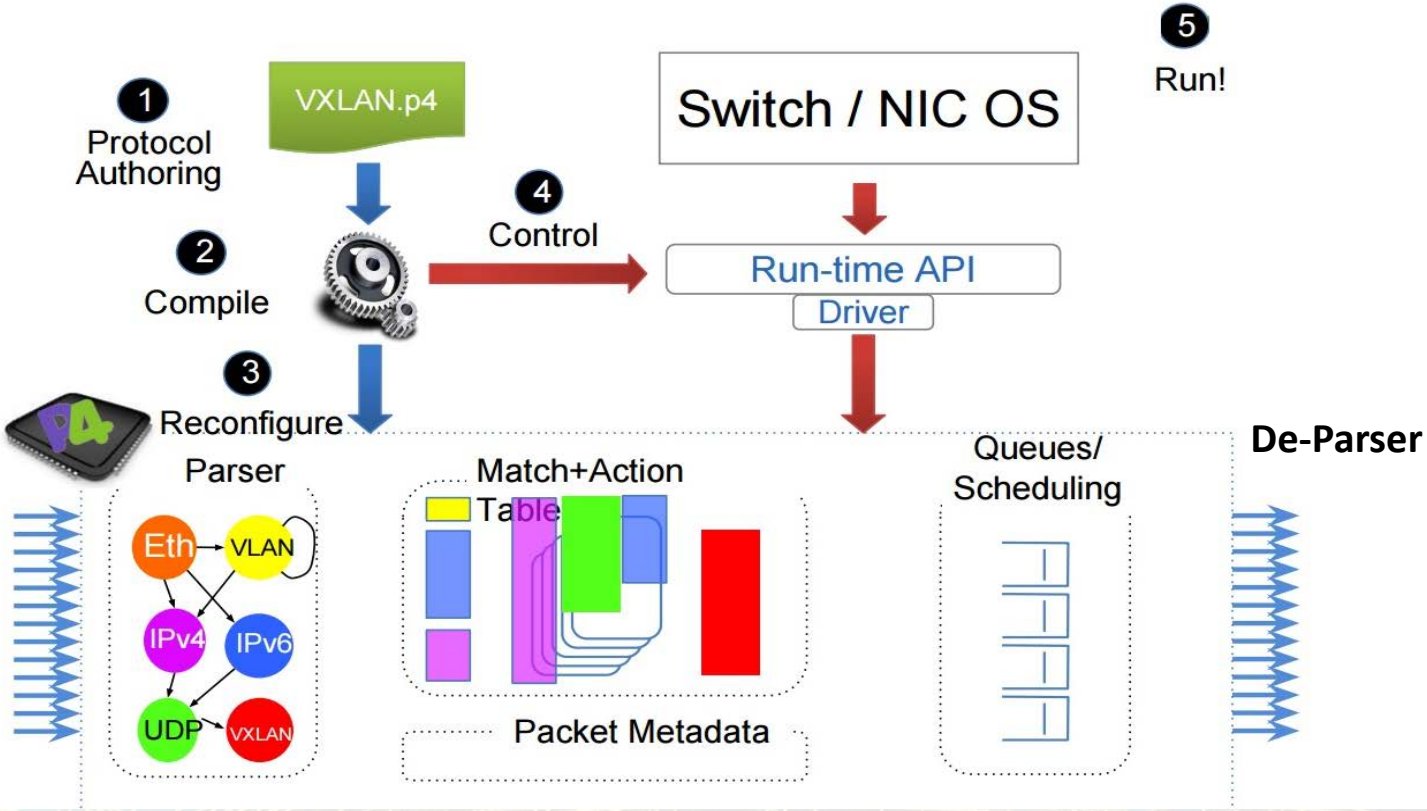
Why P4 ?

- Virtually all are closed/proprietary
- Low level programming
- Device dependent (tightly Coupled)
- No re-use, no code commonality –> wheel reinvention

What is P4 ?

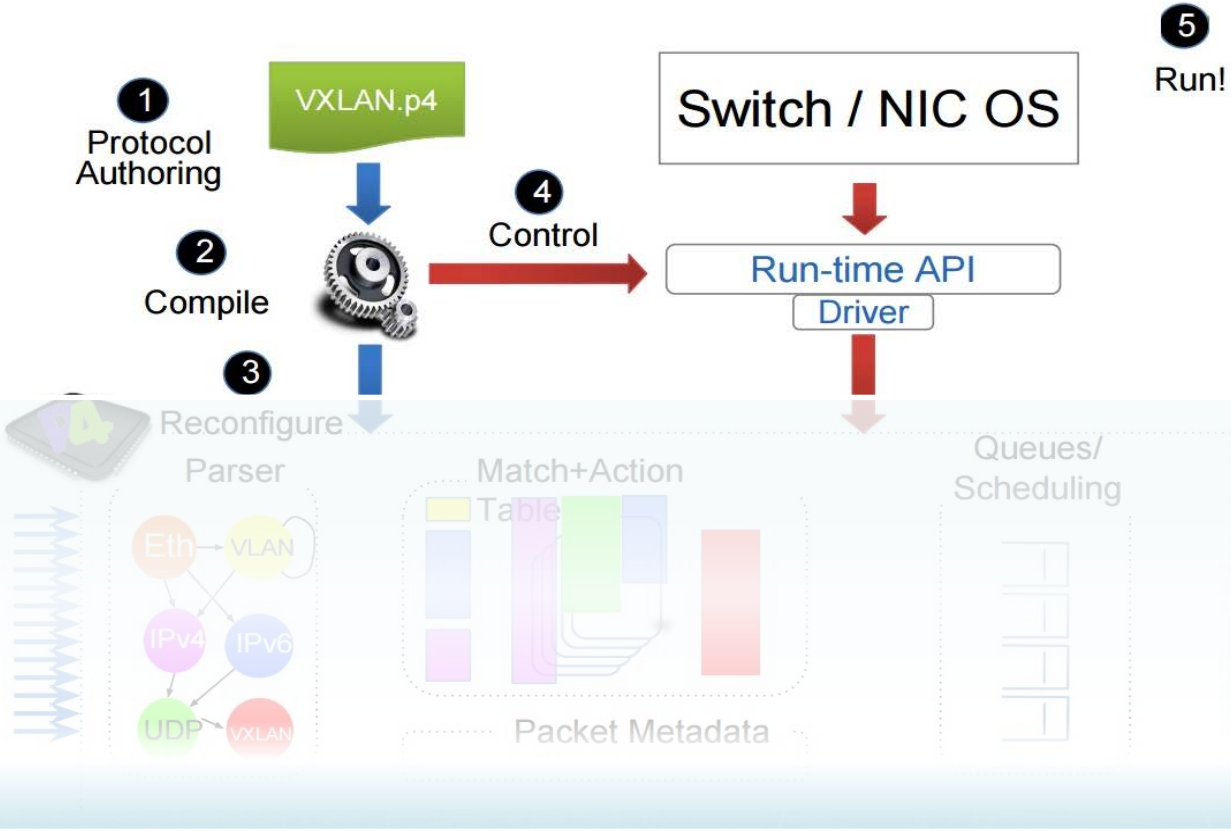
- Open source
- High Level Programming Language
- Protocol independent
- Device (“target”) independent
- Based on “**match+Action**” forwarding model
- Program can be re-used.
- Allows automatic generations of API for managing packet processing table

P4 Based Workflow

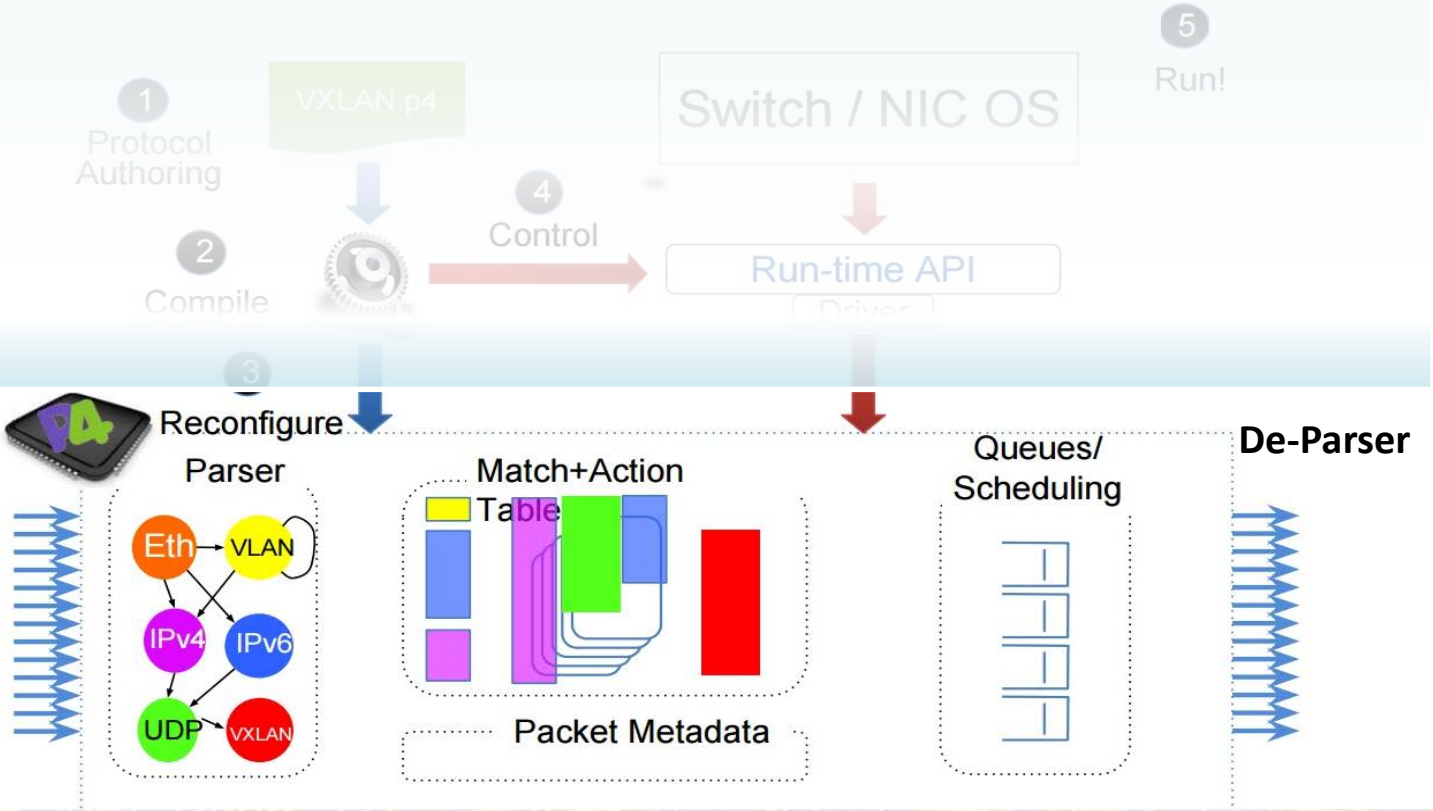


Source: https://raw.githubusercontent.com/rkuo/NetworkOS/master/images/P4_D2_2016_P4_Workflow.jpg

P4 Based Workflow

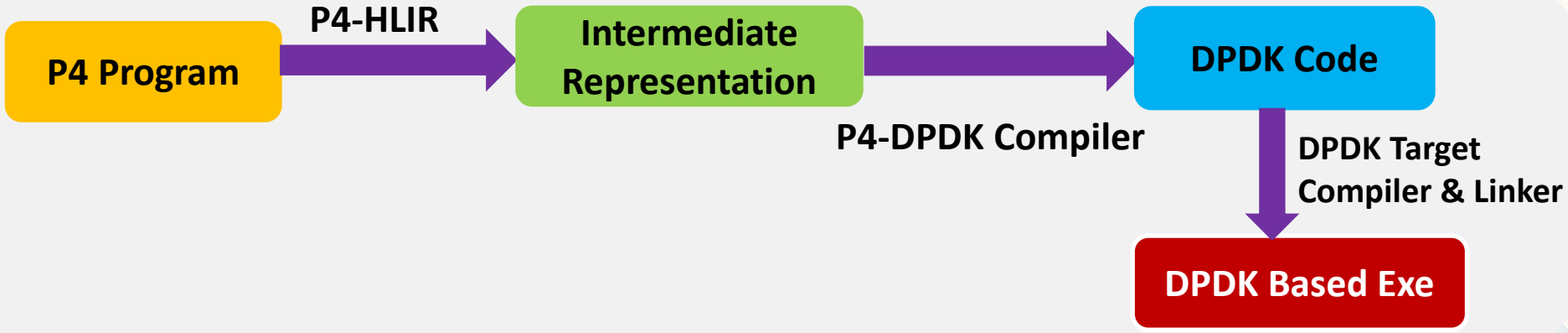


P4 Based Workflow



P4 Compiler For DPDK

Two Stage Compiler – Single Frontend and Multiple Backend	
• Single front-end (P4 to high level intermediate representation (HLIR)) o Translates p4 code to HLIR	• Multiple back-ends: o Input to HLIR o Able to generate specific DPDK code and configuration for variety of platforms.



Parsing

```
parser start {
   return parse_ethernet;
}

parser parse_ethernet {
   extract(ethernet);
   return ingress;
}
```

Match

```
table smac {
   reads {
      ethernet.srcAddr : exact;
   }
   actions {mac_learn; _nop;}
   size : 512;}
table dmac {
   reads {
      ethernet.dstAddr : exact;}
   actions {forward; bcast;}
   size : 512;}
```

Action

```
action _drop() {
   drop();
}
action _nop() {
}

action mac_learn() {  generate_digest(MAC_LEARN_RECEIVER, mac_learn_digest);}

action forward(port) {  modify_field(standard_metadata.egress_port, port); }

action bcast() {    modify_field(standard_metadata.egress_port, 100);
```

DPDK Compiler

DPDK Output

```
table_hash_create(int socketid, const char*
name, uint32_t keylen, rte_hash_function
hashfunc)
   struct rte_hash_parameters hash_params = {
      .name = NULL,
      .entries = HASH_ENTRIES,
      .key_len = keylen,
      .hash_func = hashfunc,
      .hash_func_init_val = 0,
   };
   hash_params.name = name;
   hash_params.socket_id = socketid;
   struct rte_hash *h =
rte_hash_create(&hash_params);
   if (h == NULL)
      create_error(socketid, "hash");
   return h;
}

unit8_t *
hash_table_exact_lookup(lookup_table_t* t,
uint8_t* key))
{
   ..........................................................
                              ..........................................
......
   extended_table_t* ext =
(extended_table_t*)t->table;
   int ret = rte_hash_lookup(ext->rte_table, key);
                              ..........................
                              ..........................................
}
```

P4 to DPDK code Generator – Compilation Demo

THANK YOU