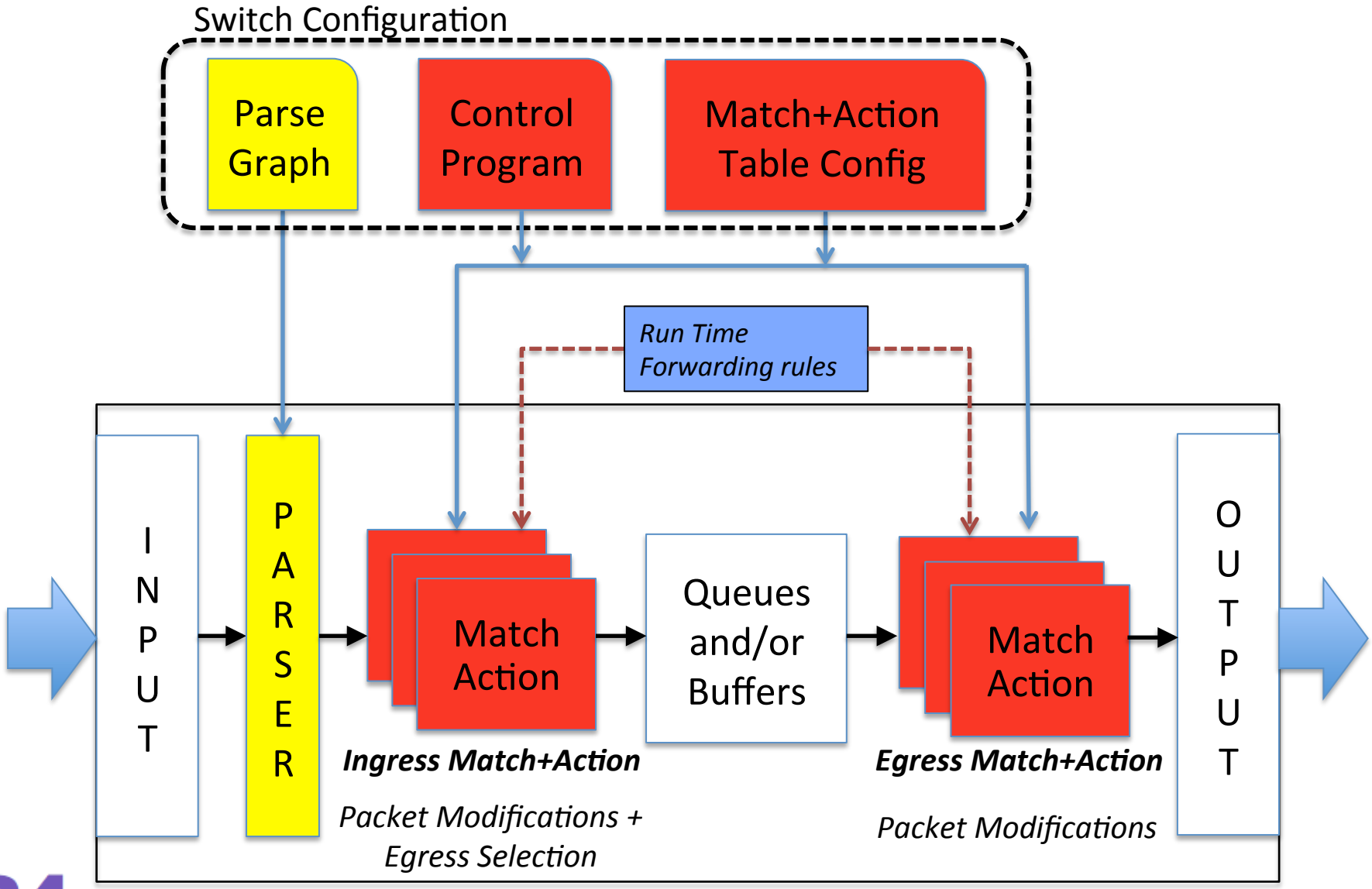
An Introduction to P4

What is P4?

- An open source language allowing the specification of packet processing logic
- Based on a Match+Action forwarding model
- Allows the automatic generation of APIs to manage the packet processing tables
- Multiple companies have written compilers

The P4 Forwarding Abstraction



Headers and Fields

- Declare headers and fields
- Fields have width and other attributes
- Headers are collections of Fields
- These are types which are used to declare instances

```
header_type ethernet_t {
  fields {
    dstAddr    : 48;
    srcAddr    : 48;
    etherType  : 16;
  }
}

/* Instance of eth header */
header ethernet_t inner_ethernet;
```

```
header_type egress_metadata_t {
  fields {
    nhop_type   : 8;  /* 0: L2, 1: L3, 2: tunnel */
    encap_type  : 8;  /* L2 Untagged; L2ST; L2DT */
    vnid        : 24; /* vxlan vnid/gre key */
    tun_type    : 8;  /* vxlan; gre; nvgre;*/
    tun_idx     : 8;  /* tunnel index */
  }
}

metadata egress_metadata_t egress_metadata;
```

The Parser

- Specify a Parser
- Imperative functions for “states”
- Extract header instances
- Select a next “state” by returning a parser function

```
parser parse_ethernet {
  extract(ethernet);
  return select(latest.etherType) {
    ETHERTYPE_CPU  : parse_cpu_header;
    ETHERTYPE_VLAN : parse_vlan;
    ETHERTYPE_MPLS : parse_mpls;
    ETHERTYPE_IPV4 : parse_ipv4;
    ETHERTYPE_IPV6 : parse_ipv6;
    ETHERTYPE_ARP  : parse_arp_rarp;
    ETHERTYPE_RARP : parse_arp_rarp;
    ETHERTYPE_NSH  : parse_nsh;
  }
}
```

- Produces a *Parsed Representation* of the packet

The Parsed Representation

- The Parser produces a "Parsed Representation" (PR) of the packet: The set of header instances to which the Parser extracted data
- Match+Action tables operate on the PR header instances
- The packet is regenerated based on the updated PR header instances.

Packet Processing

- Bottom-up view of the structures in P4 that describe how the packet is processed
  - Action Functions
  - Table Specification
  - Control Flow

Action Functions

- Actions are specified imperatively from primitives
  - Executed in parallel
  - modify field (packet header or metadata)
  - add/remove header
  - counter/meter/state-full memory operations

```
/* Ingress logical interface setup */
action set_ingress_intf(i_lif, bd, vrf, v4term) {
  modify_field(route_md.i_lif, i_lif);
  modify_field(route_md.bd, bd);
  modify_field(route_md.vrf, vrf);
  modify_field(route_md.ipv4_term, v4term, 0x1);
  . . .
}
```

Match+Action Tables

- Specification of:
  - What to examine from each packet
  - What are the permitted actions that can be applied
  - What resources to allocate to the table

```
table port_vlan {
    reads {
        standard_metadata.ingress_port : exact;
        vlan_tag.vid : exact;
    }
    actions {
        drop, ingress_intf_extract;
    }
    size 16384;
}
```

Tables and Actions on the Chip

```
table port_vlan {
    reads {
        standard_metadata.ingress_port : exact;
        vlan_tag[OUTER_VLAN_IDX].vid : exact;
    }
    actions {
        drop, ing_lif_extract;
    }
    . . . .
}
```

```
action ing_lif_extract(i_lif, bd, vrf) {
    . . . .
}
```

Table Entry

Ing Port	VLAN id	i_lif	bd	vrf	01001010110011

Match | Action Parameters | Action Instruction

Format of port_vlan table entry on the device:

P4 + Compiler: Gives format of match and action entries; determines action instruction

Run time population: Determines the values used for match and action data

Example Auto Generated API

```
table host_route {
    reads {
        routing_metadata.vrf : exact;
        ipv4.dstAddr : exact;
    }
    actions {
        host_route_miss; /* default */
        route;
    }
}

action route(ecmp_count, ecmp_base) {
    modify_field(
        routing_metadata.ecmp_count,
        ecmp_count);
    modify_field(
        routing_metadata.ecmp_base,
        ecmp_base);
    add_to_field(ipv4.ttl, -1);
}
```

P4 code

```
entry_handle_t
host_route_add_with_route(
    uint16_t routing_metadata_vrf,
    uint32_t ipv4_dstAddr,
    uint32_t action_ecmp_count,
    uint32_t action_ecmp_base);
```

Auto generated API

Control Flow

- Specified imperatively

```
control ingress {
    apply(port);
    apply(host_ip) {
        miss {
            apply(lpm_ip);
        }
    }
    if (valid(vlan_tag[0])) {
        apply(port_vlan) {
            hit { apply(remap_vni); }
        }
    }
    apply (bridge_domain);
    if (valid(mpls_bos)) {
        apply(mpls_label);
    }
    retrieve_tunnel_vni();
    if (valid(vxlan) or valid(nvgre)) {
        apply(dest_vtep);
        apply(src_vtep);
    }
    . . . .
}
```

P4 in a nutshell

- Simple Language
  - Programmable Parser
    - Headers, Fields, State Machine
  - Match+Action Tables
  - Control Flow
- C-like Syntax
- Strong “compile time” vs “run time” division
- Efficient Execution

Thank You

P4 Language Consortium

http://p4.org