TeraFlow SDN

Introduction to P4

© ETSI CC-BY-4.0

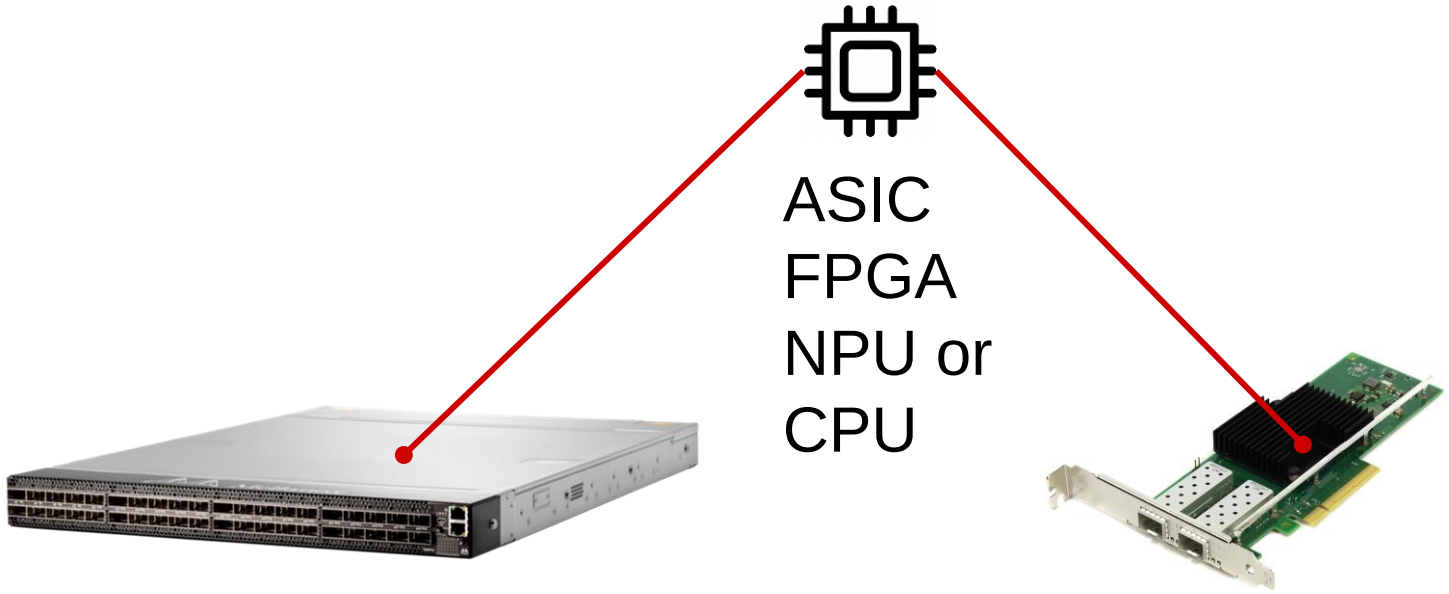
P4 Demonstration - Preparation

If you want to run this demo on your VM, please follow the next steps:

1. Install docker-compose:
   - `sudo apt-get install docker-compose`
2. Clone ONF's next-generation SDN Tutorial:
   - `cd ~`
   - `git clone -b advanced https://github.com/opennetworkinglab/ngsdn-tutorial`
3. Install dependencies:
   - `cd ngsdn-tutorial`
   - `make deps`
4. Insert the following in the Makefile (`~/ngsdn-tutorial/Makefile`):
   - ```
     start-simple: NGSDN_TOPO_PY := topo-simple.py
     start-simple: _start
     ```
5. Copy this tutorial's simple topology file into the ngsdn-tutorial repo:
   - `cp ~/tfs-ctrl/src/tests/netx22-p4/mininet/topo-simple.py ~/ngsdn-tutorial/mininet/`

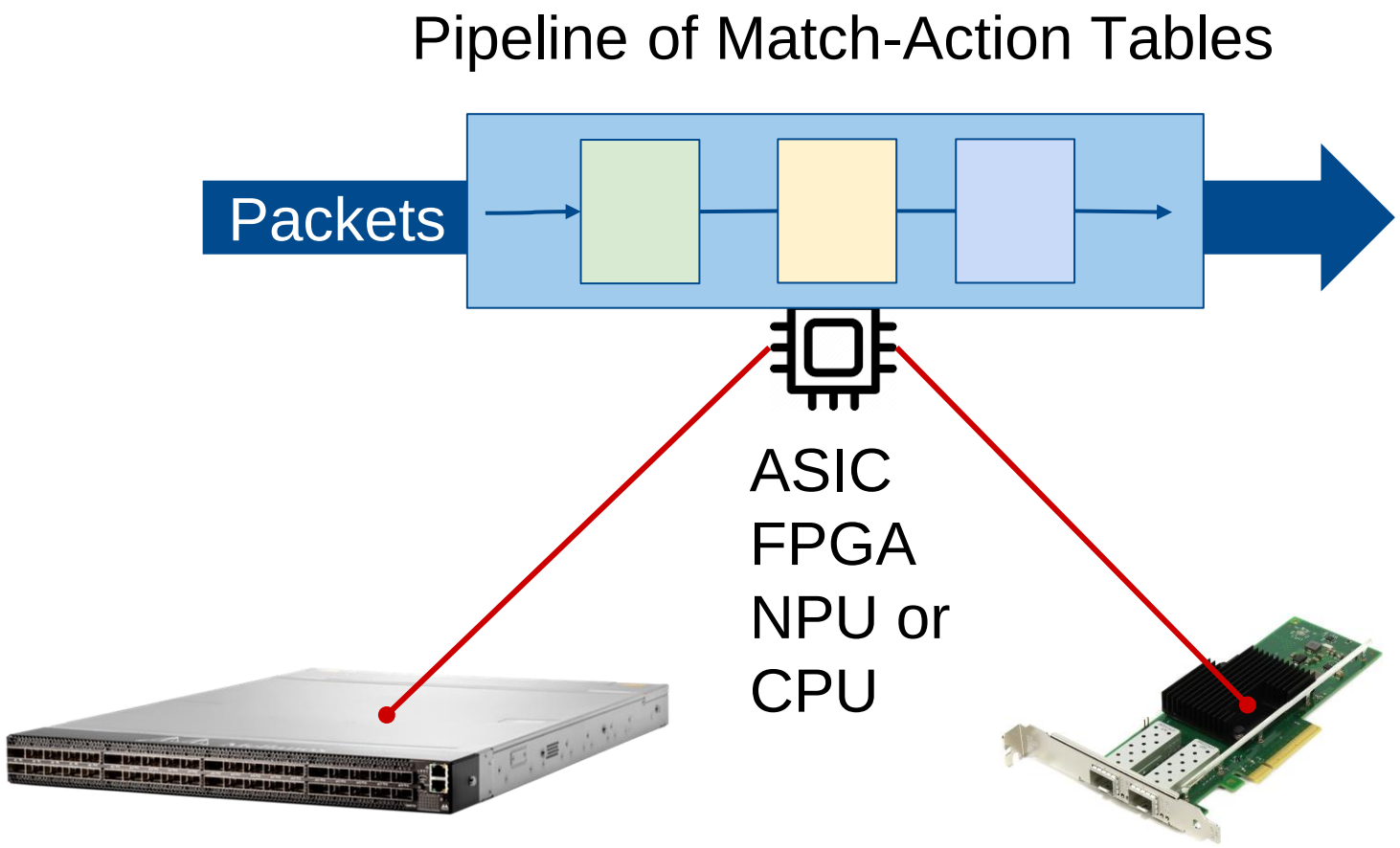
© ETSI CC-BY-4.0
     ```

What is a packet processing pipeline?



© ETSI CC-BY-4.0

What is a packet processing pipeline?



© ETSI CC-BY-4.0

The need for a different SDN

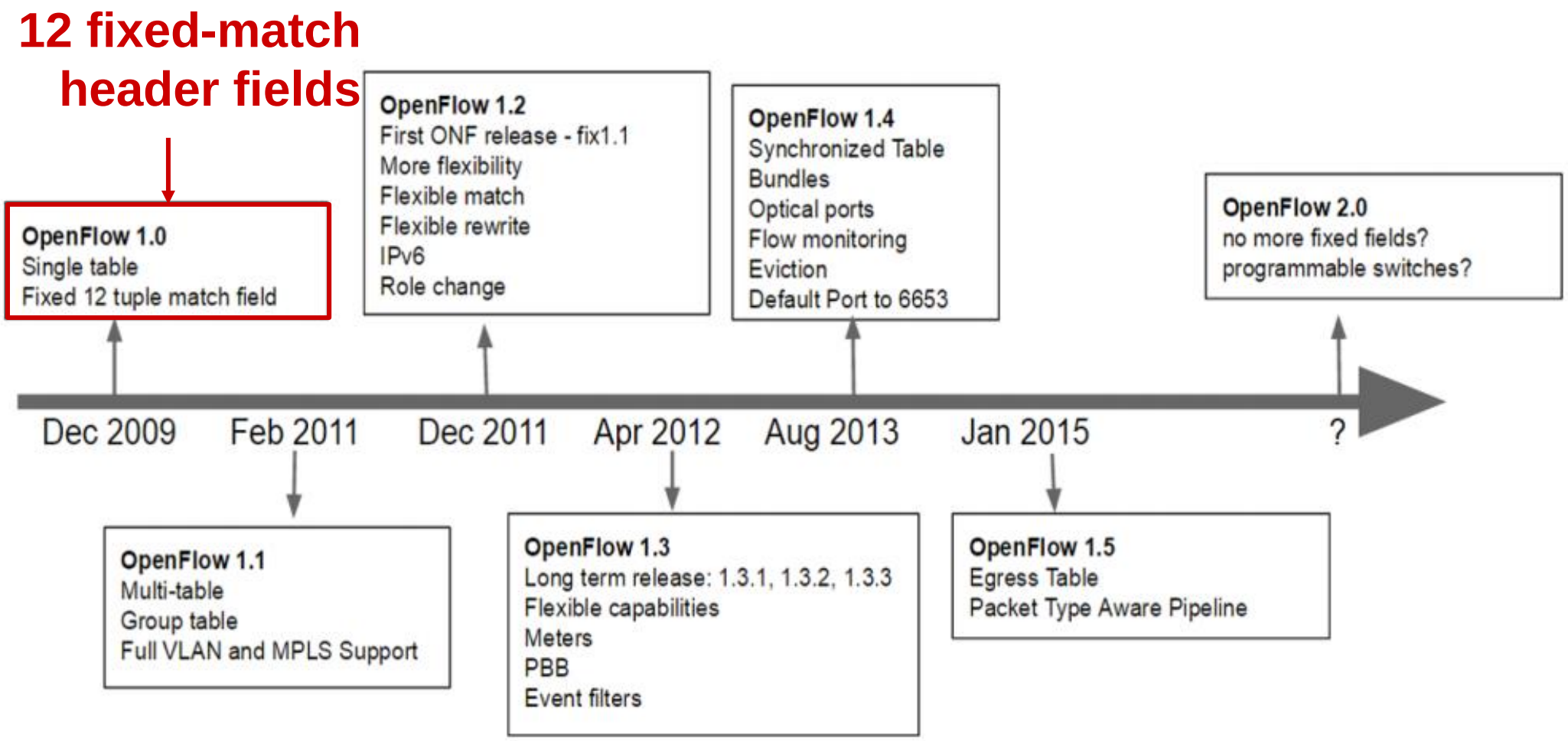


***Figure source:** https://kspviswa.github.io/OpenFlow_Version_Roadmap.html

© ETSI CC-BY-4.0

The need for a different SDN

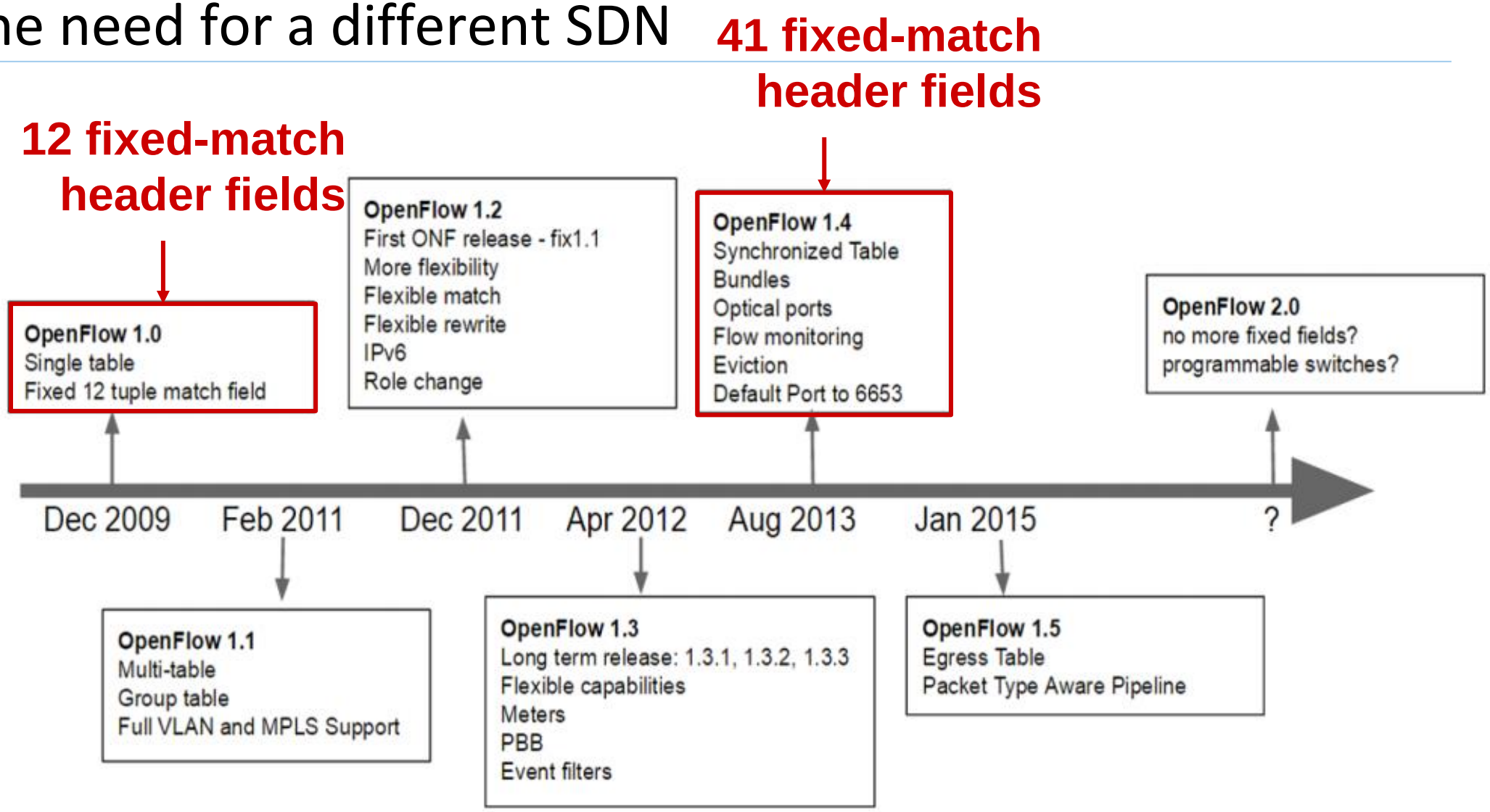


***Figure source:** https://kspviswa.github.io/OpenFlow_Version_Roadmap.html

© ETSI CC-BY-4.0

The need for a different SDN

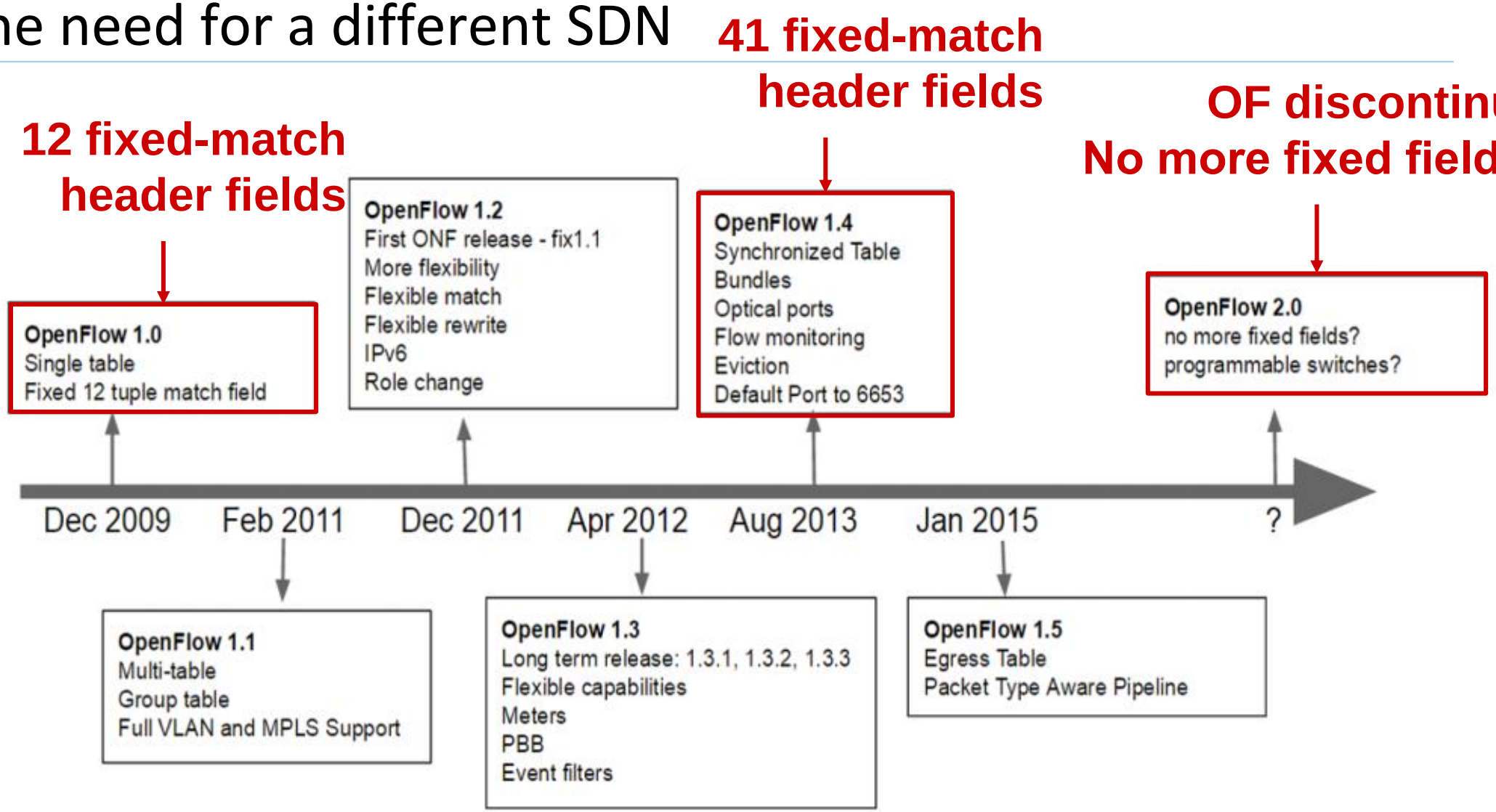


***Figure source:** https://kspviswa.github.io/OpenFlow_Version_Roadmap.html

© ETSI CC-BY-4.0

The need for a different SDN

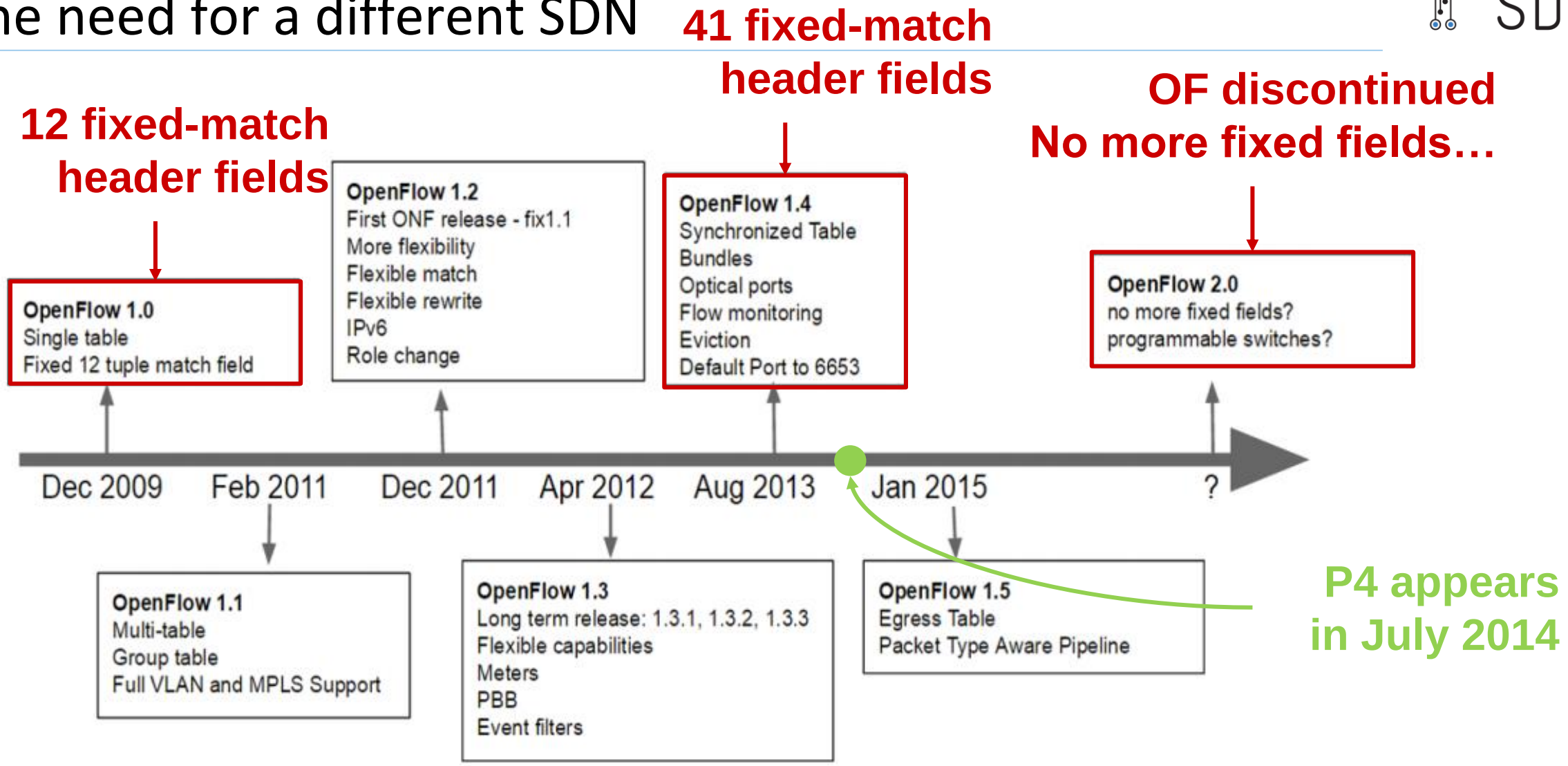


***Figure source:** https://kspviswa.github.io/OpenFlow_Version_Roadmap.html

© ETSI CC-BY-4.0

The sad reality about OpenFlow

Specification	v1.4	41 fixed-match header fields	17 action types

© ETSI CC-BY-4.0

The sad reality about OpenFlow

TeraFlow SDN

Specification	 v1.4	41 fixed-match header fields	17 action types

In reality

Most hardware switches only support a limited match/action set due to ASIC limitations

Hardware re-design requires long development cycles and increases cost

© ETSI CC-BY-4.0

Why P4?

P4 motivation: Instead of repeatedly extending the OpenFlow standards, let's define a whole new abstraction for programming the data plane

© ETSI CC-BY-4.0

Why P4?

P4 motivation: Instead of repeatedly extending the OpenFlow standards, let's define a whole new abstraction for programming the data plane

P4 principles:

- a domain-specific programming language to formally define the data plane pipeline
  - Describes proto headers (existing+new), lookup tables, actions, counters, etc.
  - Describes both fast (ASIC, FPGA) and slow (e.g., soft. switch) pipelines

© ETSI CC-BY-4.0

Why P4?

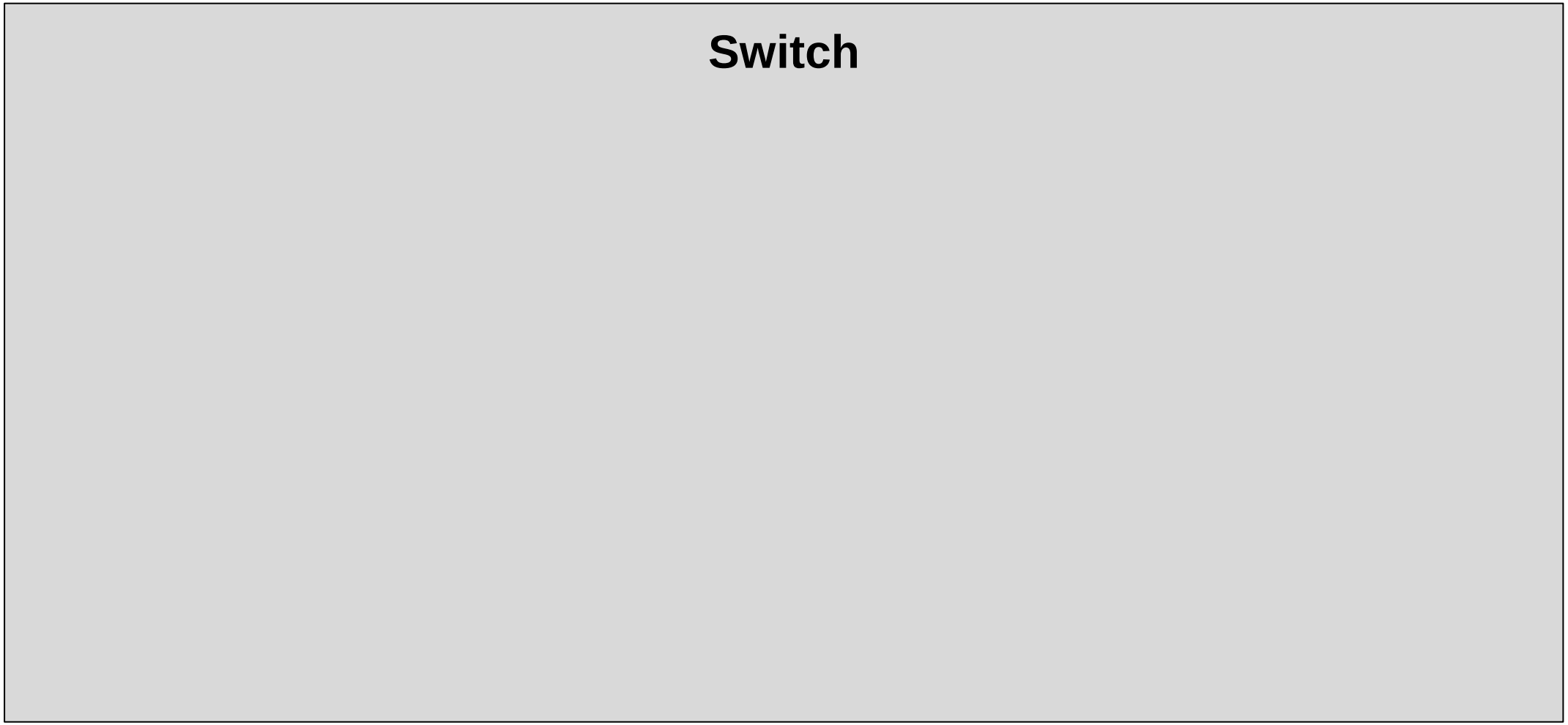
P4 motivation: Instead of repeatedly extending the OpenFlow standards, let's define a whole new abstraction for programming the data plane

P4 principles:

- a domain-specific programming language to formally define the data plane pipeline
  - Describes proto headers (existing+new), lookup tables, actions, counters, etc.
  - Describes both fast (ASIC, FPGA) and slow (e.g., soft. switch) pipelines
- a common interface to parse packets and match (arbitrary) header fields
  - Defines a "contract" between the control and data plane

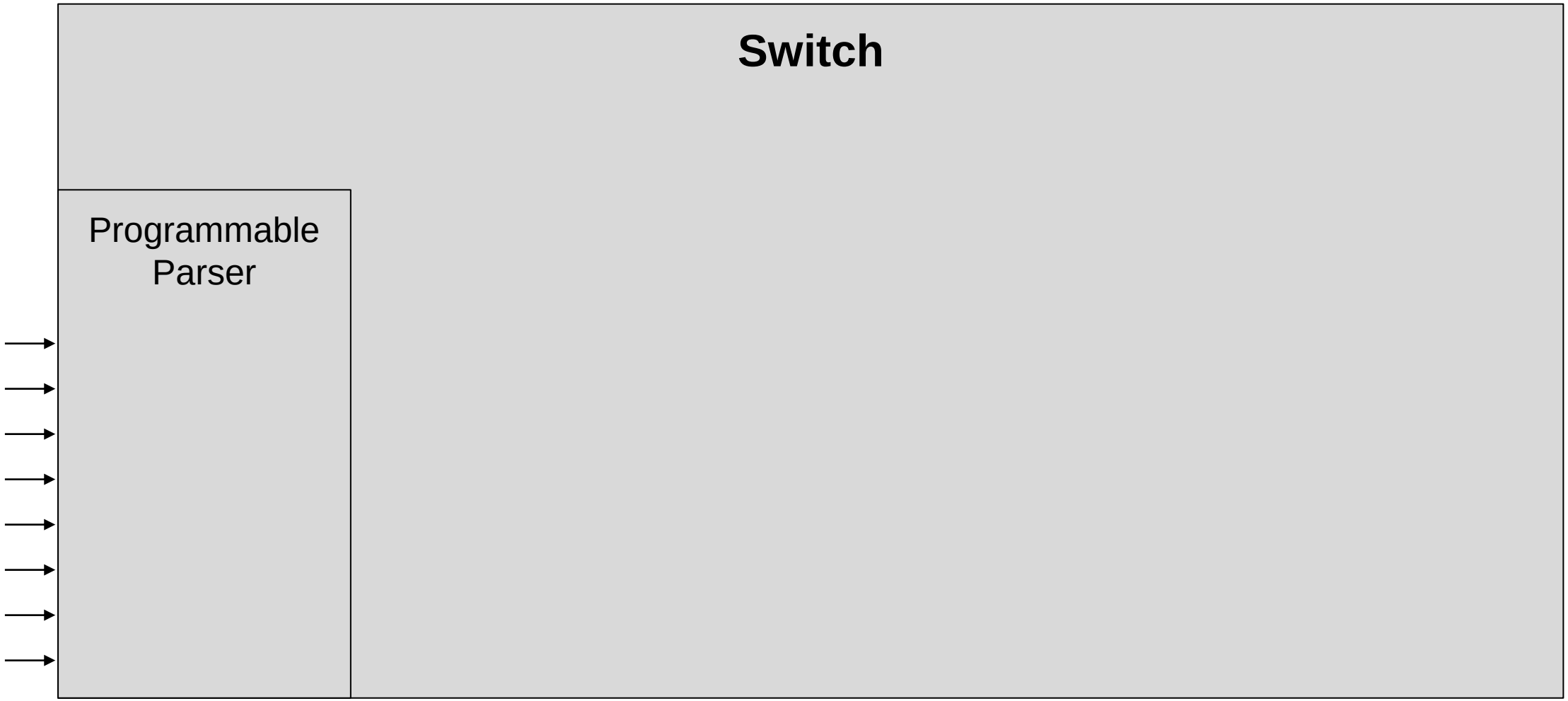
© ETSI CC-BY-4.0

PISA: Protocol-Independent Switch Architecture



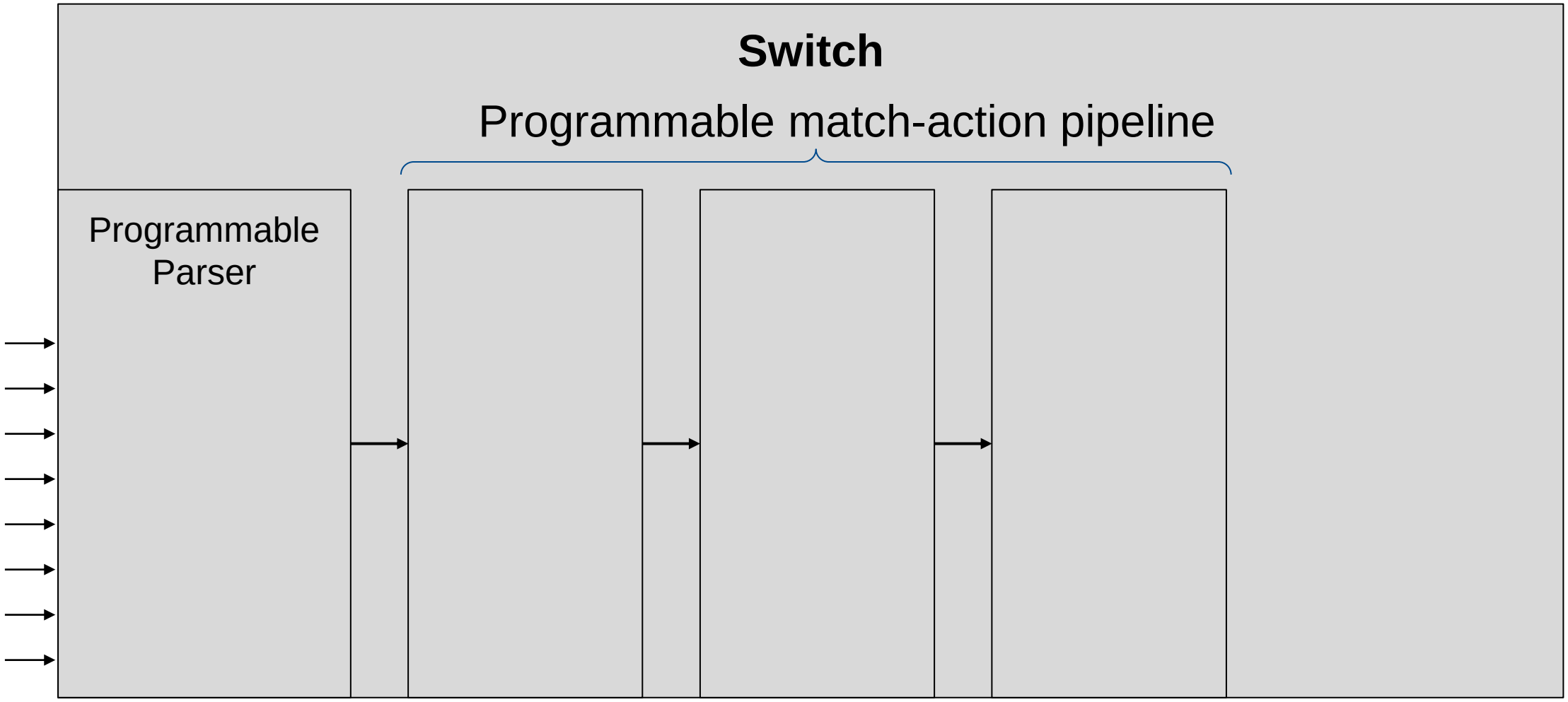
© ETSI CC-BY-4.0

PISA: Protocol-Independent Switch Architecture



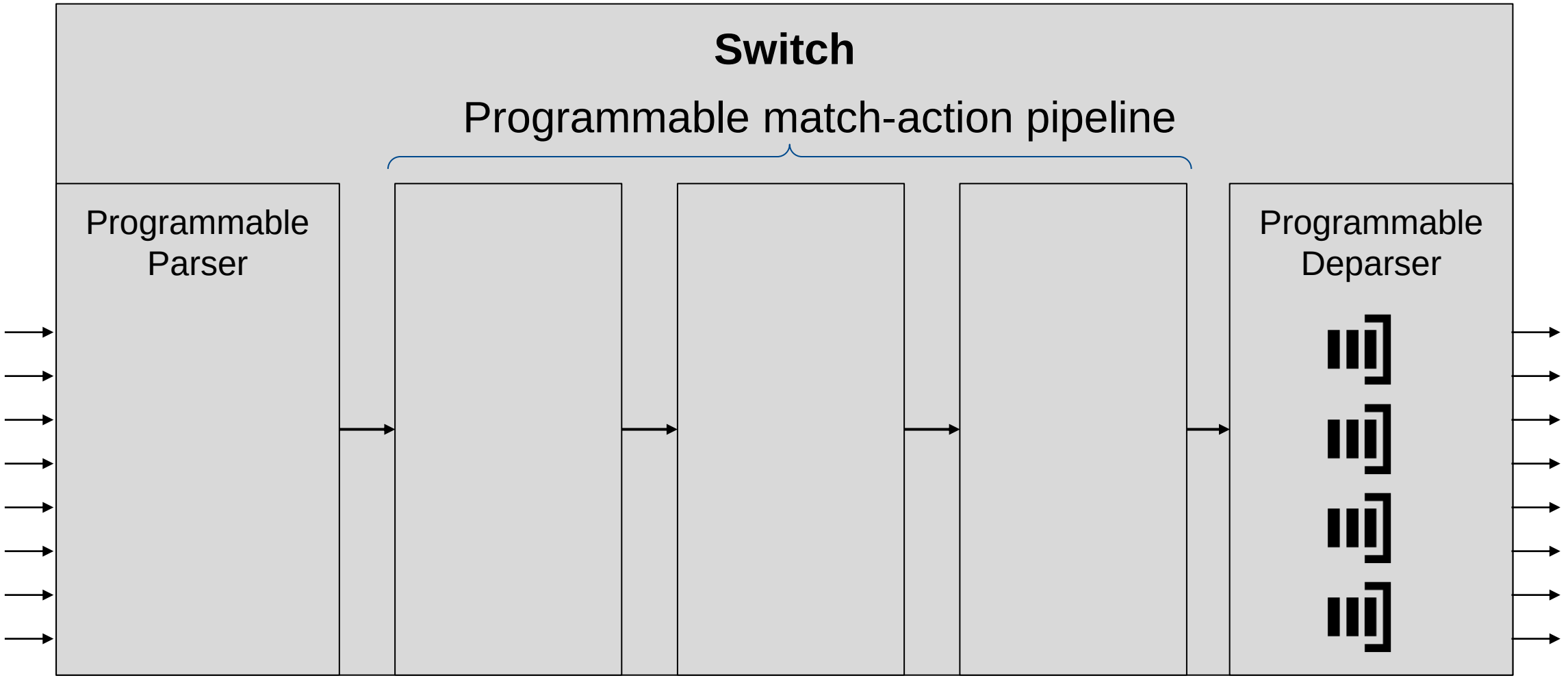
© ETSI CC-BY-4.0

PISA: Protocol-Independent Switch Architecture



© ETSI CC-BY-4.0

PISA: Protocol-Independent Switch Architecture

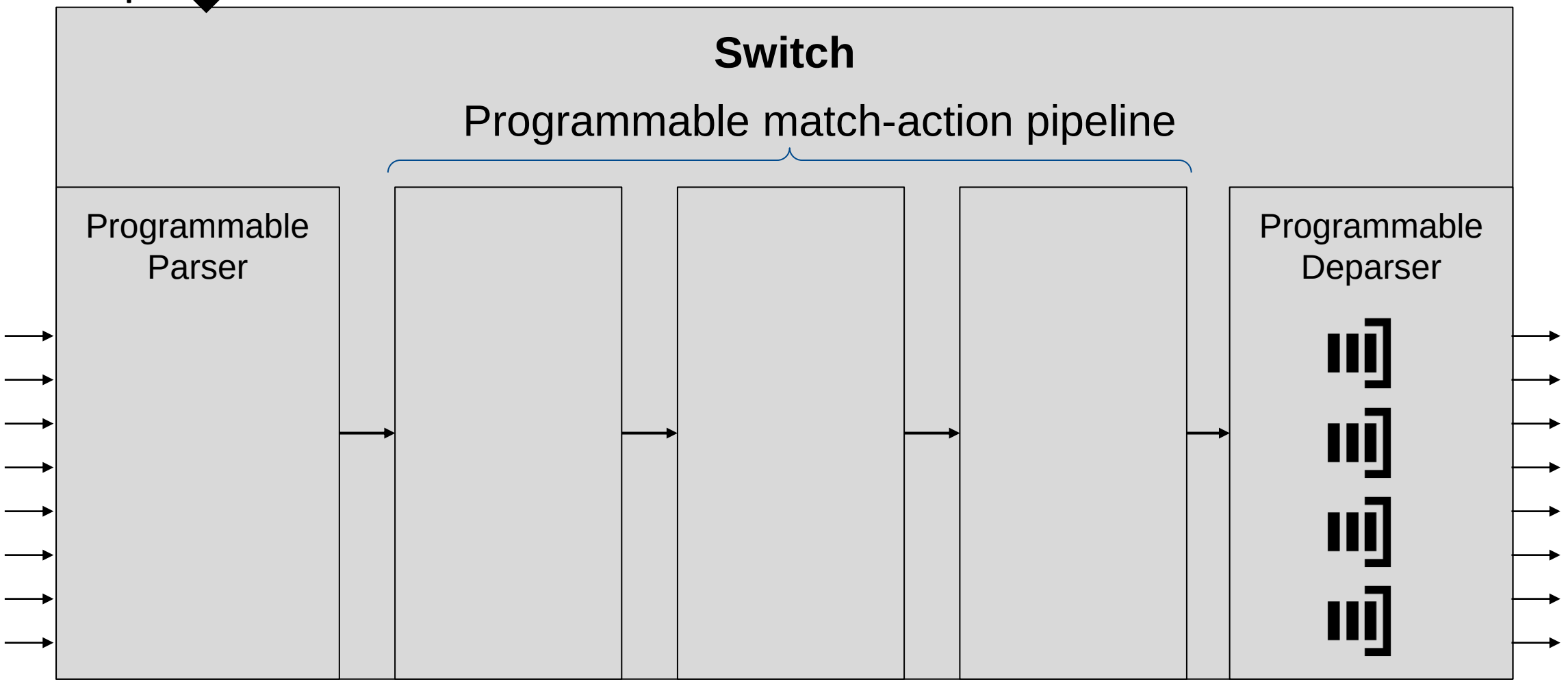


© ETSI CC-BY-4.0

PISA: Protocol-Independent Switch Architecture

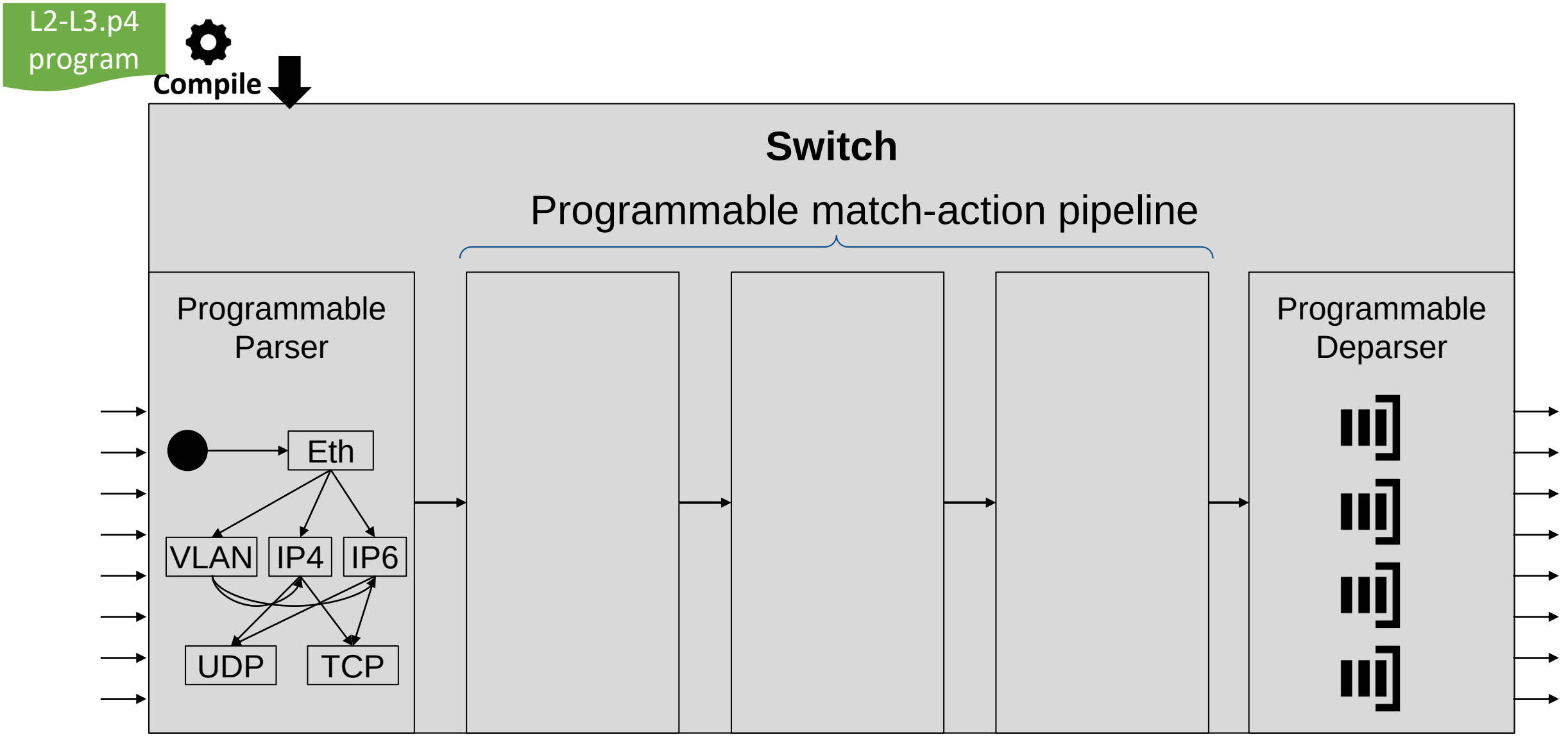
L2-L3.p4 program

Compile



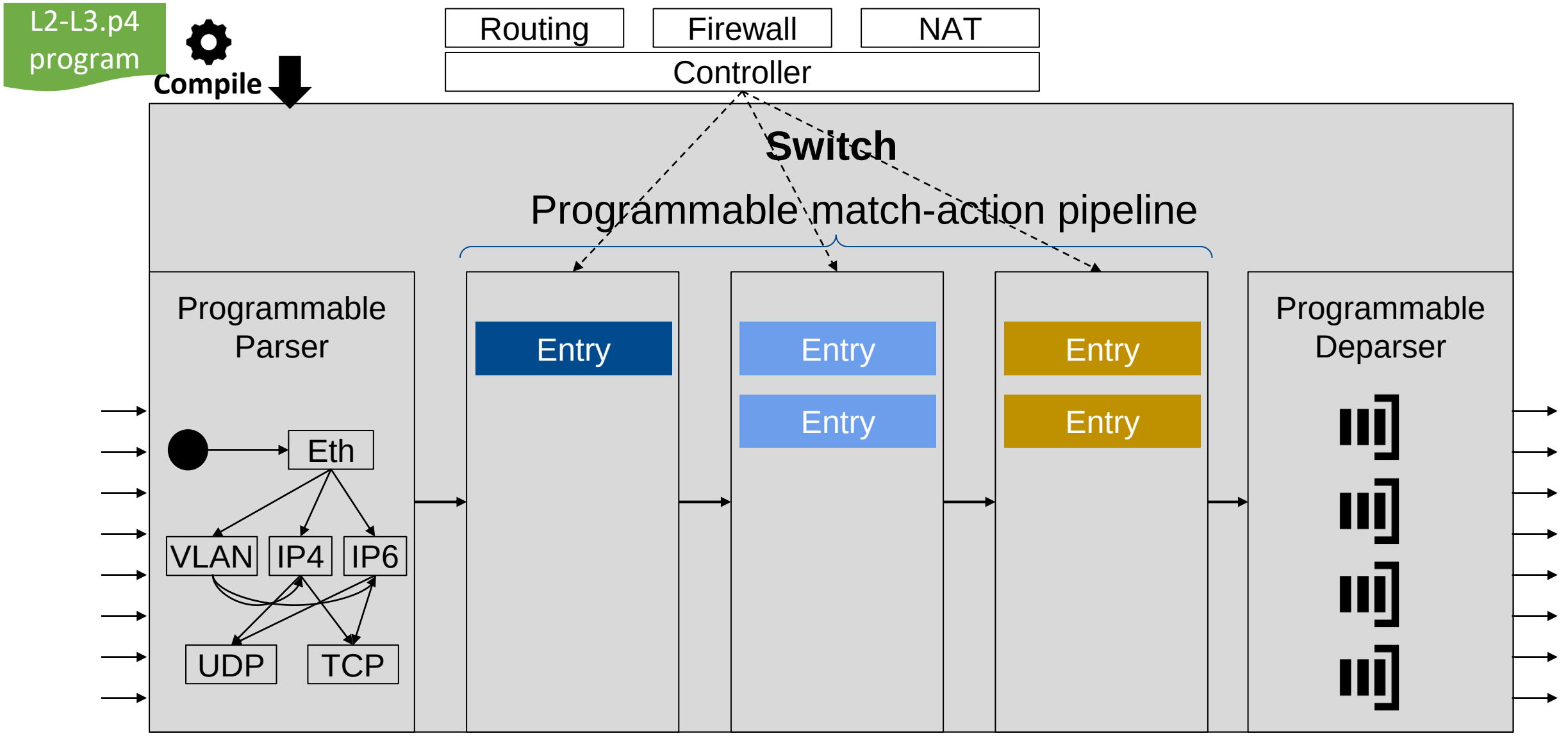
© ETSI CC-BY-4.0

PISA: Protocol-Independent Switch Architecture



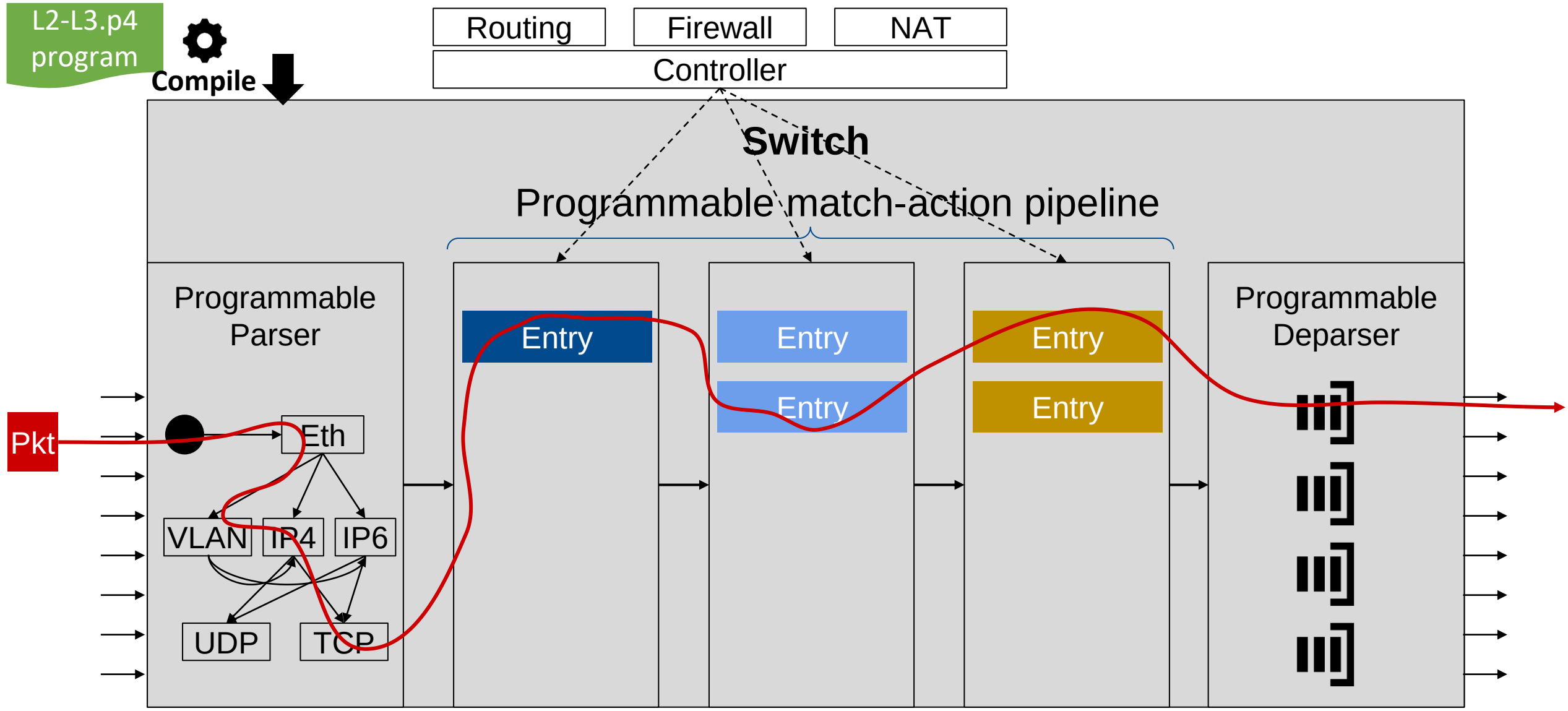
© ETSI CC-BY-4.0

PISA: Protocol-Independent Switch Architecture



© ETSI CC-BY-4.0

PISA: Protocol-Independent Switch Architecture



© ETSI CC-BY-4.0

But switches still have ASICs…

Correct, but…

© ETSI CC-BY-4.0

But switches still have ASICs…

Correct, but…

- New custom ASICs offer decent flexibility even at several Tera bits/sec

© ETSI CC-BY-4.0

But switches still have ASICs…

Correct, but…

- New custom ASICs offer decent flexibility even at several Tera bits/sec
- Some switches offer higher programmability than others:
  - FPGAs (e.g., Intel, Xilinx)
  - NPUs (e.g., Netronome, EZchip)
  - Software-based (e.g., OVS)

© ETSI CC-BY-4.0

P4Runtime: Not just yet another P4 API?

P4Runtime is a runtime control API for P4-defined data planes

© ETSI CC-BY-4.0

P4Runtime: Not just yet another P4 API?

P4Runtime is a runtime control API for P4-defined data planes

API	Target independent Same API works with different switches from different vendors
OpenFlow	
P4Runtime	

© ETSI CC-BY-4.0

P4Runtime: Not just yet another P4 API?

P4Runtime is a runtime control API for P4-defined data planes

API	Target independent Same API works with different switches from different vendors	Pipeline independent Same API allows control of many arbitrary pipelines
OpenFlow	✓	✓ With table type patterns (TTP)
P4Runtime	✓	✓ With P4

© ETSI CC-BY-4.0

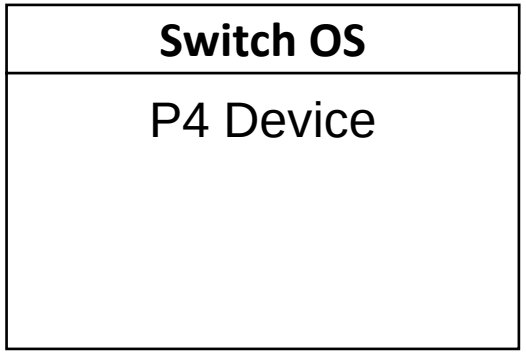
P4Runtime: Not just yet another P4 API?

P4Runtime is a runtime control API for P4-defined data planes

API	Target independent Same API works with different switches from different vendors	Pipeline independent Same API allows control of many arbitrary pipelines	Protocol independent Same API allows control of any data plane proto (standard/custom)
OpenFlow	✓	✓ With table type patterns (TTP)	✗ Proto headers and actions hardcoded in the spec
P4Runtime	✓	✓ With P4	✓

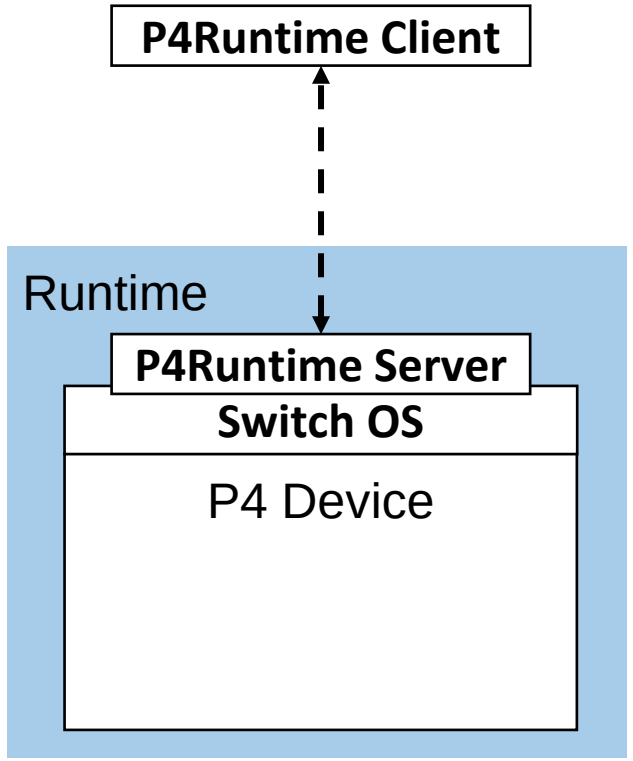
© ETSI CC-BY-4.0

P4 workflow



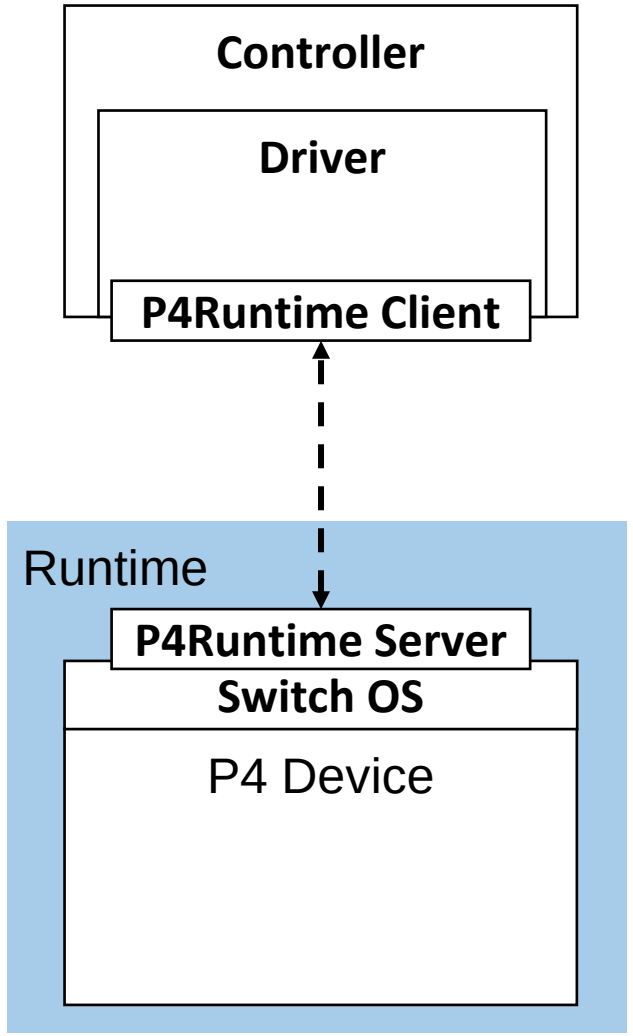
© ETSI CC-BY-4.0

P4 workflow



© ETSI CC-BY-4.0

P4 workflow

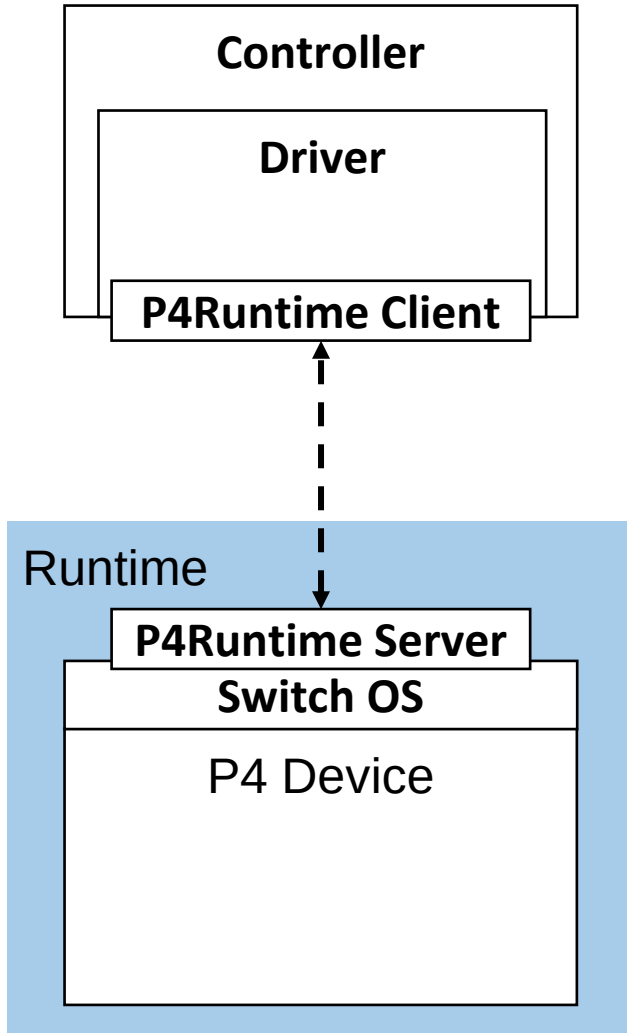


© ETSI CC-BY-4.0

P4 workflow

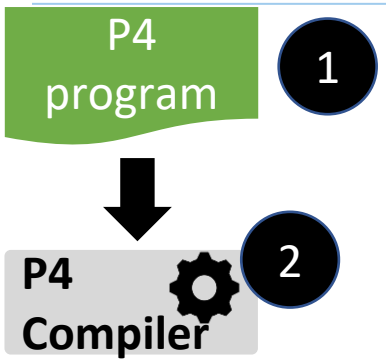
P4 program

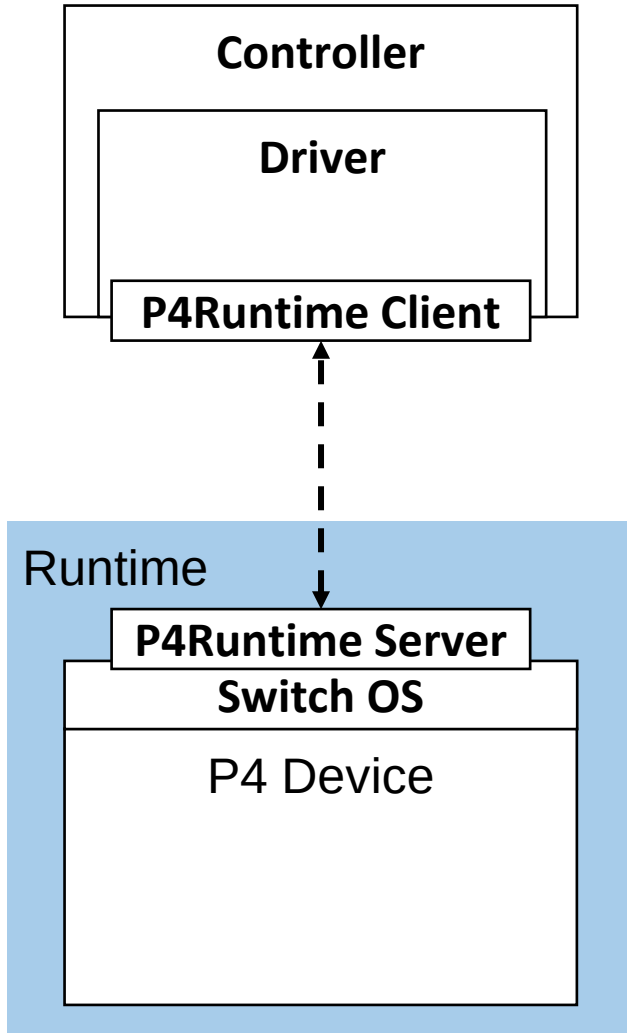
1



© ETSI CC-BY-4.0

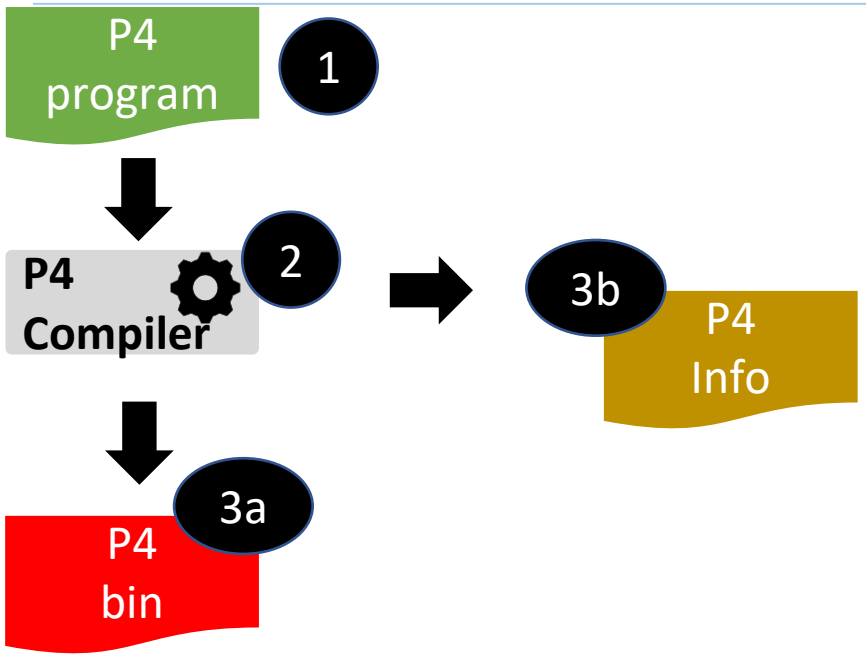
P4 workflow

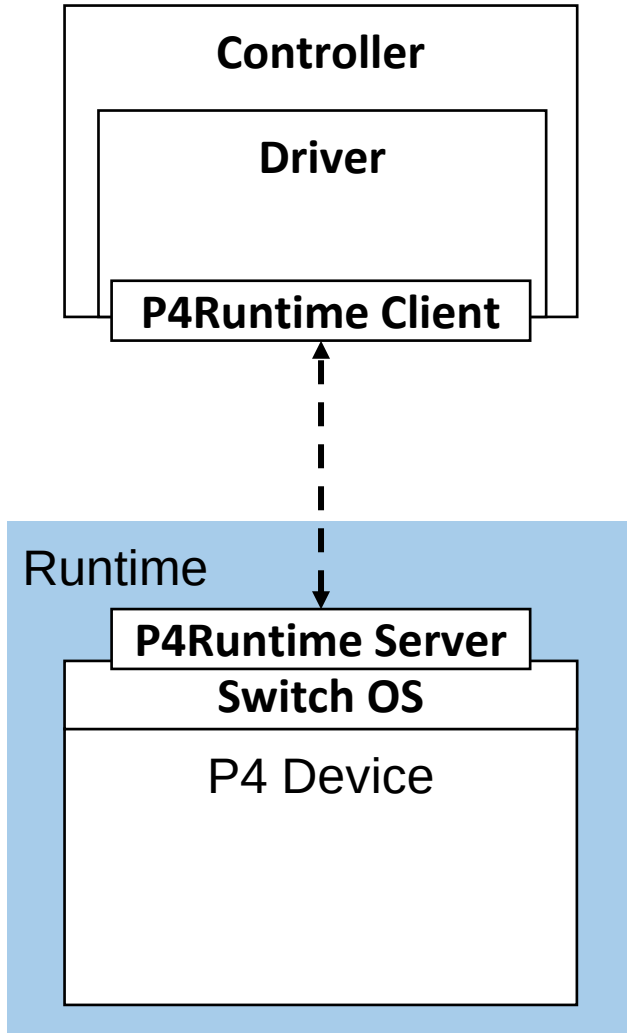




© ETSI CC-BY-4.0

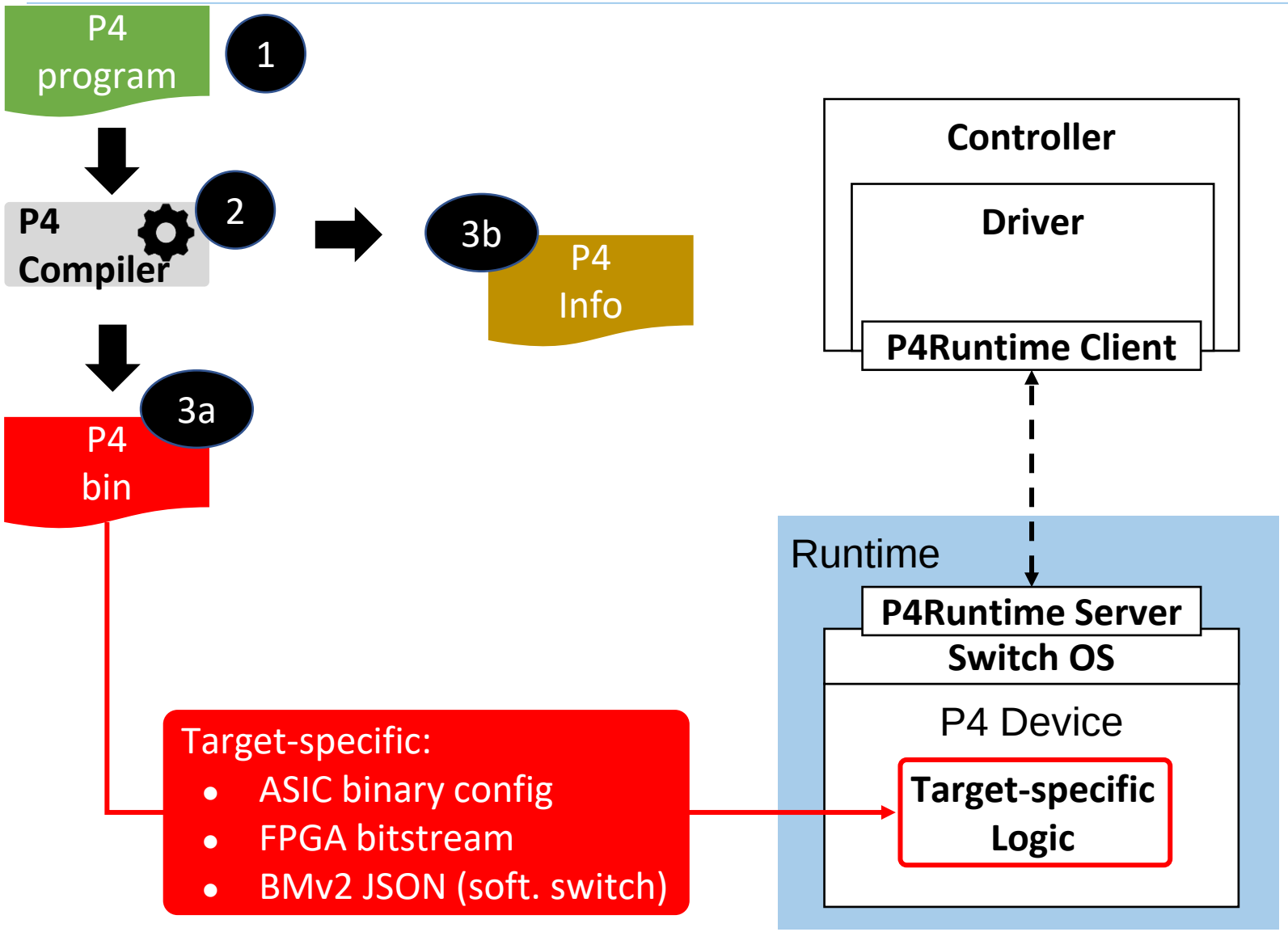
P4 workflow





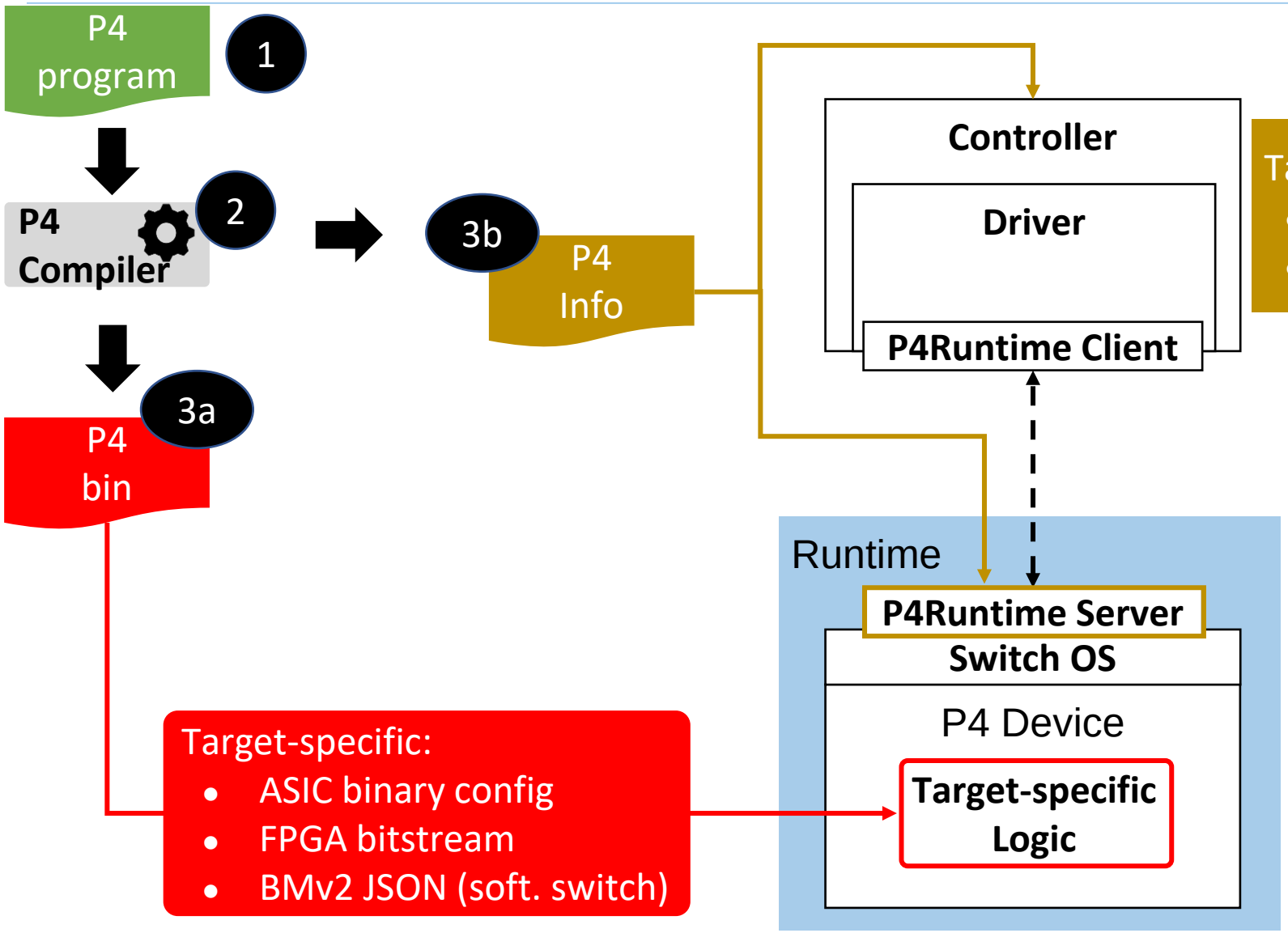
© ETSI CC-BY-4.0

P4 workflow



© ETSI CC-BY-4.0

P4 workflow



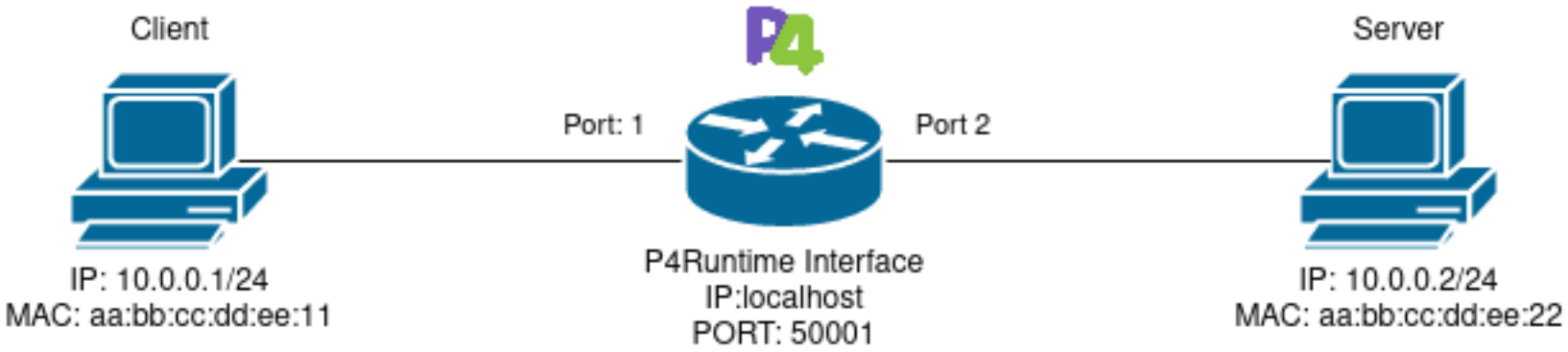
© ETSI CC-BY-4.0

Mini P4 Demonstration

© ETSI CC-BY-4.0

P4 Demonstration - Setup

This demonstration consists of a simple client/server setup, with two hosts interconnected via a software-based P4 switch



P4 Demonstration - Scenario

Objective: Realize a simple l2 forwarding program in p4

P4 Demonstration - Scenario

Objective: Realize a simple l2 forwarding program in p4

Approach: Bi-directional ICMP connectivity from client to server

- Use mininet to emulate the network
- Static ARP entries in client and server (for the needs of this simple demo)
- Client sends an ICMP echo-request to the server
- Server replies to the client with an ICMP echo-reply

P4 Demonstration - L2 forwarding program

Types definitions:

```
typedef bit<9>   port_num_t;
typedef bit<48>  mac_addr_t;
```

l2 headers (ethernet):

```
header ethernet_t {
       mac_addr_t   dst_addr;
       mac_addr_t   src_addr;
       bit<16>
       ether_type;
}
```

Parser:

```
state start {
       transition parse_ethernet;
}

state parse_ethernet {
       packet.extract(hdr.ethernet);
       transition accept;
}
```

P4 Demonstration - L2 forwarding program

Actions:

```
action drop() {
  mark_to_drop(standard_metadata);
}

action set_egress_port(port_num_t
port_num) {
  standard_metadata.egress_spec =

      port_num;
}
```

Table:

```
table l2_exact_table {
  key = {
    hdr.ethernet.dst_addr: exact;
  }
    actions = {
      set_egress_port;
      @defaultonly drop;
    }
      const default_action = drop;
}
```

P4 Demonstration - L2 forwarding program

Actions:

```
action drop() {
  mark_to_drop(standard_metadata);
}

action set_egress_port(port_num_t
port_num) {
  standard_metadata.egress_spec =

      port_num;
}
```

Table:

```
table l2_exact_table {
  key = {
    hdr.ethernet.dst_addr: exact;
  }
  actions = {
    set_egress_port;
    @defaultonly drop;
  }
    const default_action = drop;
}
```

P4 Demonstration - Device details

The TeraFlowSDN controller connects to a device as follows:

```
DEVICE_SW1_CONNECT_RULES = json_device_connect_rules(
        DEVICE_SW1_IP_ADDR,                             # localhost
        DEVICE_SW1_PORT,                                # 50001
        {
                    'id': DEVICE_SW1_DPID,              # 1
                  'name': DEVICE_SW1_NAME,              # SW1
                'vendor': DEVICE_SW1_VENDOR,            # ONF
                'hw_ver': DEVICE_SW1_HW_VER,            # BMv2 simple_switch
                'sw_ver': DEVICE_SW1_SW_VER,            # Stratum
               'timeout': DEVICE_SW1_TIMEOUT,           # 60
                 'p4bin':      DEVICE_SW1_BIN_PATH,     # P4 binary in the device pod
                'p4info': DEVICE_SW1_INFO_PATH          # P4 info in the device pod
        }
)
```

P4 Demonstration - Rules

P4 Rules are composed based on the p4info.txt:

```
'table-name': 'IngressPipeImpl.l2_exact_table',
  'match-fields': [
      {
            'match-field': 'hdr.ethernet.dst_addr',
            'match-value': 'aa:bb:cc:dd:ee:11'
      }
  ],
  'action-name': 'IngressPipeImpl.set_egress_port',
  'action-params': [
      {
            'action-param': 'port_num',
            'action-value': '1'
      }
  ]
}
```

P4 Demonstration - Rules

P4 Rules are composed based on the p4info.txt:

```
'table-name': 'IngressPipeImpl.l2_exact_table',
  'match-fields': [
      {
            'match-field': 'hdr.ethernet.dst_addr',
            'match-value': 'aa:bb:cc:dd:ee:11'
      }
  ],
  'action-name': 'IngressPipeImpl.set_egress_port',
  'action-params': [
      {
            'action-param': 'port_num',
            'action-value': '1'
      }
  ]
}
```

Table to insert rule

P4 Demonstration - Rules

P4 Rules are composed based on the p4info.txt:

```
'table-name': 'IngressPipeImpl.l2_exact_table',
  'match-fields': [
      {
          'match-field': 'hdr.ethernet.dst_addr',
          'match-value': 'aa:bb:cc:dd:ee:11'
      }
  ],
  'action-name': 'IngressPipeImpl.set_egress_port',
  'action-params': [
      {
          'action-param': 'port_num',
          'action-value': '1'
      }
  ]
}
```

Table to insert rule

Match client's destination MAC address (Table key)

P4 Demonstration - Rules

P4 Rules are composed based on the p4info.txt:

```
'table-name': 'IngressPipeImpl.l2_exact_table',
  'match-fields': [
      {
          'match-field': 'hdr.ethernet.dst_addr',
          'match-value': 'aa:bb:cc:dd:ee:11'
      }
  ],
  'action-name': 'IngressPipeImpl.set_egress_port',
  'action-params': [
      {
          'action-param': 'port_num',
          'action-value': '1'
      }
  ]
}
```

Table to insert rule

Match client's destination MAC address (Table key)

Action to call

Parameter value for the action

P4 Demonstration - Step 1

- Run mininet:

```
cd ~/ngsdn-tutorial
make start-simple
make mn-cli
```

Expected Results:

- mininet starts with the correct topology
  - Run nodes to check

P4 Demonstration - Step 1

- Run mininet:

```
cd ~/ngsdn-tutorial
make start-simple
make mn-cli
```

- Try to ping using the mininet CLI:

```
client ping server
```

- Let it run

Expected Results:

- mininet starts with the correct topology
  - Run nodes to check
- We do not get any reply on client

P4 Demonstration - Step 2

On another terminal:

- Run setup script: `src/tests/netx22-p4/setup.sh`

Expected Results:

- The p4 artifacts are copied to the device pod

P4 Demonstration - Step 2

On another terminal:

- Run setup script: `src/tests/netx22-p4/setup.sh`

Expected Results:

- The p4 artifacts are copied to the device pod

- Run bootstrap script `src/tests/netx22-p4/run_test_01_bootstrap.sh`

- The device should be registered on the TeraFlowSDN UI

P4 Demonstration - Step 3

- Run create service script:
  `src/tests/netx22-p4/run_test_02_create_service.sh`

Expected Results:

- Server should reply
  - Check mininet terminal

P4 Demonstration - Step 3

- Run create service script:
  `src/tests/netx22-p4/run_test_02_create_service.sh`

Expected Results:

- Server should reply
  - Check mininet terminal

- Run delete service script
  `src/tests/netx22-p4/run_test_03_delete_service.sh`

- Server should stop replying
  - Check mininet terminal

P4 Demonstration - Step 4

- Retry the previous step

Expected Results:

- Server should begin replying
  - Check that the `icmp_seq` has advanced

P4 Demonstration - Step 4

- Retry the previous step

Expected Results:

- Server should begin replying
  - Check that the `icmp_seq` has advanced

- Run cleanup script `src/tests/netx22-p4/run_test_04_cleanup.sh`

- Device is no longer registered to the TeraFlowSDN UI

P4 Demonstration - Discussion

- For the needs of this demo ARP entries are hardcoded

P4 Demonstration - Discussion

- For the needs of this demo ARP entries are hardcoded
- How would we implement ARP in P4?
  - Insert separate entries for each port?
    - Really bad idea, does not scale!

P4 Demonstration - Discussion

- For the needs of this demo ARP entries are hardcoded
- How would we implement ARP in P4?
  - Insert separate entries for each port?
    - Really bad idea, does not scale!
  - **Use native P4 multicast group IDs**

P4 Demonstration - ARP

Actions:

```
action set_multicast_group(
                mcast_group_id_t gid)
{
  standard_metadata.mcast_grp =  gid;
}
```

Table:

```
table l2_exact_table {
  key = {
    hdr.ethernet.dst_addr: exact;
  }
    actions = {
      set_egress_port;
      set_multicast_group;
      @defaultonly drop;
    }
    const default_action = drop;
}
```

P4 Demonstration - ARP

P4Runtime Rule:

```
multicast_group_entry {
      multicast_group_id: 1
      replicas {
            egress_port: 1
      }
      replicas {
            egress_port: 2
      }
}
```

TeraFlow SDN by ETSI

Thank You!

© ETSI CC-BY-4.0