UNIVERSITY OF TWENTE.

# P4 (Programming Protocol-independent Packet Processors) Lab

Advanced Networking

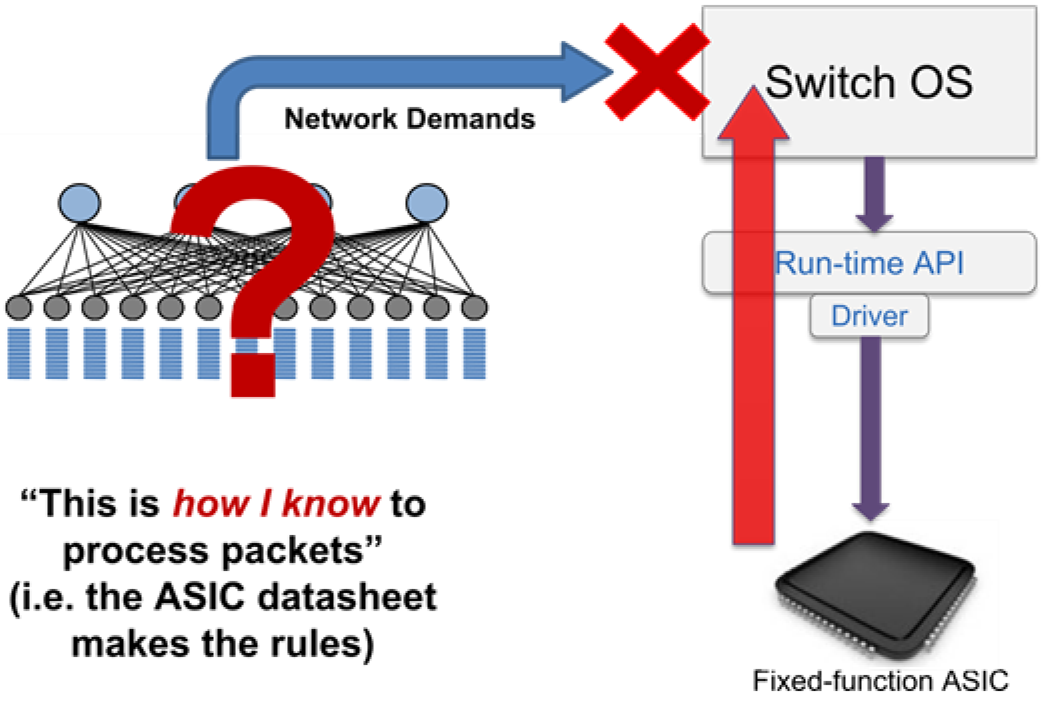
Friday Sep 5, 2025

## P4 Lab
Teaching Assistant

Shyam Krishna Khadka (s.k.khadka@utwente.nl)

- Most of the contents are from
  - P4 Language Consortium: "01 Introduction to Data Plane Programming (Stephen Ibanez)".

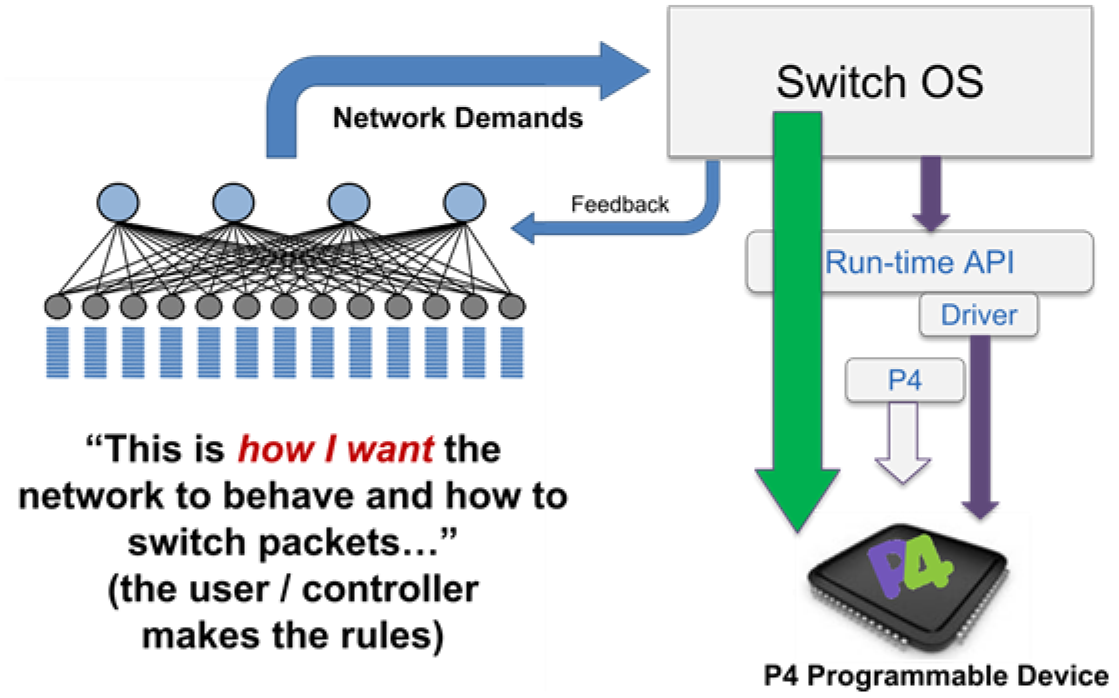
## Introduction
Status Quo: Bottom-up



"This is *how I know* to process packets"
(i.e. the ASIC datasheet makes the rules)

## Introduction

A Better Approach: Top-down design



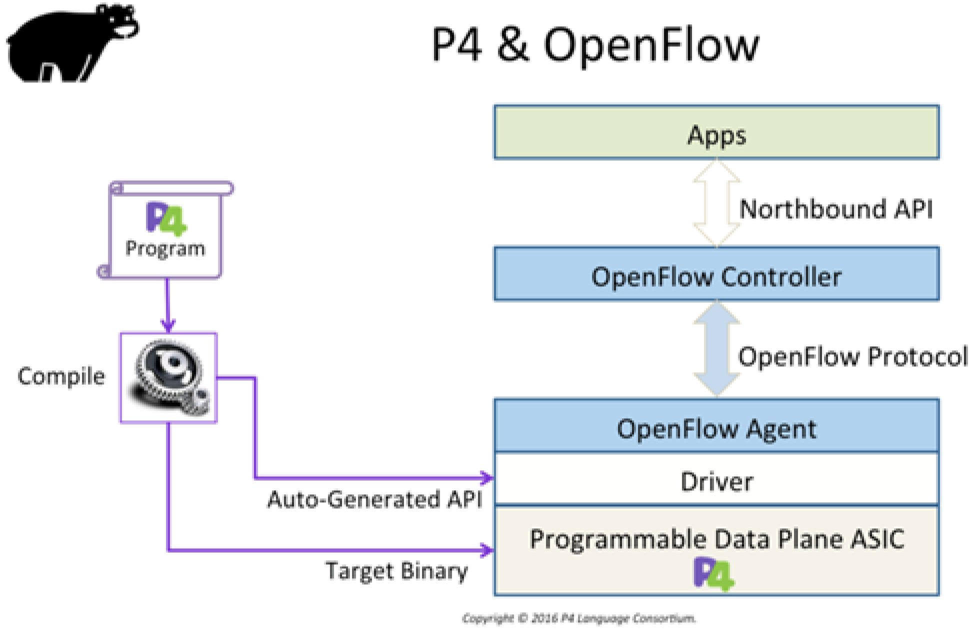
**“This is *how I want* the network to behave and how to switch packets…” (the user / controller makes the rules)**

*p4.org*

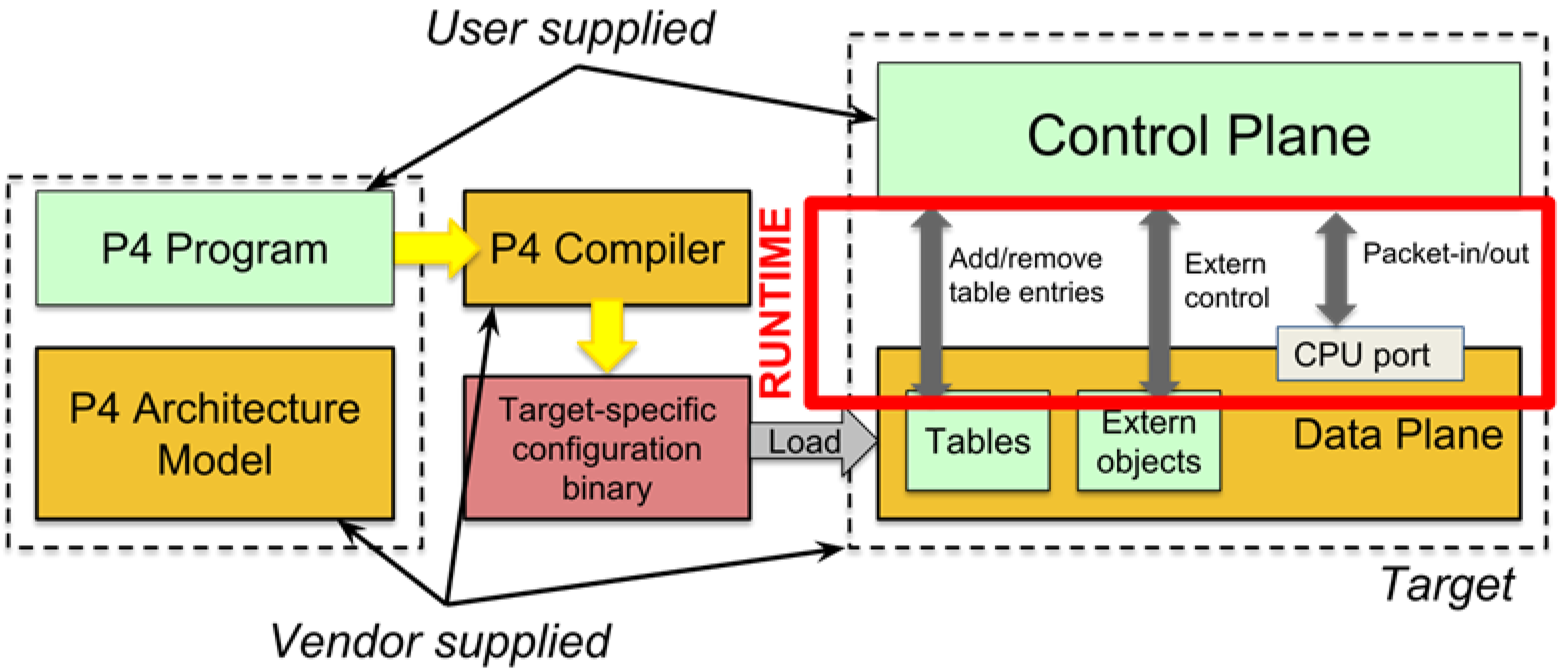
UNIVERSITY OF TWENTE.

## Architectural Overview

Control Plane & Data Plane

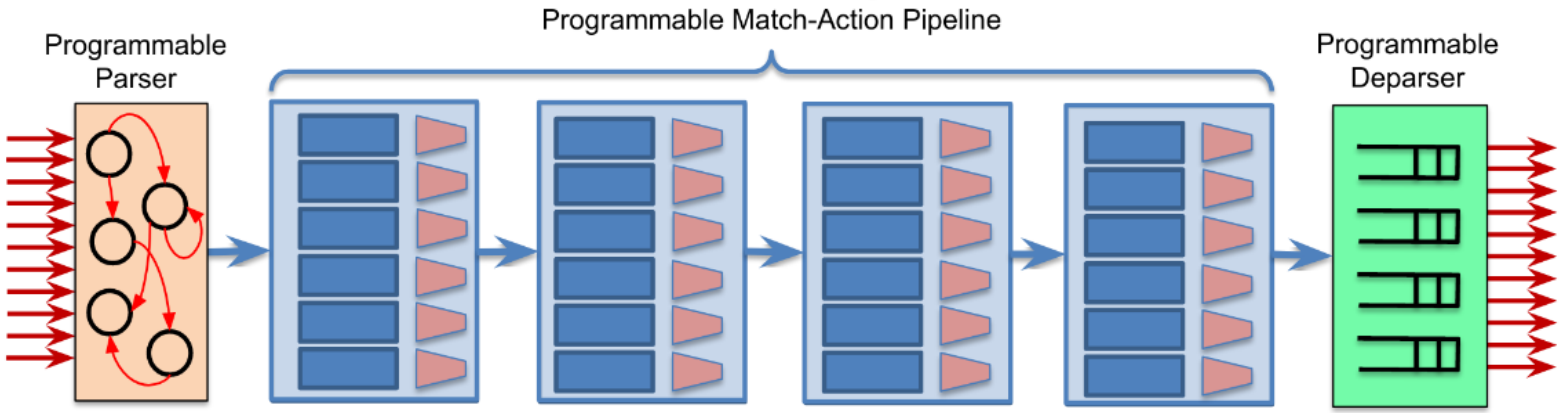


## Architectural Overview
P4



## Architectural Overview

PISA: Protocol Independent Switch Architecture (e.g. Intel Tofino Switch)

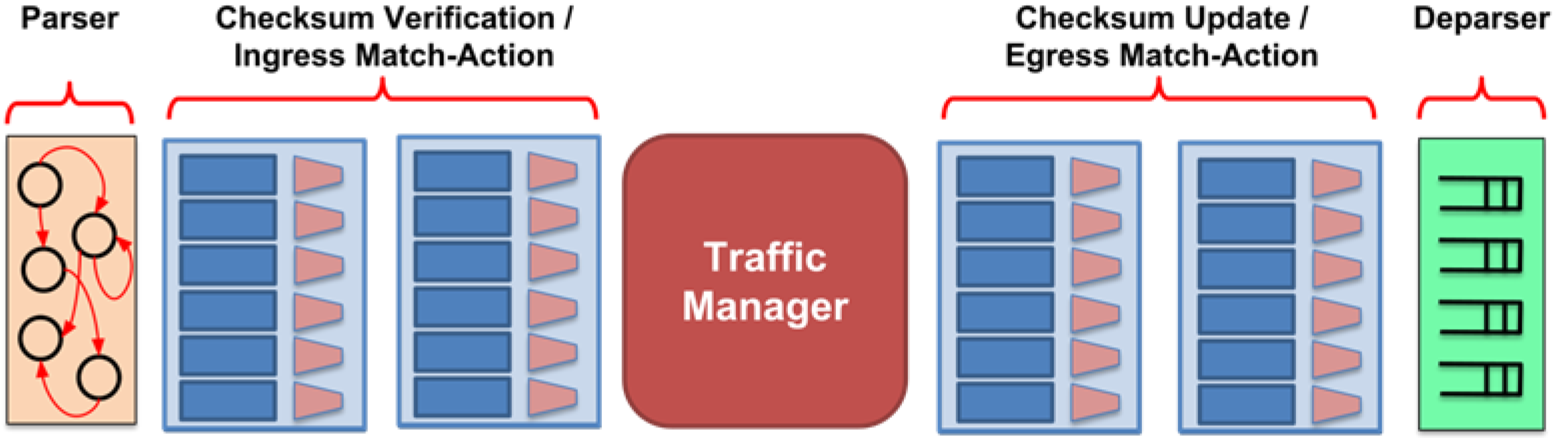


## Architectural Overview
bmv2 Switch

- Software switch: P4 **Target**
- Used in the lab assignments
- For developing, testing and debugging
- V1Model: P4 **Architecture** model for bmv2

## Architectural Overview

V1Model stages


Parser
Checksum Verification / Ingress Match-Action
Traffic Manager
Checksum Update / Egress Match-Action
Deparser

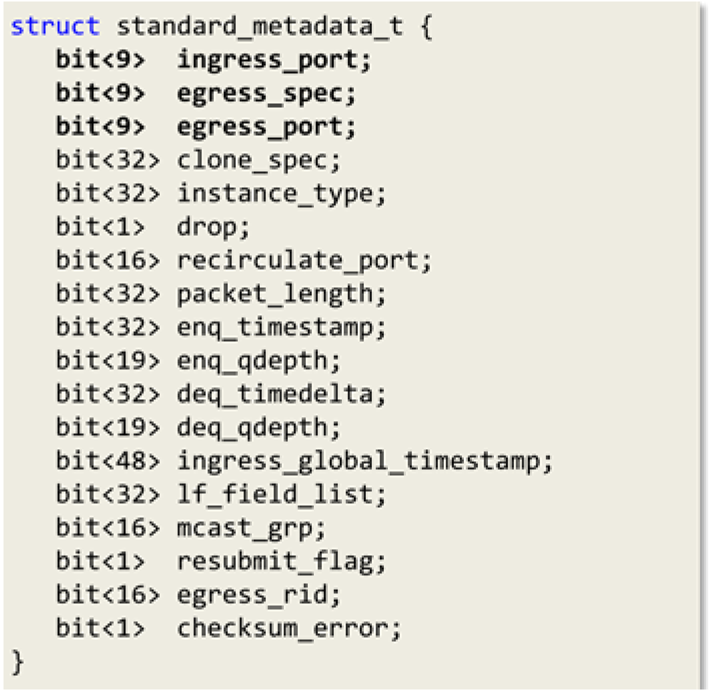
## Programming in P4
V1Model stages

```
#include <core.p4>
#include <v1model.p4>
/* HEADERS */
struct metadata { ... }
struct headers {
  ethernet_t   ethernet;
  ipv4_t       ipv4;
}
/* PARSER */
parser MyParser(packet_in packet,
                out headers hdr,
                inout metadata meta,
                inout standard_metadata_t smeta) {
  ...
}
/* CHECKSUM VERIFICATION */
control MyVerifyChecksum(in headers hdr,
                         inout metadata meta) {
   ...
}
/* INGRESS PROCESSING */
control MyIngress(inout headers hdr,
                  inout metadata meta,
                  inout standard_metadata_t std_meta) {
  ...
}
```

```
/* EGRESS PROCESSING */
control MyEgress(inout headers hdr,
                 inout metadata meta,
                 inout standard_metadata_t std_meta) {
  ...
}
/* CHECKSUM UPDATE */
control MyComputeChecksum(inout headers hdr,
                          inout metadata meta) {
  ...
}
/* DEPARSER */
control MyDeparser(inout headers hdr,
                   inout metadata meta) {
  ...
}
/* SWITCH */
V1Switch(
  MyParser(),
  MyVerifyChecksum(),
  MyIngress(),
  MyEgress(),
  MyComputeChecksum(),
  MyDeparser()
) main;
```

UNIVERSITY OF TWENTE.

## Programming in P4
### Metadata and V1Model Standard Metadata



```
struct standard_metadata_t {
   bit<9>  ingress_port;
   bit<9>  egress_spec;
   bit<9>  egress_port;
   bit<32> clone_spec;
   bit<32> instance_type;
   bit<1>  drop;
   bit<16> recirculate_port;
   bit<32> packet_length;
   bit<32> enq_timestamp;
   bit<19> enq_qdepth;
   bit<32> deq_timedelta;
   bit<19> deq_qdepth;
   bit<48> ingress_global_timestamp;
   bit<32> lf_field_list;
   bit<16> mcast_grp;
   bit<1>  resubmit_flag;
   bit<16> egress_rid;
   bit<1>  checksum_error;
}
```

- **`ingress_port`** - the port on which the packet arrived
- **`egress_spec`** - the port to which the packet should be sent to
- **`egress_port`** - the port on which the packet is departing from (read only in egress pipeline)

# Programming in P4
Basic example

```
#include <core.p4>
#include <v1model.p4>
struct metadata {}
struct headers {}

parser MyParser(packet_in packet,
   out headers hdr,
   inout metadata meta,
   inout standard_metadata_t standard_metadata) {

    state start { transition accept; }
}

control MyVerifyChecksum(inout headers hdr, inout metadata
meta) {   apply {  }   }

control MyIngress(inout headers hdr,
   inout metadata meta,
   inout standard_metadata_t standard_metadata) {
apply {
        if (standard_metadata.ingress_port == 1) {
            standard_metadata.egress_spec = 2;
        } else if (standard_metadata.ingress_port == 2) {
            standard_metadata.egress_spec = 1;
        }
    }
}
```

```
control MyEgress(inout headers hdr,
   inout metadata meta,
   inout standard_metadata_t standard_metadata) {
    apply {  }
}

control MyComputeChecksum(inout headers hdr, inout metadata
meta) {
     apply { }
}

control MyDeparser(packet_out packet, in headers hdr) {
    apply { }
}

V1Switch(
   MyParser(),
   MyVerifyChecksum(),
   MyIngress(),
   MyEgress(),
   MyComputeChecksum(),
   MyDeparser()
) main;
```

## Programming in P4
Basic example elaborated.

```
#include <core.p4>
#include <v1model.p4>
struct metadata {}
struct headers {}

parser MyParser(packet_in packet, out headers hdr,
   inout metadata meta,
   inout standard_metadata_t standard_metadata) {
    state start { transition accept; }
}

control MyIngress(inout headers hdr, inout metadata meta,
   inout standard_metadata_t standard_metadata) {
    action set_egress_spec(bit<9> port) {
        standard_metadata.egress_spec = port;
    }
    table forward {
        key = { standard_metadata.ingress_port: exact; }
        actions = {
            set_egress_spec;
            NoAction;
        }
        size = 1024;
        default_action = NoAction();
    }
    apply {   forward.apply();   }
}
```

```
control MyEgress(inout headers hdr,
   inout metadata meta,
   inout standard_metadata_t standard_metadata) {
    apply {  }
}

control MyVerifyChecksum(inout headers hdr, inout metadata
meta) {   apply { }   }

control MyComputeChecksum(inout headers hdr, inout metadata
meta) {   apply { }   }

control MyDeparser(packet_out packet, in headers hdr) {
    apply { }
}

V1Switch( MyParser(), MyVerifyChecksum(), MyIngress(),
MyEgress(), MyComputeChecksum(), MyDeparser() ) main;
```

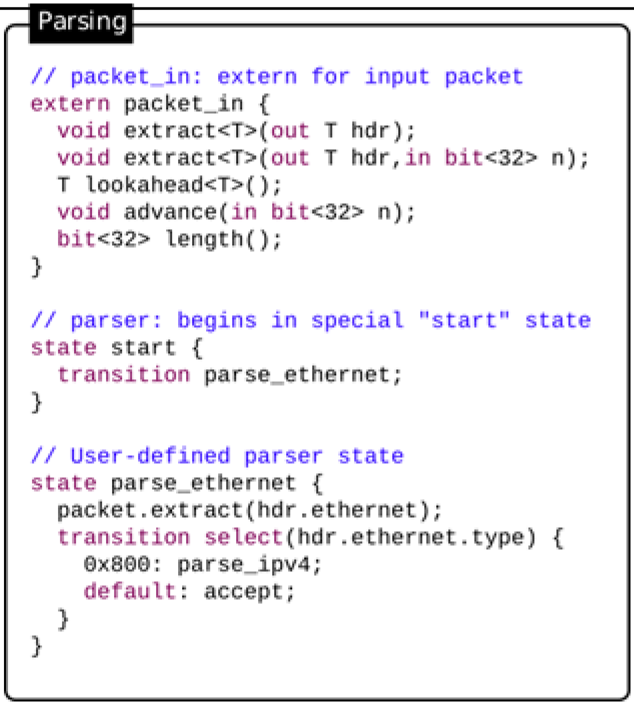
| Key | Action ID | Action Data |
|---|---|---|
| 1 | set_egress_spec ID | 2 |
| 2 | set_egress_spec ID | 1 |

UNIVERSITY OF TWENTE.

# Programming in P4

## Parsing

- `extern:` interface for functionality provided by switch vendor
  - Similar to abstract classes/methods in OOP
- State machine
- Transitions
  - `select`: change state
  - `accept`: finish parsing



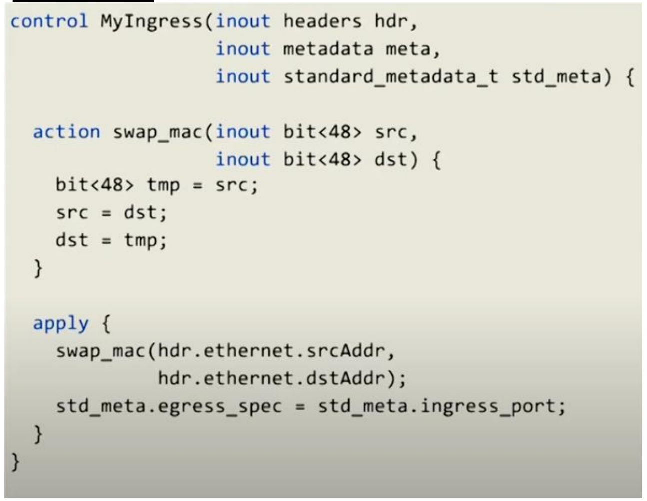
Parsing

```
// packet_in: extern for input packet
extern packet_in {
  void extract<T>(out T hdr);
  void extract<T>(out T hdr,in bit<32> n);
  T lookahead<T>();
  void advance(in bit<32> n);
  bit<32> length();
}

// parser: begins in special "start" state
state start {
  transition parse_ethernet;
}

// User-defined parser state
state parse_ethernet {
  packet.extract(hdr.ethernet);
  transition select(hdr.ethernet.type) {
    0x800: parse_ipv4;
    default: accept;
  }
}
```

# Programming in P4

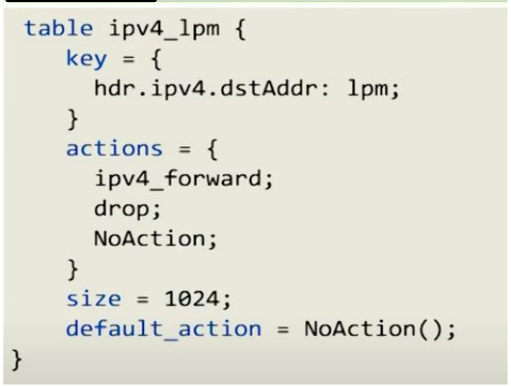
## Actions, Control Flow & Tables

### Action



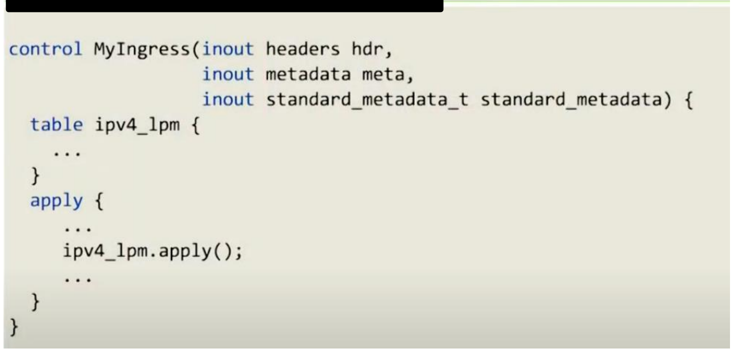
```
control MyIngress(inout headers hdr,
                  inout metadata meta,
                  inout standard_metadata_t std_meta) {

  action swap_mac(inout bit<48> src,
                  inout bit<48> dst) {
    bit<48> tmp = src;
    src = dst;
    dst = tmp;
  }

  apply {
    swap_mac(hdr.ethernet.srcAddr,
             hdr.ethernet.dstAddr);
    std_meta.egress_spec = std_meta.ingress_port;
  }
}
```

### Tables



```
table ipv4_lpm {
   key = {
     hdr.ipv4.dstAddr: lpm;
   }
   actions = {
     ipv4_forward;
     drop;
     NoAction;
   }
   size = 1024;
   default_action = NoAction();
}
```

### Applying Tables in Controls



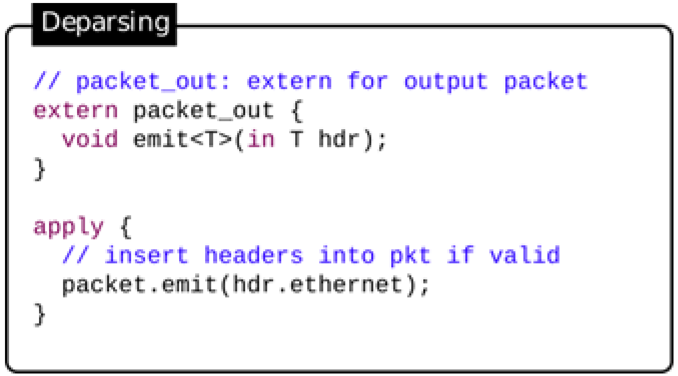
```
control MyIngress(inout headers hdr,
                  inout metadata meta,
                  inout standard_metadata_t standard_metadata) {
  table ipv4_lpm {
    ...
  }
  apply {
    ...
    ipv4_lpm.apply();
    ...
  }
}
```

UNIVERSITY OF TWENTE.

## Programming in P4

### Deparsing

- Emit headers in front of payload.



Deparsing

```
// packet_out: extern for output packet
extern packet_out {
  void emit<T>(in T hdr);
}

apply {
  // insert headers into pkt if valid
  packet.emit(hdr.ethernet);
}
```

## Lab Assignments
Repository

- Instructions on gitlab:
- https://gitlab.utwente.nl/m7717102/p4-labs-2025
- Fork of P4 language tutorials

## Lab Assignments
What you need to do

- Assignment 1
  - Basic Forwarding
  - Basic Tunneling
- Assignment 2
  - P4Runtime
- Assignment 3
  - Firewall
- Assignment 4
  - Load balancing
  - Controlled load balancing

## Lab Assignments

How to carry them out

- Follow tutorial instructions (in the README files)
- Add comments to P4 code briefly explaining
  - What the code does
  - Why you did it that way
  - Parts of the cheat sheet that you used
- Only use the cheat sheet, do not use the answers (obviously)
- Upload P4 code to Canvas

## Lab Assignments

Signing off

- Demonstrate your code and its behavior
- Briefly explain what you did
- Might ask more in-depth questions

## Lab Assignments
Sessions

- Monday 6th October: Sign-off session #1
- Thursday 3rd November: Sign-off session #2

Grading:

- Pass if everything signed off on Monday 3rd November

## Lab Assignments
Tips

- Assume you need around **16 hours** to do all the assignments

  ⇒ around half the work needs to be done outside the lab sessions
- Try to have the **Assignments 1 & 2** finished at the **first lab session**
- Fully read what you have to implement before actually writing any code
- Understand the files you have to edit
- In the VM, if *Backspace* suddenly does not work anymore:
  use *CTRL + Backspace*

## Links

P4 resources: http://P4.org/learn

Assignment Repo: https://gitlab.utwente.nl/m7717102/p4-labs-2025

V1Model source code and docs:

https://github.com/p4lang/p4c/blob/main/p4include/v1model.p4