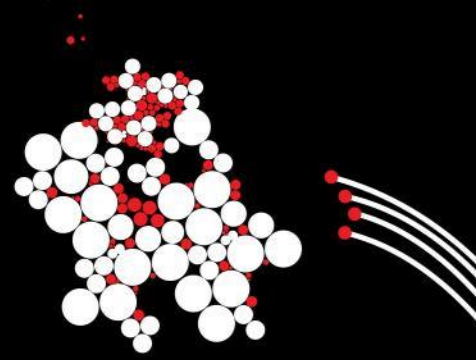


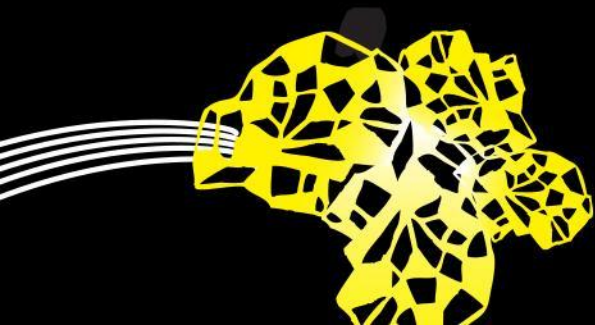
# UNIVERSITY OF TWENTE.



## P4 Lab

Advanced Networking

Bas Bleijerveld





# P4 Lab

Teaching Assistant

---

Bas Bleijerveld

[b.bleijerveld@student.utwente.nl](mailto:b.bleijerveld@student.utwente.nl)

UNIVERSITY OF TWENTE.



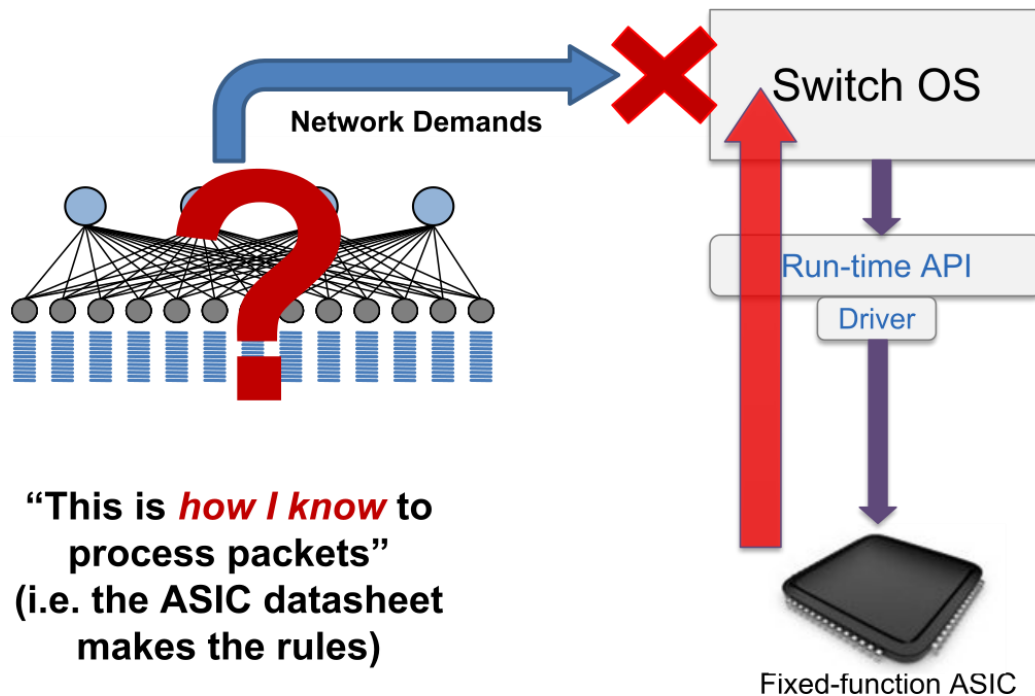
# Overview

---

- Introduction
- Architectural Overview
- Programming in P4
- Lab Assignments

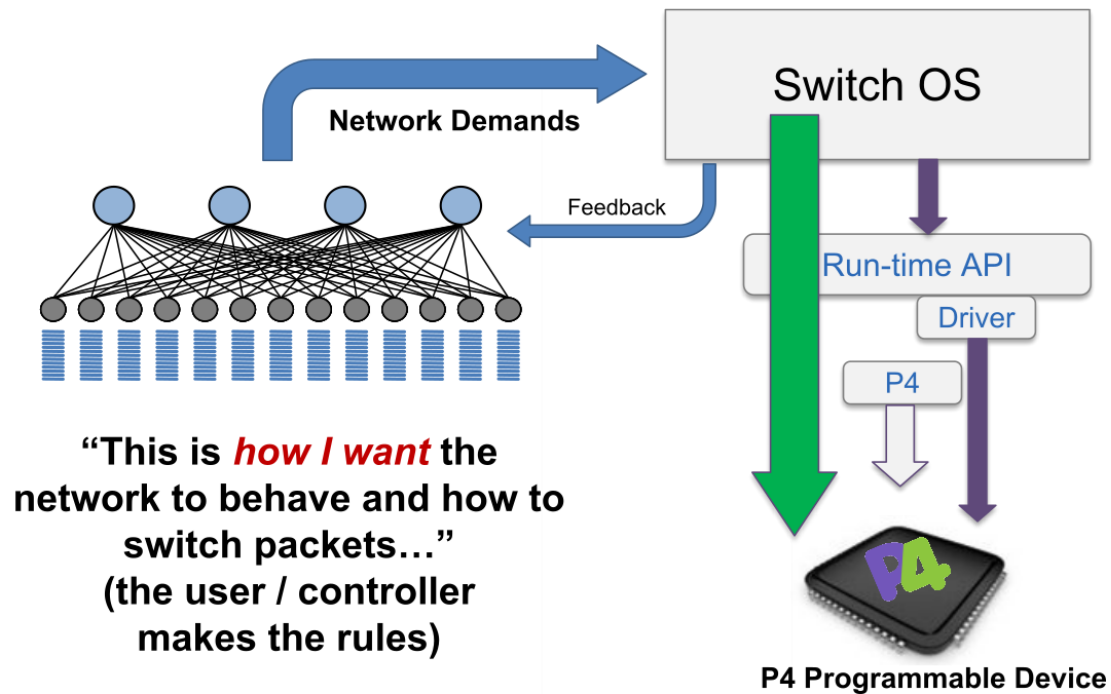
# Introduction

Bottom-up



# Introduction

Top-down

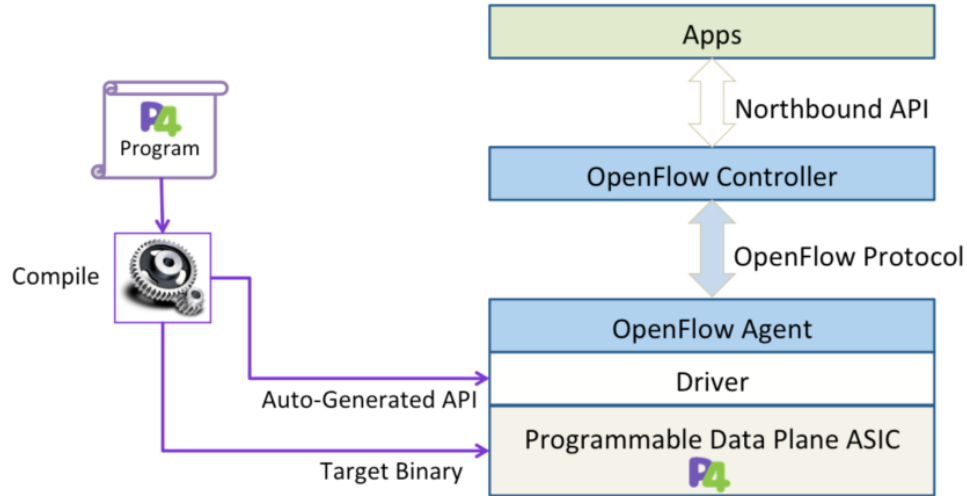


# Architectural Overview

## Control Plane & Data Plane



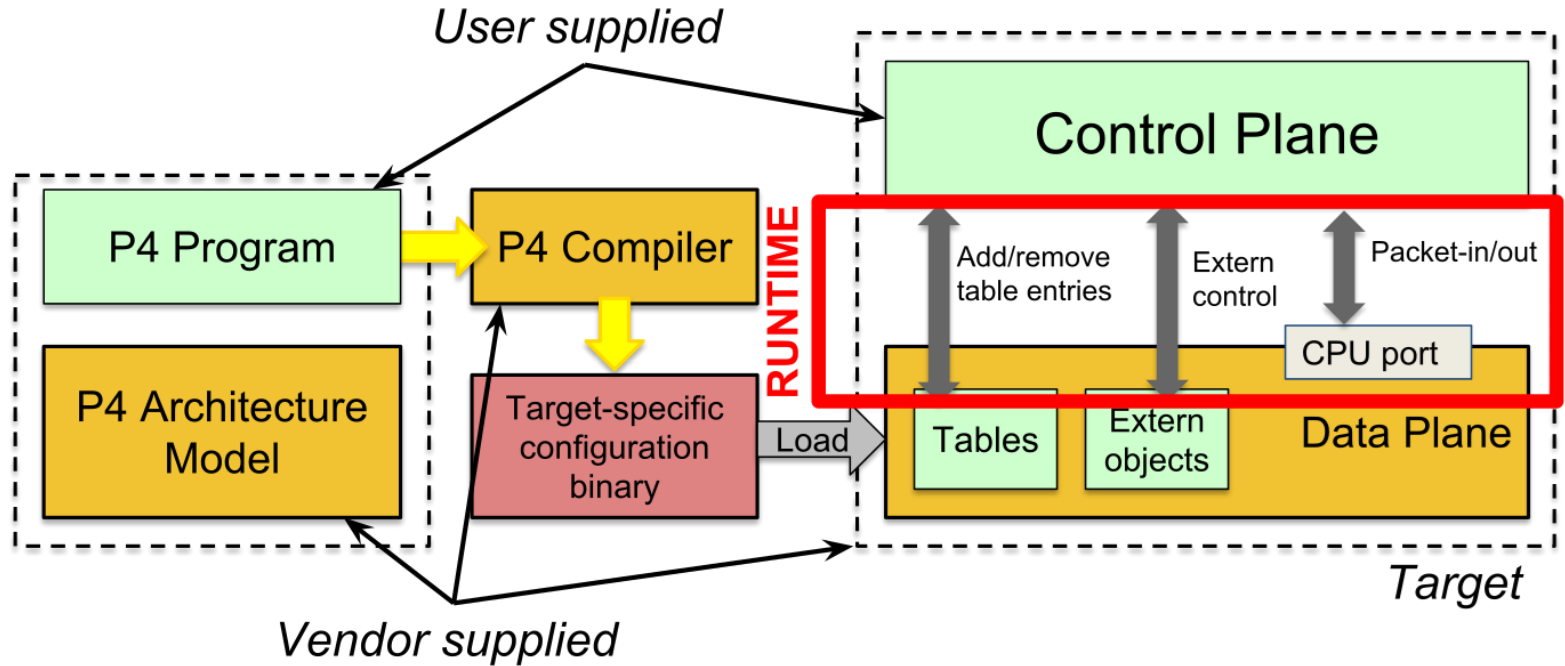
### P4 & OpenFlow

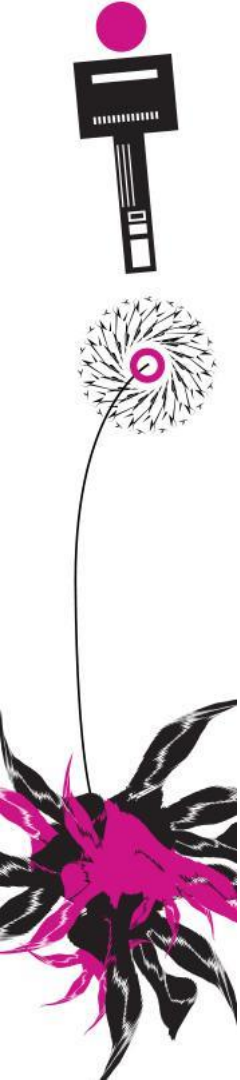


Copyright © 2016 P4 Language Consortium.

# Architectural Overview

P4





# Architectural Overview

## BMV2 Switch

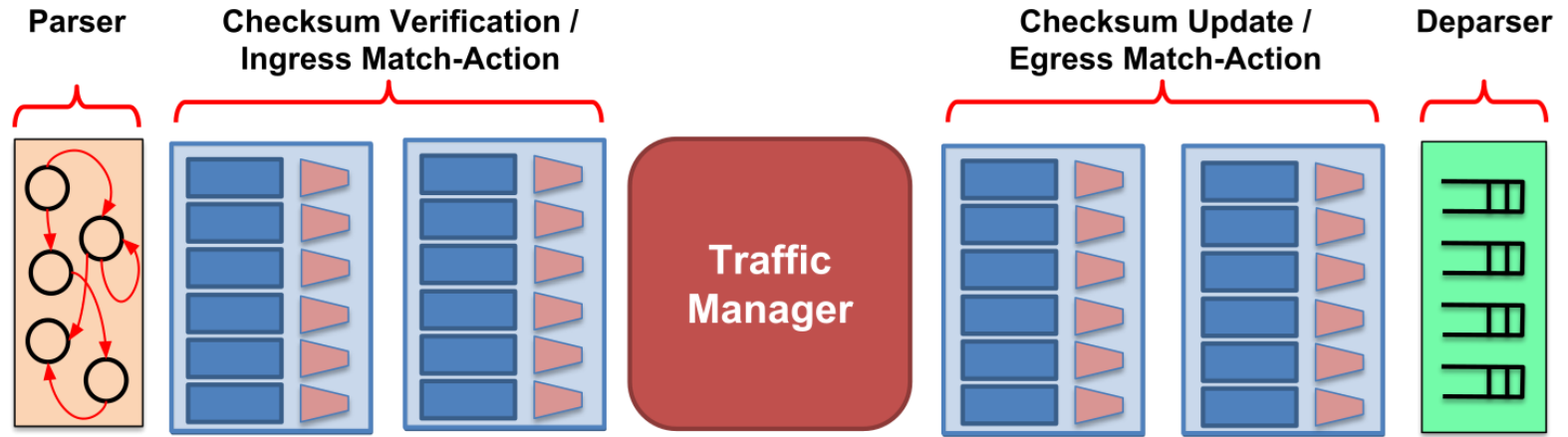
---

- Simulated switch
- Used for lab assignments



# Architectural Overview

## Stages





# Architectural Overview

## 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;
```



# Programming in P4

## Headers

---

```
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

## Parsing

- Externs are functions provided by architecture
- State machine

### 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

### Actions

```
// Inputs provided by control-plane
action set_next_hop(bit<32> next_hop) {
  if (next_hop == 0) {
    metadata.next_hop = hdr.ipv4.dst;
  } else {
    metadata.next_hop = next_hop;
  }
}

// Inputs provided by data-plane
action swap_mac(inout bit<48> x,
               inout bit<48> y) {
  bit<48> tmp = x;
  x = y;
  y = tmp;
}

// Inputs provided by control/data-plane
action forward(in bit<9> p, bit<48> d) {
  standard_metadata.egress_spec = p;
  headers.ethernet.dstAddr = d;
}

// Remove header from packet
action decap_ip_ip() {
  hdr.ipv4 = hdr.inner_ipv4;
  hdr.inner_ipv4.setInvalid();
}
```

### Tables

```
table ipv4_lpm {
  key = {
    hdr.ipv4.dstAddr : lpm;
    // standard match kinds:
    // exact, ternary, lpm
  }
  // actions that can be invoked
  actions = {
    ipv4_forward;
    drop;
    NoAction;
  }
  // table properties
  size = 1024;
  default_action = NoAction();
}
```

### Control Flow

```
apply {
  // branch on header validity
  if (hdr.ipv4.isValid()) {
    ipv4_lpm.apply();
  }
  // branch on table hit result
  if (local_ip_table.apply().hit) {
    send_to_cpu();
  }
  // branch on table action invocation
  switch (table1.apply().action_run) {
    action1: { table2.apply(); }
    action2: { table3.apply(); }
  }
}
```



# Programming in P4

## Actions & Control Flow

```
#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

## Actions & Control Flow

```
#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



# Programming in P4

## Deparsing

---

- Emit headers in front of payload
- Watch the order!

### 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

---

- <https://github.com/happyfield7/p4lab>
- Fork of P4 language tutorials



# Lab Assignments

## Virtual Machine

---

- All necessary tools installed
- Repository cloned
- Link on Canvas



# Lab Assignments

What you need to do

---

- Week 1
  - Basic Forwarding
  - Basic Tunneling
- Week 2
  - P4Runtime
- Week 3
  - Explicit Congestion Notification
  - Multi-Hop Route Inspection
- Week 4
  - Source Routing
  - Firewall



# 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

Signing off / Submission files

---

- Thursday 23:59 deadline for submission files that week
- Monday 10 minutes sign-off (with 5 minutes between each student)
- I will announce the schedule for Monday on Canvas
- Fixed schedule or different times?



# Lab Assignments

## Sessions

---

- Mon 28th September: Online sign-off session #1
- Mon 5th October: Online sign-off session #2
- Mon 12th October: Online sign-off session #3
- Mon 19th October: Online sign-off session #4



# Lab Assignments

## Integrity statement

---

- Submit the integrity statement on Canvas
- After that you can start with the P4 Lab