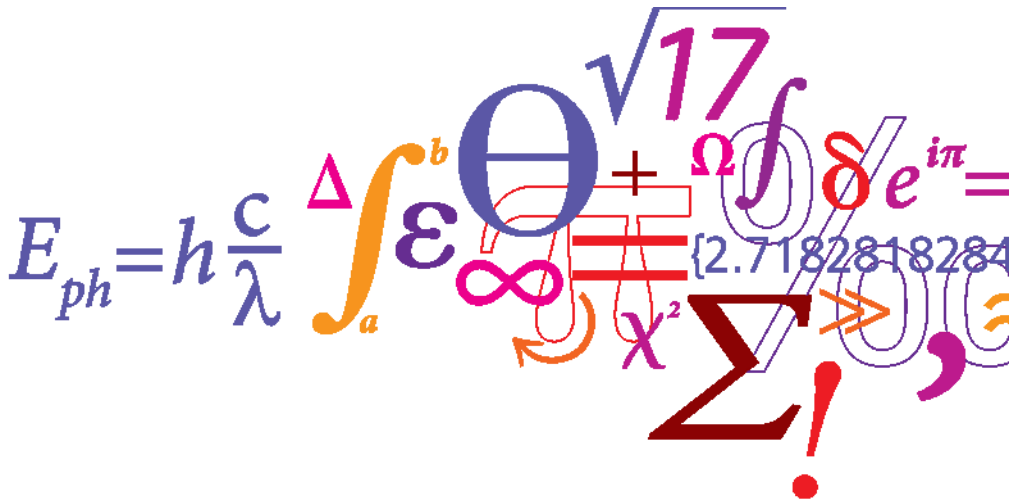
A short intro to P4

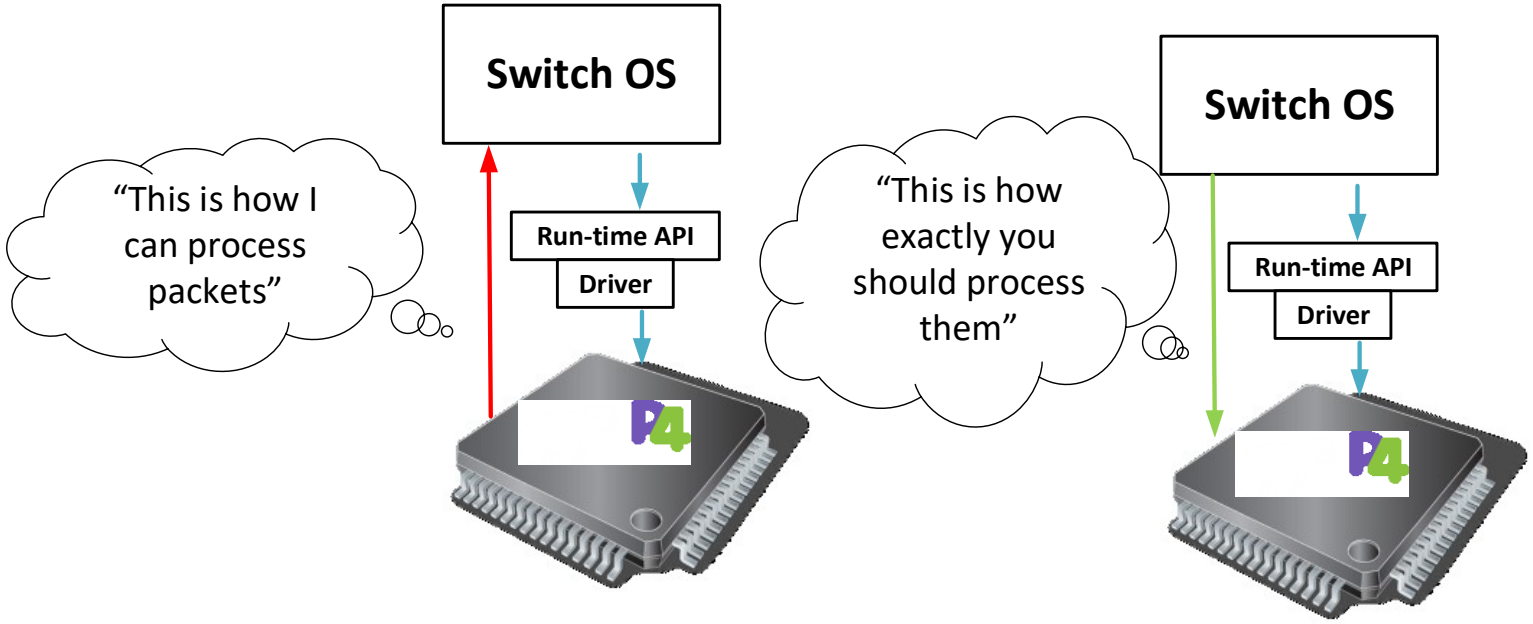
Eder Ollora Zaballa



DTU Fotonik
Department of Photonics Engineering

Motivation

- SDN related: Each Openflow specification added more and more headers. Instead of incrementing the support for new headers in each specification, why not let programmers define the data plane?



Terminology

- This is a very important part since you will find a lot of information, typically mixing terms:

- **Target**: The embodiment of a specific hardware implementation.
  - For instance: **Simple Switch**. So Simple Switch is a target and is built on top of a set of tools named Behavioral Model v2 (aka Bmv2) [1]. Sometimes people refer to their switch as a Bmv2 switch but in reality you can implement another target based on Bmv2 [2].

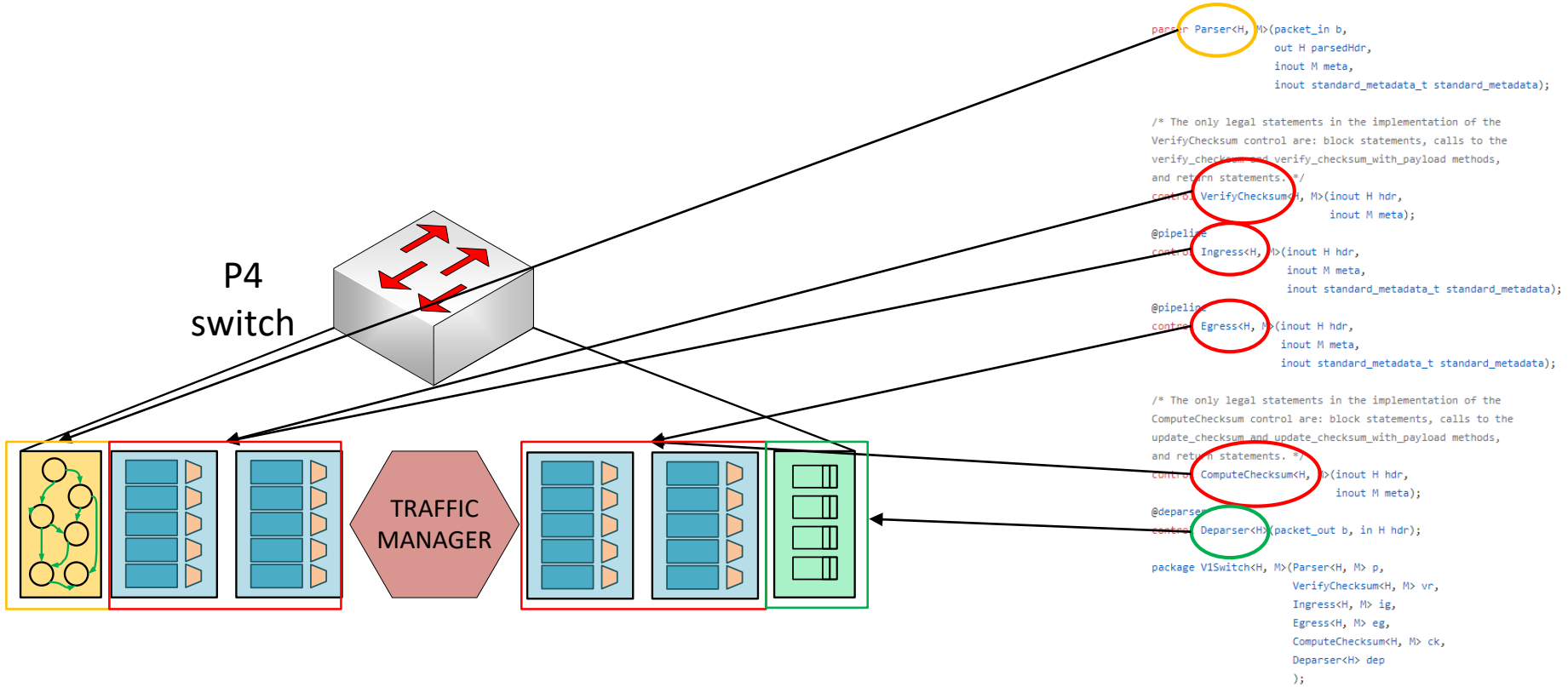
- **Architecture**: The interface to program a target, and defines a set of P4-programmable components, externs or even fixed components.
  - For instance: **V1 model** [3]. The architecture defines which blocks we can program.

[1] https://github.com/p4lang/behavioral-model

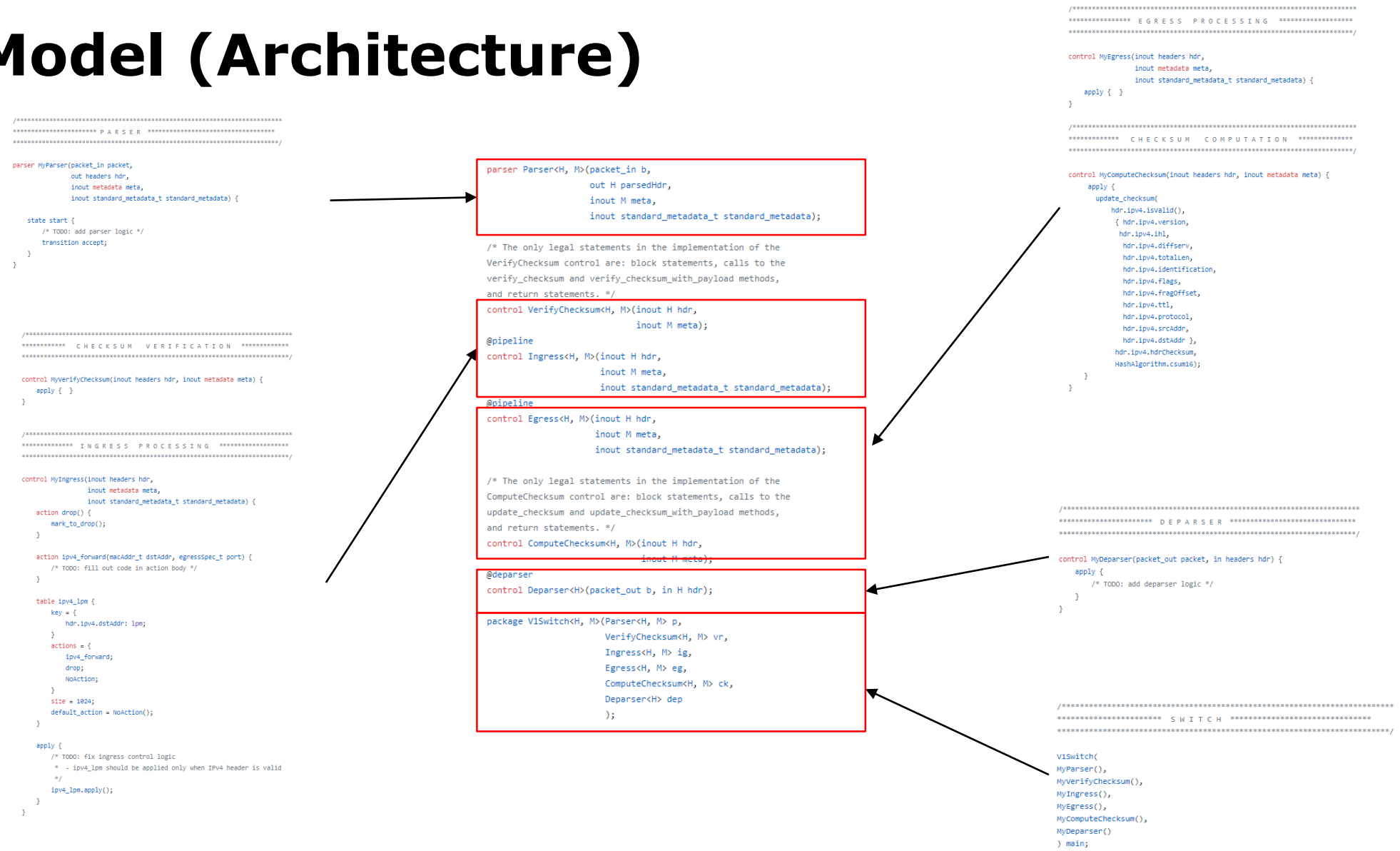
[2] https://github.com/p4lang/behavioral-model/blob/master/docs/simple_switch.md

[3]https://github.com/p4lang/p4c/blob/master/p4include/v1model.p4

V1 Model (Architecture)



V1 Model (Architecture)



```
/*************************************************************************
*********************** P A R S E R  ***********************************
*************************************************************************/

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

   state start {
       /* TODO: add parser logic */
       transition accept;
   }
}
```

```
/*************************************************************************
************   C H E C K S U M    V E R I F I C A T I O N   *************
*************************************************************************/

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

```
/*************************************************************************
*************  I N G R E S S   P R O C E S S I N G   *******************
*************************************************************************/

control MyIngress(inout headers hdr,
                  inout metadata meta,
                  inout standard_metadata_t standard_metadata) {
    action drop() {
        mark_to_drop();
    }

    action ipv4_forward(macAddr_t dstAddr, egressSpec_t port) {
        /* TODO: fill out code in action body */
    }

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

    apply {
        /* TODO: fix ingress control logic
         *  - ipv4_lpm should be applied only when IPv4 header is valid
         */
        ipv4_lpm.apply();
    }
}
```

```
parser Parser<H, M>(packet_in b,
                    out H parsedHdr,
                    inout M meta,
                    inout standard_metadata_t standard_metadata);

/* The only legal statements in the implementation of the
VerifyChecksum control are: block statements, calls to the
verify_checksum and verify_checksum_with_payload methods,
and return statements. */
control VerifyChecksum<H, M>(inout H hdr,
                             inout M meta);
@pipeline
control Ingress<H, M>(inout H hdr,
                      inout M meta,
                      inout standard_metadata_t standard_metadata);
@pipeline
control Egress<H, M>(inout H hdr,
                     inout M meta,
                     inout standard_metadata_t standard_metadata);

/* The only legal statements in the implementation of the
ComputeChecksum control are: block statements, calls to the
update_checksum and update_checksum_with_payload methods,
and return statements. */
control ComputeChecksum<H, M>(inout H hdr,
                              inout M meta);
@deparser
control Deparser<H>(packet_out b, in H hdr);

package V1Switch<H, M>(Parser<H, M> p,
                       VerifyChecksum<H, M> vr,
                       Ingress<H, M> ig,
                       Egress<H, M> eg,
                       ComputeChecksum<H, M> ck,
                       Deparser<H> dep
                       );
```

```
/*************************************************************************
*************  E G R E S S   P R O C E S S I N G   *******************
*************************************************************************/

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

/*************************************************************************
************   C H E C K S U M    C O M P U T A T I O N   **************
*************************************************************************/

control MyComputeChecksum(inout headers hdr, inout metadata meta) {
     apply {
       update_checksum(
           hdr.ipv4.isValid(),
            { hdr.ipv4.version,
             hdr.ipv4.ihl,
              hdr.ipv4.diffserv,
              hdr.ipv4.totalLen,
              hdr.ipv4.identification,
              hdr.ipv4.flags,
              hdr.ipv4.fragOffset,
              hdr.ipv4.ttl,
              hdr.ipv4.protocol,
              hdr.ipv4.srcAddr,
              hdr.ipv4.dstAddr },
             hdr.ipv4.hdrChecksum,
             HashAlgorithm.csum16);
    }
}
```

```
/*************************************************************************
***********************  D E P A R S E R  *******************************
*************************************************************************/

control MyDeparser(packet_out packet, in headers hdr) {
    apply {
        /* TODO: add deparser logic */
    }
}
```

```
/*************************************************************************
***********************  S W I T C H  *******************************
*************************************************************************/

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

Standard metadata (V1 model)

[4]

```
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)

Target programming