P4 Compiler in SDN

Federico Bruzzone[1], PhD Student

Milan, Italy – 30 October 2024

Slides available at: federicobruzzone.github.io/activities/presentations/p4-compiler-in-SDN.pdf

[1] ADAPT Lab – Università degli Studi di Milano,
Website: federicobruzzone.github.io,
Github: github.com/FedericoBruzzone,
Email: federico.bruzzone@unimi.it

Network Programmability

The ability of the software or the hardware to extecute an externally defined processing algorithm[2]

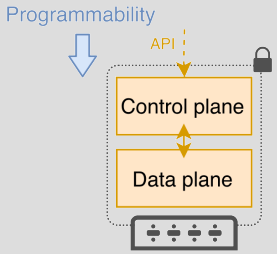
[2]Hauser et al., “A Survey on Data Plane Programming with P4: Fundamentals, Advances, And Applied Research”.

Open Networking Foundation (ONF)

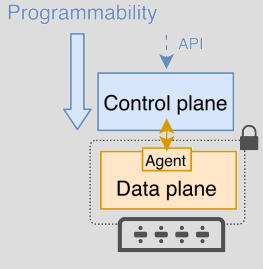
- Non-profit consortium founded in 2011
- Promotes networking through **Software Defined Networking** (SDN)
- Standardizes the **OpenFlow** protocol

Software Defined Networking (SDN)

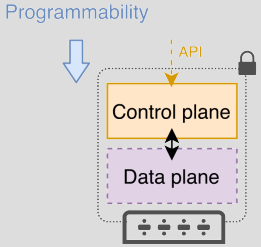
- Born to overcome the limitations of traditional network architectures
- Decouples the control plane from the data plane
- Centralizes the control of the network



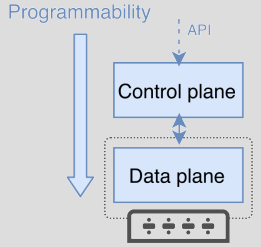
Traditional networking



SDN with fixed-function data plane



Vendor-based creation of network devices with data plane programming



Full network programability with data plane programming

OpenFlow Protocol

- Gives access to the **forwarding plane** (data plane) of a network device
- Mainly used by switches and controllers
- Layered on top of the **Transport Control Protocol** (TCP)
- De-facto standard for SDN

OpenFlow Development

- First appeared in 2008[3]
- In April 2012, Google deploys OpenFlow in its internal network with significant improvements (Urs Hölzle promotes it[4])
- In January 2013, NEC rolls out OpenFlow for Microsoft Hyper-V
- Latest version is 1.5.1 (Apr 2015)

[3] McKeown et al., "Openflow: Enabling Innovation in Campus Networks".

[4] Inter-Datacenter WAN with centralized TE using SDN and OpenFlow.

Fields in OpenFlow Standard

Version	Date	Header Fields
OF 1.0	Dec 2009	12 fields (Ethernet, TCP/IPv4)
OF 1.1	Feb 2011	15 fields (MPLS, inter-table metadata)
OF 1.2	Dec 2011	36 fields (APR, ICMP, IPv6, etc.)
OF 1.3	Jun 2012	40 fields
OF 1.4	Oct 2013	41 fields

More Details on the OpenFlow v1.0.0 Switch Specification[5]

[5] https://opennetworking.org/wp-content/uploads/2013/04/openflow-spec-v1.0.0.pdf

OpenFlow is protocol-dependent

- Fixed set of fields and parser based on standard protocols

  (Ethernet, IPv4/IPv6, TCP/UDP)

- Assumes that the match+action table are in series

P4: Programming Protocol-Independent Packet Processors

Bosshart believed that future generations of OpenFlow would have allowed the controller to *tell the switch how to operate*[6]

[6]Bosshart et al., "P4: Programming Protocol-Independent Packet Processors".

Goals and Challenges

Reconfigurability: the controller should be able to redefine the packet parsing and processing in the field

Protocol Independence: the switch should *headers* using parsing and processing using *match+action* tables

Target Independence: a compiler from *target-independent* description to *target-dependent* binary

Goals and Challenges

Reconfigurability: the controller should be able to redefine the packet parsing and processing in the field

Protocol Independence: the switch should *headers* using parsing and processing using *match+action* tables

Target Independence: a compiler from *target-independent* description to *target-dependent* binary

Goals and Challenges

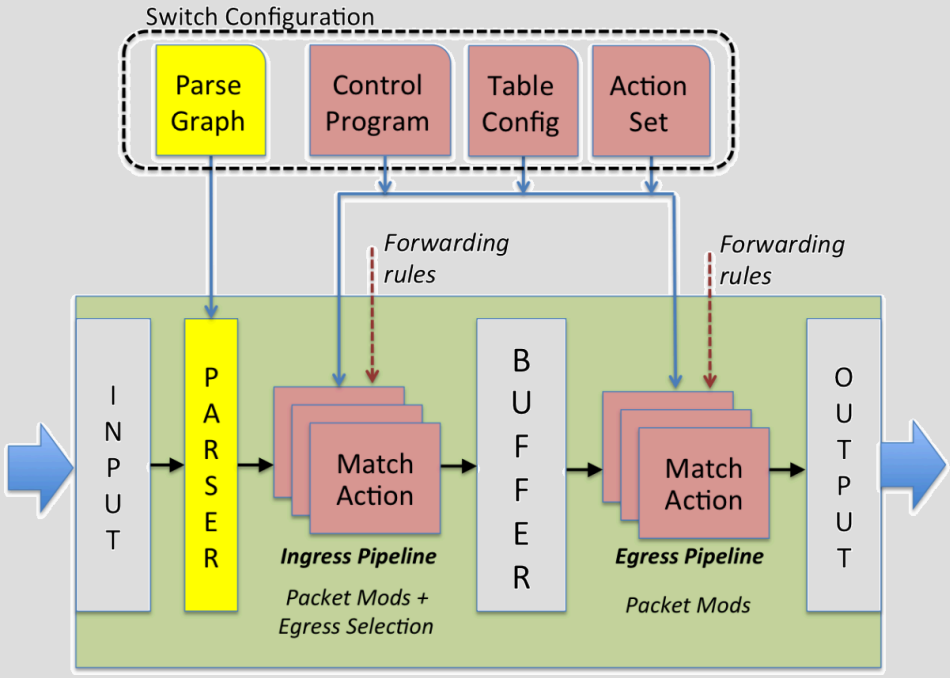
Reconfigurability: the controller should be able to redefine the packet parsing and processing in the field

Protocol Independence: the switch should *headers* using parsing and processing using *match+action* tables

Target Independence: a compiler from *target-independent* description to *target-dependent* binary

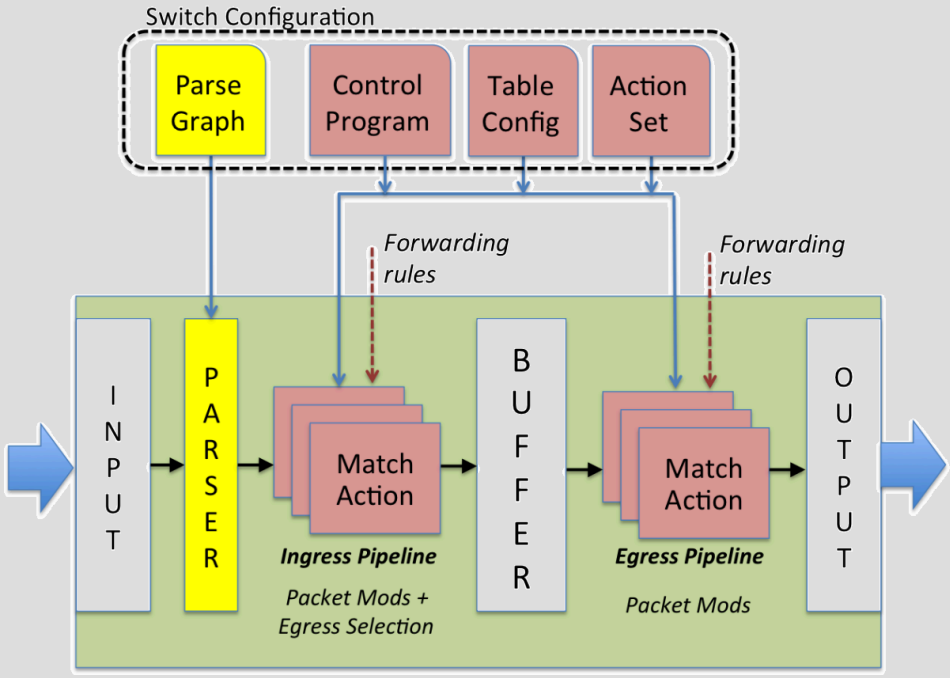
Abstract Forwarding Model (AFM)

1. Parsing the packet headers
2. The fields are passed to the match-action pipeline.
   - **Ingrees**: determines the egress port/queue
   - **Egress**: per-instance header modifications
3. Metadata processing (e.g., timestamp)
4. As in OpenFlow, the queuing discipline is chosen at switch configuration time



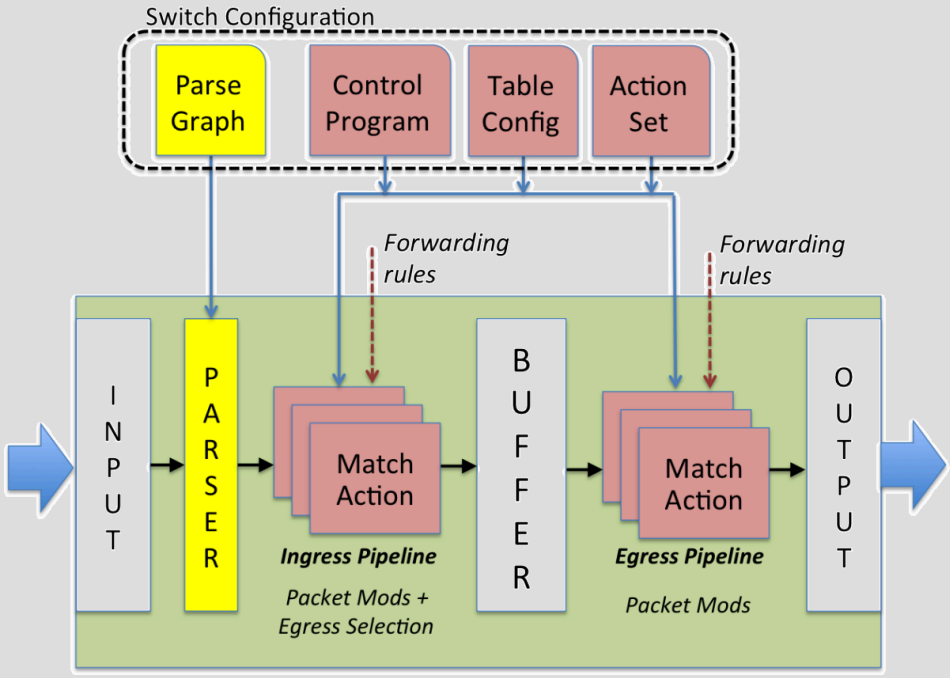
Abstract Forwarding Model (AFM)

1. Parsing the packet headers
2. The fields are passed to the match-action pipeline.
   - **Ingrees**: determines the egress port/queue
   - **Egress**: per-instance header modifications
3. Metadata processing (e.g., timestamp)
4. As in OpenFlow, the queuing discipline is chosen at switch configuration time



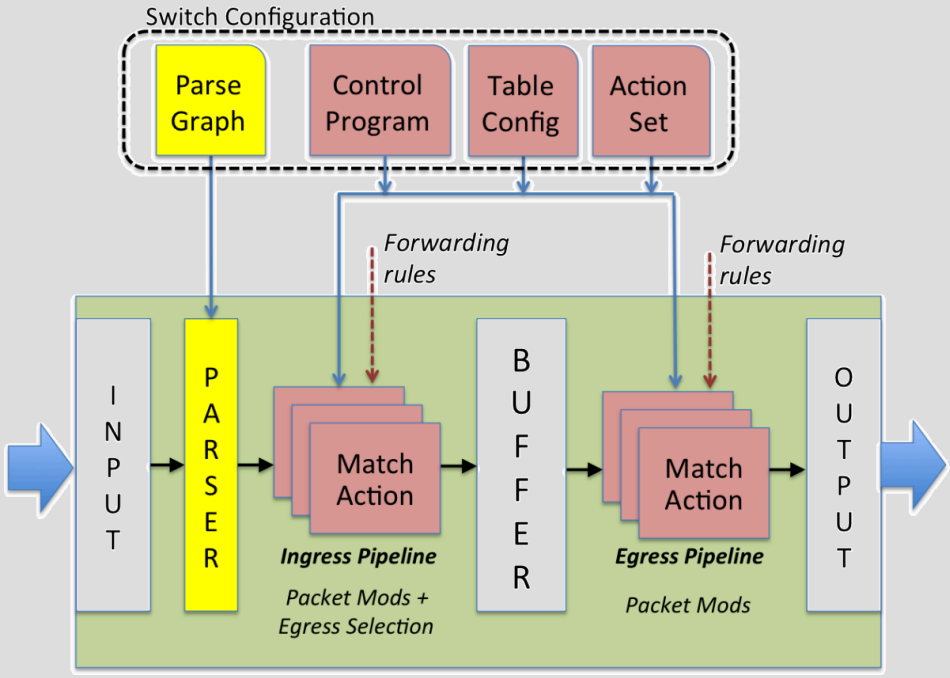
Abstract Forwarding Model (AFM)

1. Parsing the packet headers
2. The fields are passed to the match-action pipeline.
   - **Ingrees**: determines the egress port/queue
   - **Egress**: per-instance header modifications
3. Metadata processing (e.g., timestamp)
4. As in OpenFlow, the queuing discipline is chosen at switch configuration time



Abstract Forwarding Model (AFM)

1. Parsing the packet headers
2. The fields are passed to the match-action pipeline.
   - **Ingrees**: determines the egress port/queue
   - **Egress**: per-instance header modifications
3. Metadata processing (e.g., timestamp)
4. As in OpenFlow, the queuing discipline is chosen at switch configuration time



Two-stage Compilation

Imperative control flow program based on **AFM**

1. The compiler translate the P4 program into **TDGs** (Table Dependency Graphs)

2. The **TDGs** are compiled into **target-dependent** code

Language Design - Header

Describes the structure of a series of fields including the widths and constraints on values.

```
header ethernet {
  fields {
    dst_addr: 48; // bits
    src_addr: 48;
    ethertype: 16;
  }
}
```

```
header vlan {
  fields {
    pcp: 3;
    cfi: 1;
    vid: 12;
    ethertype: 16;
  }
}
```