

# Laboratorio de redes

## Proyecto de redes y servicios

### Programación de dispositivos del plano de datos con P4

Clemente Barreto Pestana

[cbarretp@ull.edu.es](mailto:cbarretp@ull.edu.es)

Profesor Asociado

Área de Ingeniería Telemática  
Departamento de Ingeniería Industrial  
Escuela Superior de Ingeniería y Tecnología

# Índice

- **Introducción**
- **Entregables**
- **Evaluación**
- **Cronograma**



# Introducción (De la guía docente):

- El **bloque II** se cubre con el desarrollo de un **proyecto en grupo**, cuya memoria se deberá presentar en **inglés** y que se deberá **exponer** ante los compañeros y defender.
- El bloque II trata del **diseño avanzado** de redes y proyecto de redes:
  - **Simuladores** de red.
  - Protocolos de comunicaciones
  - **Estudio avanzado de protocolos y dispositivos** de los niveles 1 a 4.
  - Desarrollo de un proyecto en el ámbito de las redes.



# Descripción (I)

Se propone el desarrollo de un proyecto sobre el diseño e implementación avanzada de protocolos y dispositivos de los niveles 1 a 4:

- Las **redes definidas por software** (Software Defined Networks o SDN).
- **Programación de los dispositivos** del **plano de datos** con el lenguaje P4.



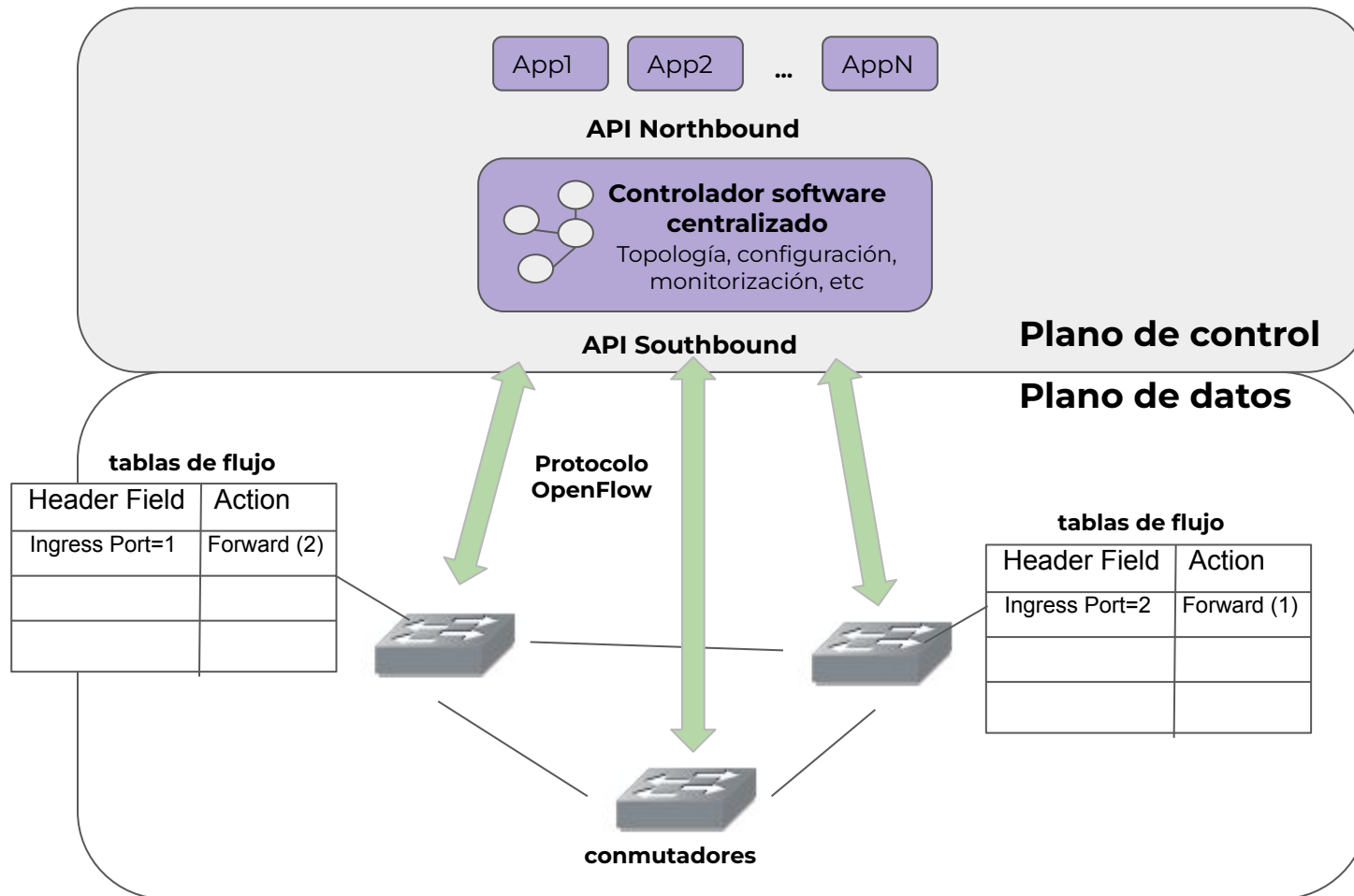
## Descripción (II)

El objeto del proyecto es implementar (en GNS3) un dispositivo del plano de datos capaz de realizar un NAT básico. Tecnologías que usaremos:

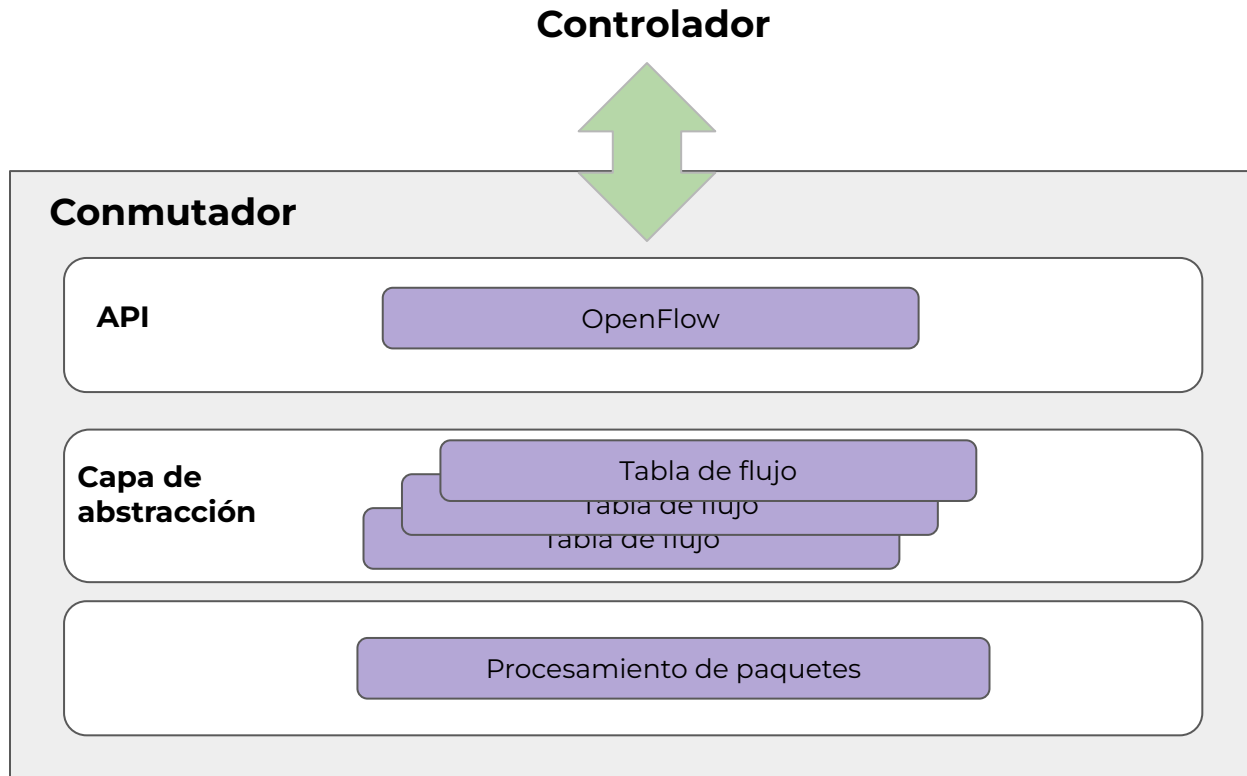
- Lenguaje de programación: P4 (Programming Protocol-independent Packet Processors).
- Conmutador software programable: BMv2.



# Redes definidas por software



# Conmutador del plano de datos



# Programación del plano de datos

## Estandarización de arquitectura y lenguaje

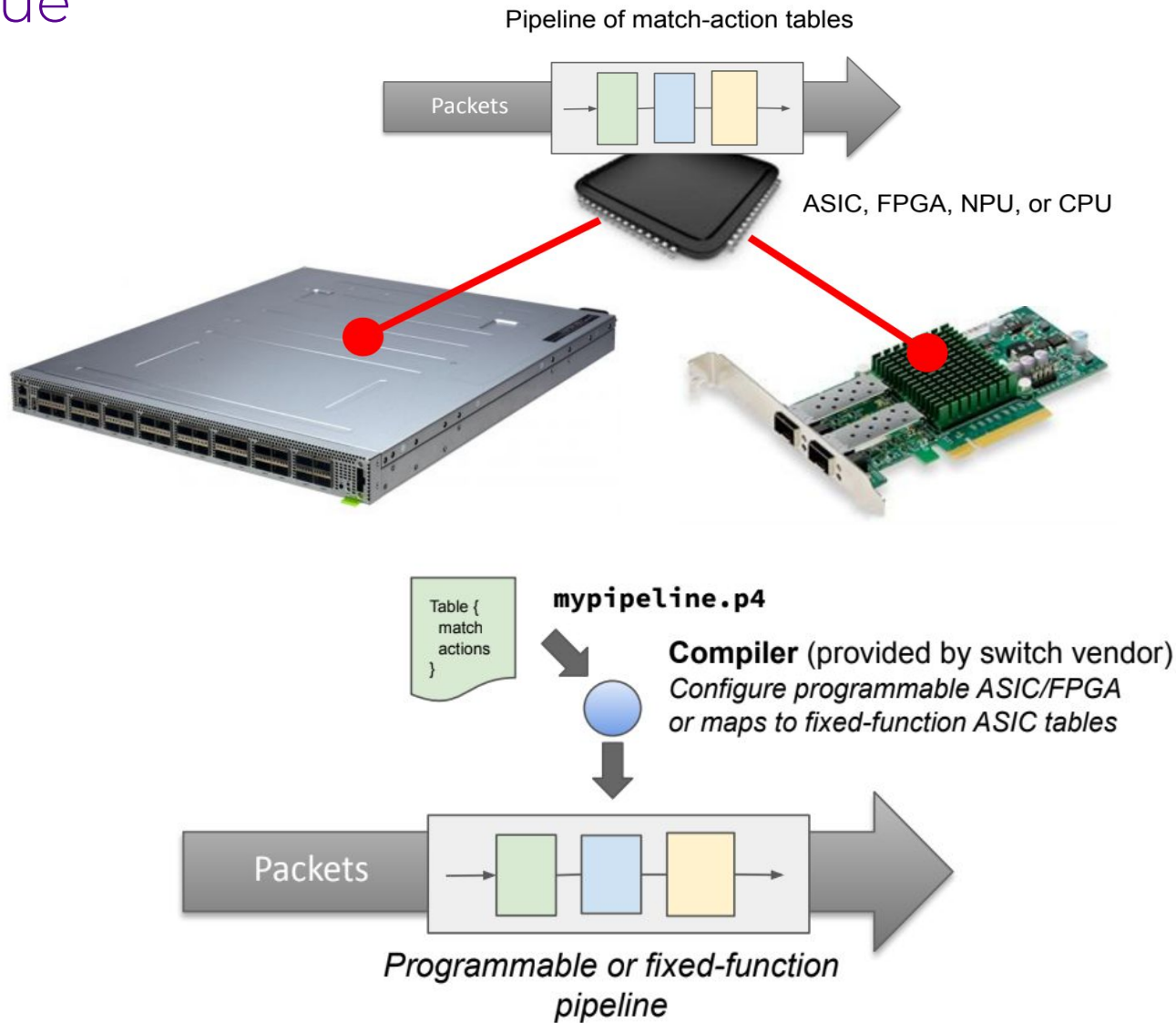
Domain	Year	Processing Unit	Main Language/s
General computing	1971	Central Processing Unit (CPU)	C, Java, Python, etc.
Signal processing	1979	Digital Signal Processor (DSP)	Matlab
Graphics	1994	Graphics Processing Unit (GPU)	Open Computing Language
Machine learning	2015	Tensor Processing Unit (TPU)	Tensor Flow
Computer networks	2016	Protocol Independent Switch Architecture (PISA)	P4



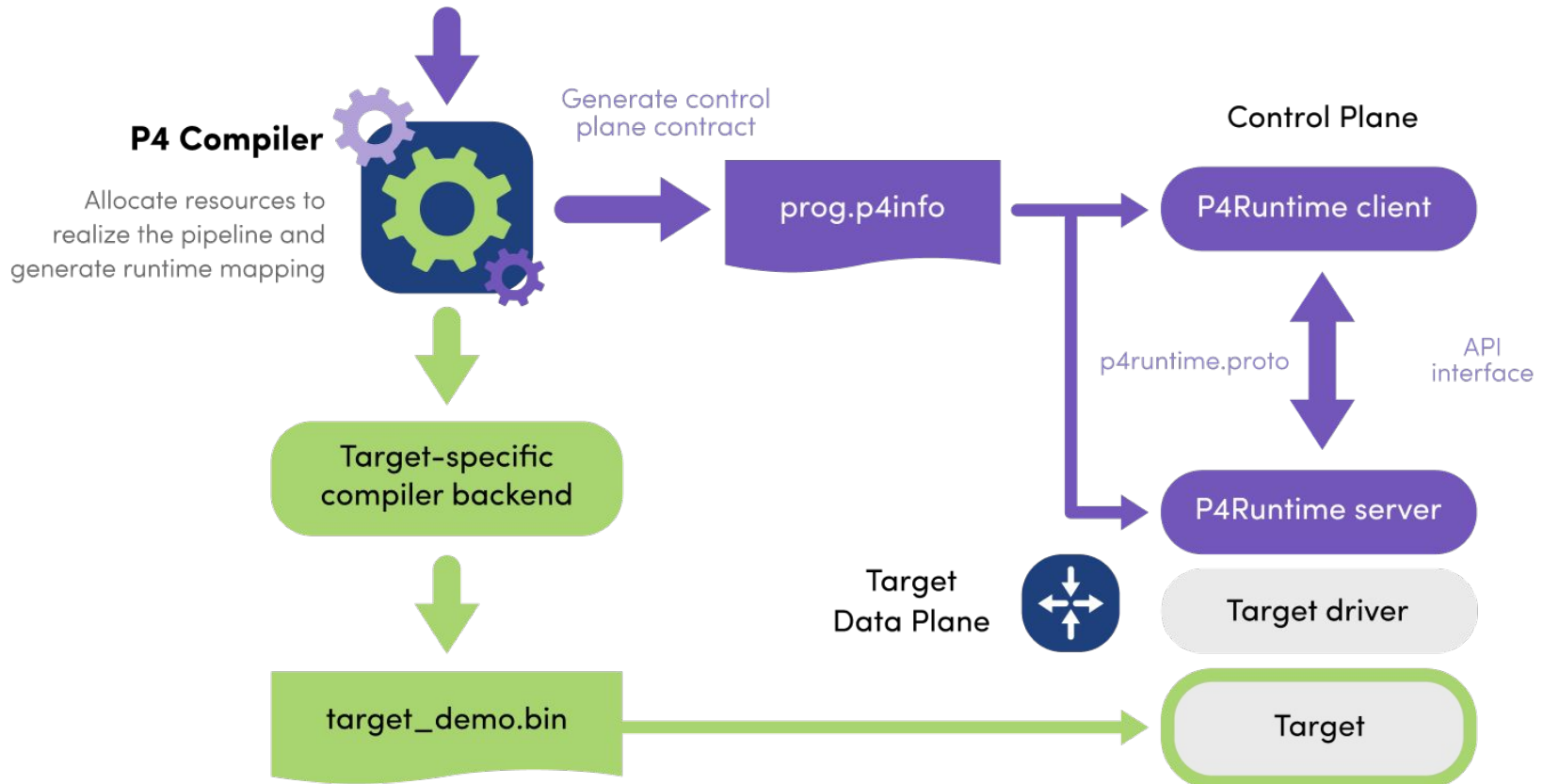
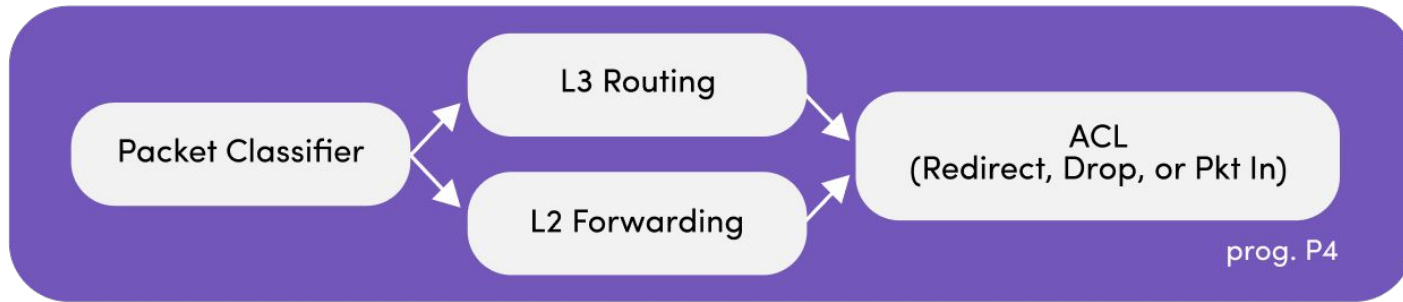



# Programación del plano de datos

## Enfoque



## P4 Program





```
control ingress () {  
  table routing {  
    key = { hdr.ipv4.dstAddr : lpm; }  
    actions = { drop; route; }  
    size = 2048;  
  }  
  apply {  
    routing.apply();  
  }  
}
```

# Evaluación (De la guía docente):

- **9. Sistema de evaluación y calificación**

- Bloque II - Práctica proyecto (PP) (**25%**):  
Calificación del proyecto obtenida en una escala de **0 a 10**.

- **Criterios:**

- Adecuación a lo solicitado.
- Funcionalidad de la configuración.
- Nivel de conocimientos adquiridos.
- Aplicabilidad y viabilidad de la solución propuesta
- Capacidad de trabajo en grupo.
- Concreción en la redacción.
- Expresión oral y presentación.



# Entregables (I)

Los 10 ptos del proyecto se repartirán así:

- **Memoria (6 ptos)**

- Documento (**en inglés**).
  - Evidencias realización laboratorios y explicación de resultados (1 pto).
  - Análisis/diseño, implementación y pruebas de las mejoras propuestas (5 ptos).

- **Cronograma (1 pto)**

- Diagrama Gantt.
  - EDT y reparto de horas entre miembros del equipo con tareas, personas y horas.



# Entregables (II)

- **Presentación (3 ptos):** 15 minutos
  - Explicación + defensa (**en español**).
    - Análisis/diseño, implementación y pruebas de las mejoras propuestas (2 ptos).
    - Propuesta, detalle y justificación de otras mejoras (1 pto).



# Vinculación a itinerario IC

- **Competencias:**

- a. C31 - Capacidad de **diseñar y construir** sistemas digitales, incluyendo computadores, sistemas basados en microprocesador y **sistemas de comunicaciones**.
- b. C34 - Capacidad de **diseñar e implementar software** de sistema y de **comunicaciones**.
- c. C38 - Capacidad para **diseñar, desplegar, administrar** y gestionar redes de computadores

- **Resultados de aprendizaje:**

- a. Demostrar **conocimientos prácticos** para **diseñar y analizar protocolos y dispositivos de red** que abarquen las **capas de la 1 hasta la 4** del modelo OSI.
- b. Demostrar la destreza necesaria para llevar a cabo las **configuraciones necesarias para desplegar y mantener una infraestructura** de red.
- c. Demostrar capacidad para **desarrollar un proyecto** en el ámbito de las redes.



# Planificación temporal

- **Esfuerzos por persona** (Guía didáctica):  
20 h (pres) + 30h (autó) = 50 h/persona
- **Desarrollo**
  - En grupo.
  - Presencial (laboratorio) y autónomo (la mayor parte de la dedicación).
  - Horario habitual.
- **Entregables:** hasta viernes 5/5 (a las 20h).
- **Defensas:** semana 8/5 (en el laboratorio en el horario habitual).





# Plan de trabajo

- **Primeras 2 semanas:**

- Desarrollo inicial guiado (2 partes, una cada semana).
- Inicio desarrollo autónomo de mejoras.
- Inicio de preparación de entregables.

- **Siguientes 2 semanas:**

- Fin de desarrollo autónomo de mejoras.
- Fin preparación de entregables.
- Subida de entregables en el aula virtual.

- **Última semana:**

- Preparación presentación.
- Realizar presentación.



# Cronograma

## ABRIL

## 2023

LUNES	MARTES	MIÉRCOLES	JUEVES	VIERNES	SABADO	DOMINGO
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Semanas de desarrollo

## MAYO

## 2023

LUNES	MARTES	MIÉRCOLES	JUEVES	VIERNES	SABADO	DOMINGO
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21

Envío de entregables

Presentaciones



# Semana 2. Uso en P4 de tablas match-action

Programmer declares the headers that should be recognized and their order in the packet

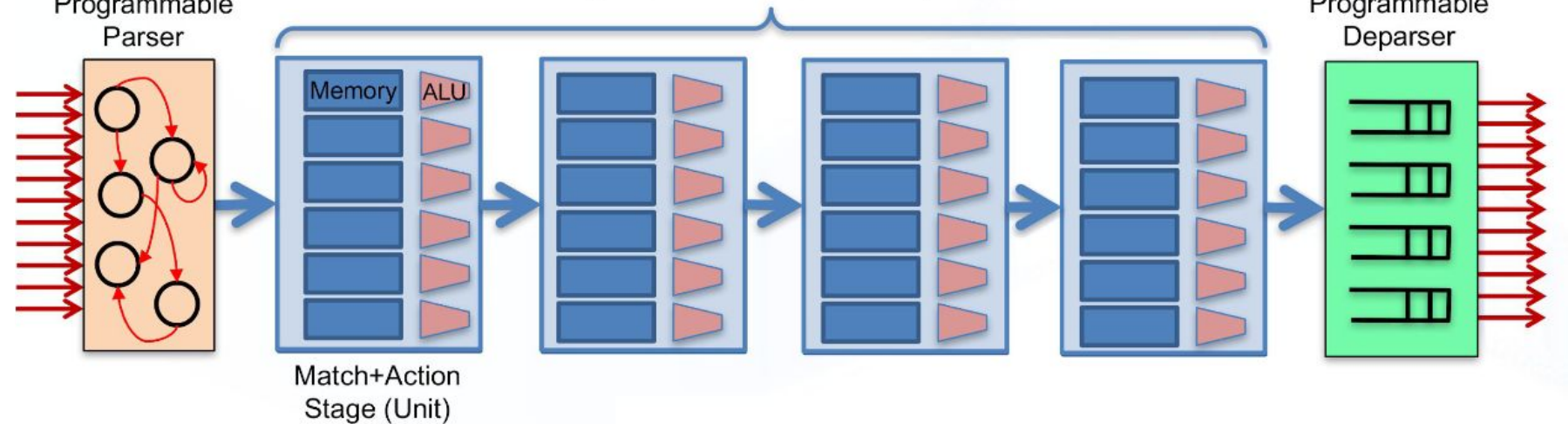
Programmer defines the tables and the exact processing algorithm

Programmer declares how the output packet will look on the wire

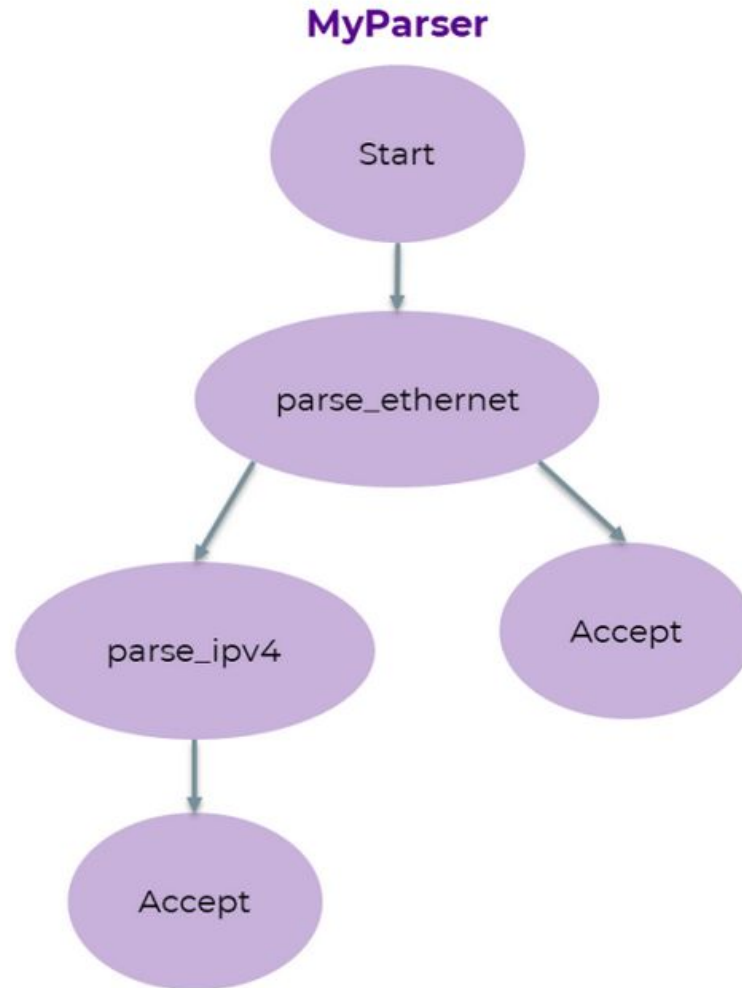
Programmable Parser

Programmable Match-Action Pipeline

Programmable Deparser



# Parser (DAG)



# Programa P4 con tablas: router sencillo

```
header ethernet_t {
    bit<48> dst_addr;
    bit<48> src_addr;
    bit<16> eth_type;
}

header ipv4_t {
    bit<4> version;
    bit<4> ihl;
    bit<8> diffserv;
    ...
}

parser parser_impl(packet_in pkt, out headers_t hdr) {
    /* Parser state machine to extract header fields */
}
```

```
action set_next_hop(bit<48> dst_addr) {
    ethernet.dst_addr = dst_addr;
    ipv4.ttl = ipv4.ttl - 1;
}

...

table ipv4_routing_table {
    key = {
        ipv4.dst_addr : LPM; // longest-prefix match
    }
    actions = {
        set_next_hop();
        drop();
    }
    size = 4096; // table entries
}
```



# Plano de datos vs control

## Simple router example

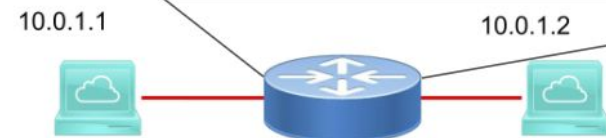
- **Data plane (P4) program**

- Defines the format of the table
  - Match fields, actions, action data (parameters)
- Performs the lookup
- Executes the chosen action

- **Control plane**

- Populates table entries with specific information
  - Based on configuration, automatic discovery, protocol calculations

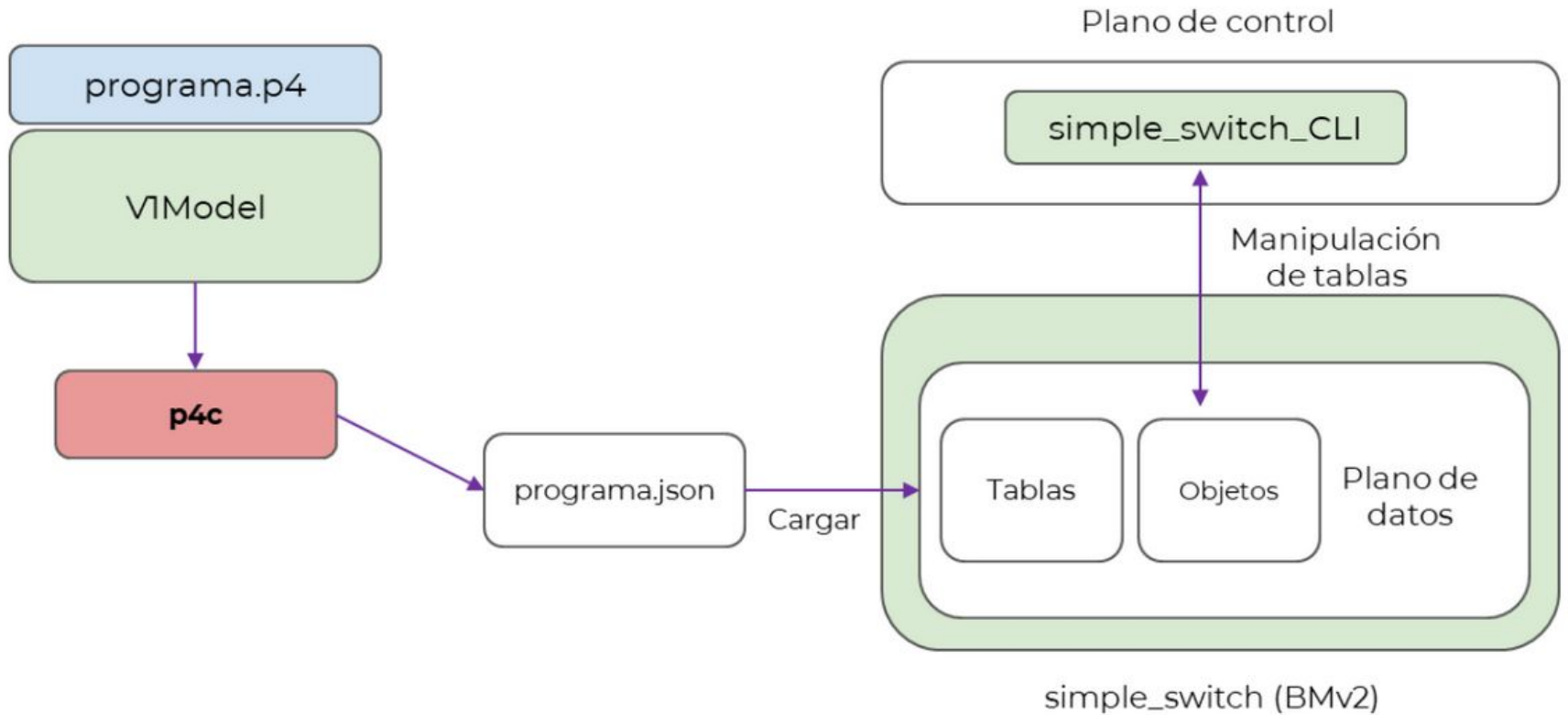
```
action ipv4_forward(bit<48> dst_addr, bit<9> port) {  
    ethernet.dst_addr = dst_addr;  
    standard_metadata.egress_spec = port;  
    ipv4.ttl = ipv4.ttl - 1;  
}  
table ipv4_routing_table {  
    key = {  
        ipv4.dst_addr : LPM; // longest-prefix match  
    }  
    actions = {  
        ipv4_forward();  
        drop();  
    }  
}
```



**Control plane populates table entries**

Key	Action	Action Data
10.0.1.1/32	ipv4_forward	dstAddr=00:00:00:00:01:01 port=1
10.0.1.2/32	drop	
*	NoAction	

# Arquitectura de despliegue



# Emulación del plano de control

simple\_switch\_CLI



Para añadir las entradas que hemos comentado debemos lanzar los siguientes comandos:

```
table_set_default ipv4_lpm drop
table_add MyIngress.ipv4_lpm MyIngress.ipv4_forward 10.0.0.0/24 => 00:00:00:00:00:01 0
table_add MyIngress.ipv4_lpm MyIngress.ipv4_forward 10.0.1.0/24 => 00:00:00:00:00:02 1
```

Para los dos últimos comandos, como ejemplo, las salidas serían:

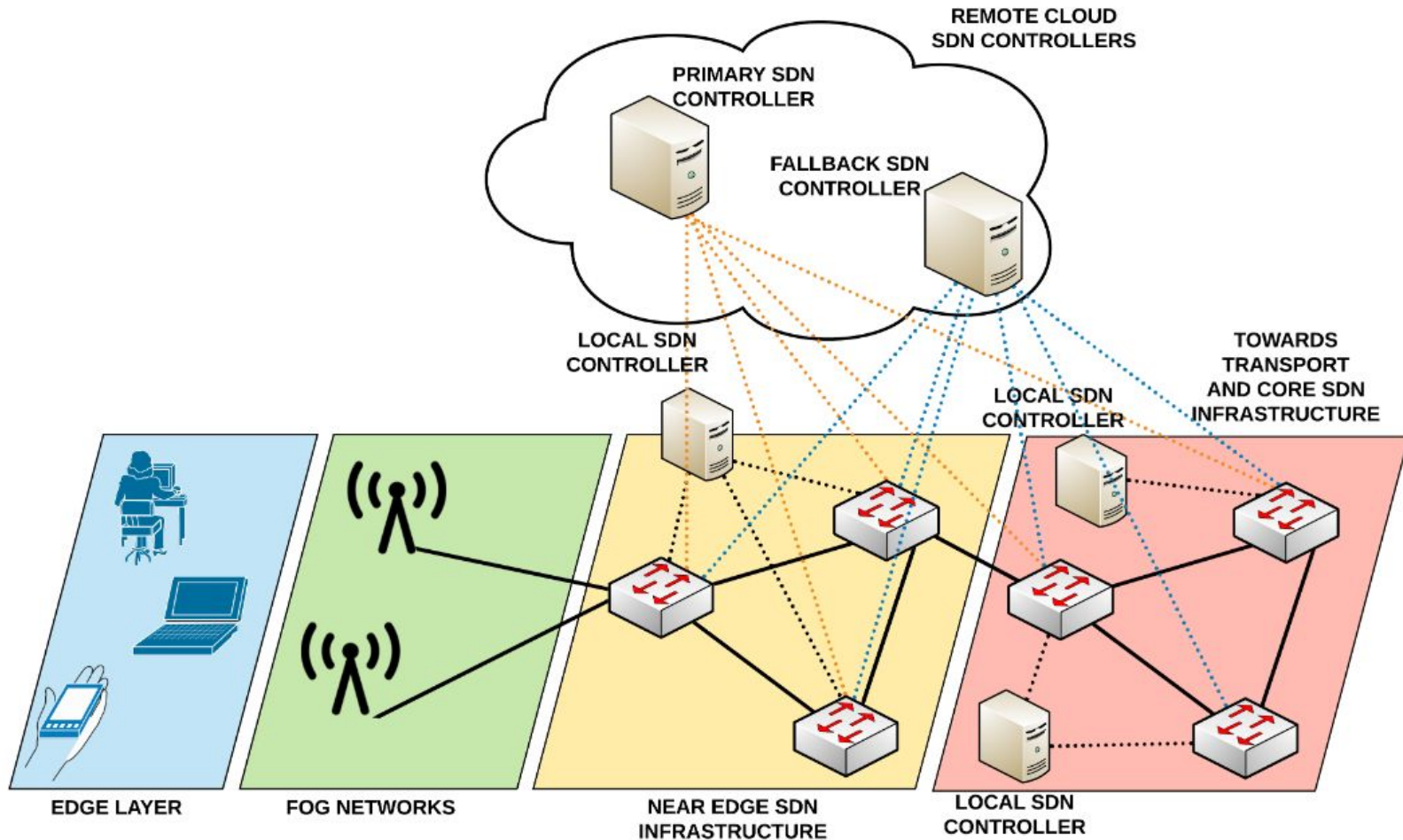
```
RuntimeCmd: table_add MyIngress.ipv4_lpm MyIngress.ipv4_forward 10.0.0.0/24 => 00:00:00:00:00:01 0
Adding entry to lpm match table MyIngress.ipv4_lpm
match key:          LPM-0a:00:00:00/24
action:             MyIngress.ipv4_forward
runtime data:       00:00:00:00:00:01    00:00
Entry has been added with handle 0
```

```
RuntimeCmd: table_add MyIngress.ipv4_lpm MyIngress.ipv4_forward 10.0.1.0/24 => 00:00:00:00:00:02 1
Adding entry to lpm match table MyIngress.ipv4_lpm
match key:          LPM-0a:00:01:00/24
action:             MyIngress.ipv4_forward
runtime data:       00:00:00:00:00:02    00:01
Entry has been added with handle 1
```



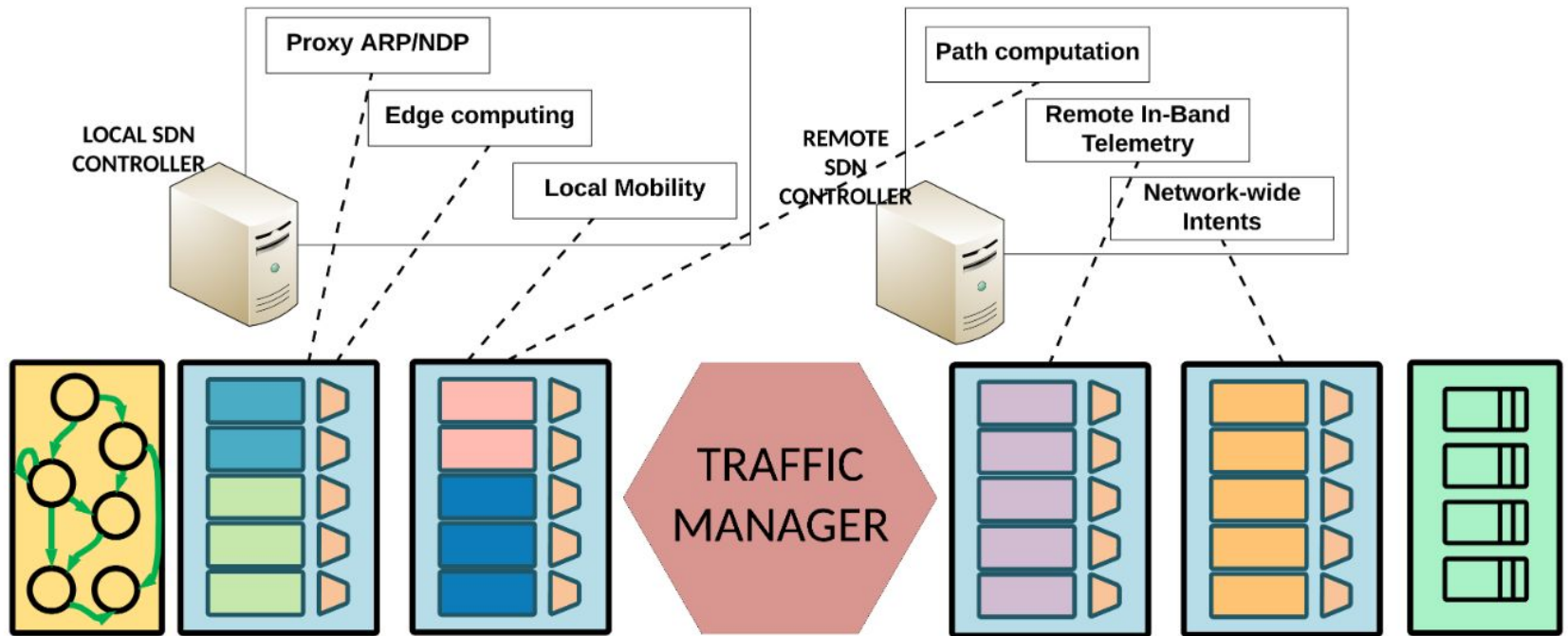
# Ejemplos de uso actual de P4 (I)

## 3 P4 and P4Runtime as a tool for fog computing



# Ejemplos de uso actual de P4 (II)

## Next-Generation SDN and Fog Computing



**Figure 4** Local and remote control planes managing different match-action parts of a P4 pipeline.



