ETH zürich

Networked Systems
ETH Zürich — seit 2015

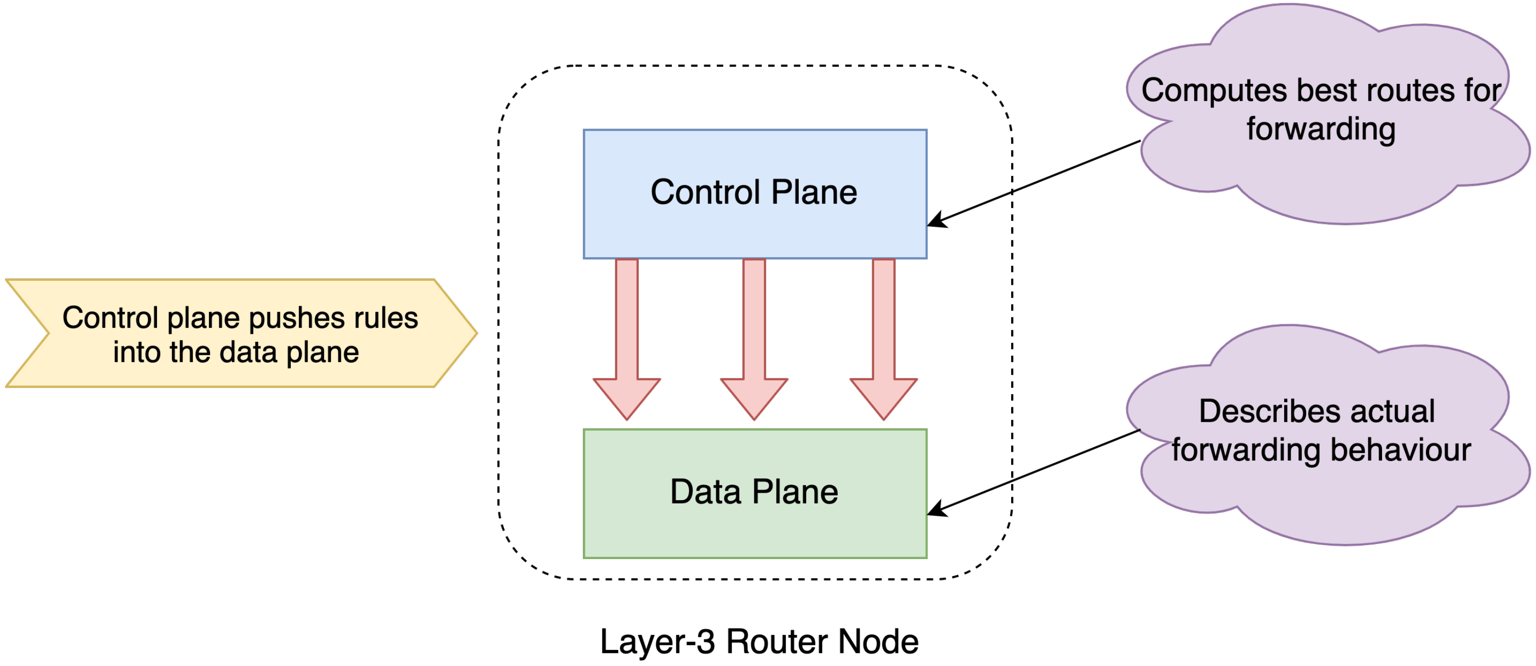
# Towards a new framework for integration of network planes


**Siddhant Ray**


Semester Thesis Presentation

25th June 2021, Zürich

# Traditional Layer-3 router:



# Programmable data planes : Why should we care?

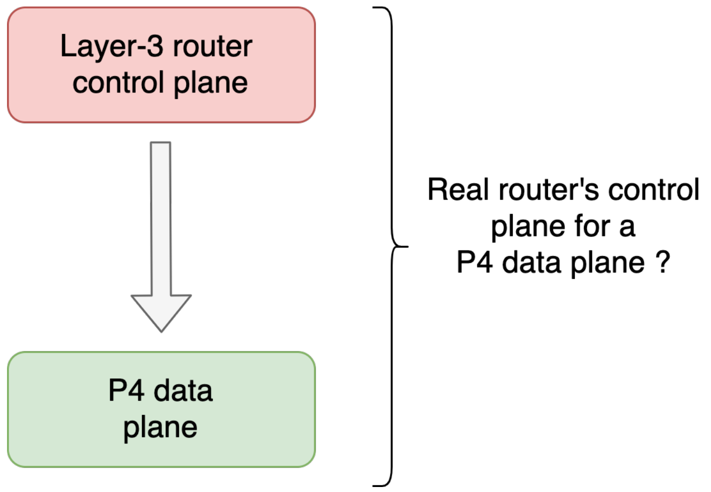
- Data planes have usually been static for eg. *NIX kernels
- The control plane tells the data plane how to forward packets
- With programmable data planes , we can
  - Make intelligent forwarding decisions at the data plane level
  - Greater packet level control over forwarding
  - Send feedback to control plane, custom forwarding etc.

# Programmable data planes : Challenges

- Existing P4 data planes : do not have supporting routing protocols
- Need a custom controller : compute routes, populate data plane
- Limited : Doesn't provide all properties of actual routing

So, can we do better?

Can we integrate existing real routing protocols to work with P4 data planes?



# In this project, we present :

- A new Super-Node for Layer-3 forwarding
- Each Super-Node has :
  - An FRR control plane
  - A P4 programmable data plane

- We create and run :
  - FRR control planes in individual Linux namespaces
  - P4 data planes in a common root namespace
  - Protocols OSPF and iBGP for Super-Nodes in the same Autonomous System (AS), eBGP on Super-Nodes at the border of an AS.
  - Load balancing for end-to-end forwarding

# Framework: Mininet [1] and p4-utils [2]

- Mininet :

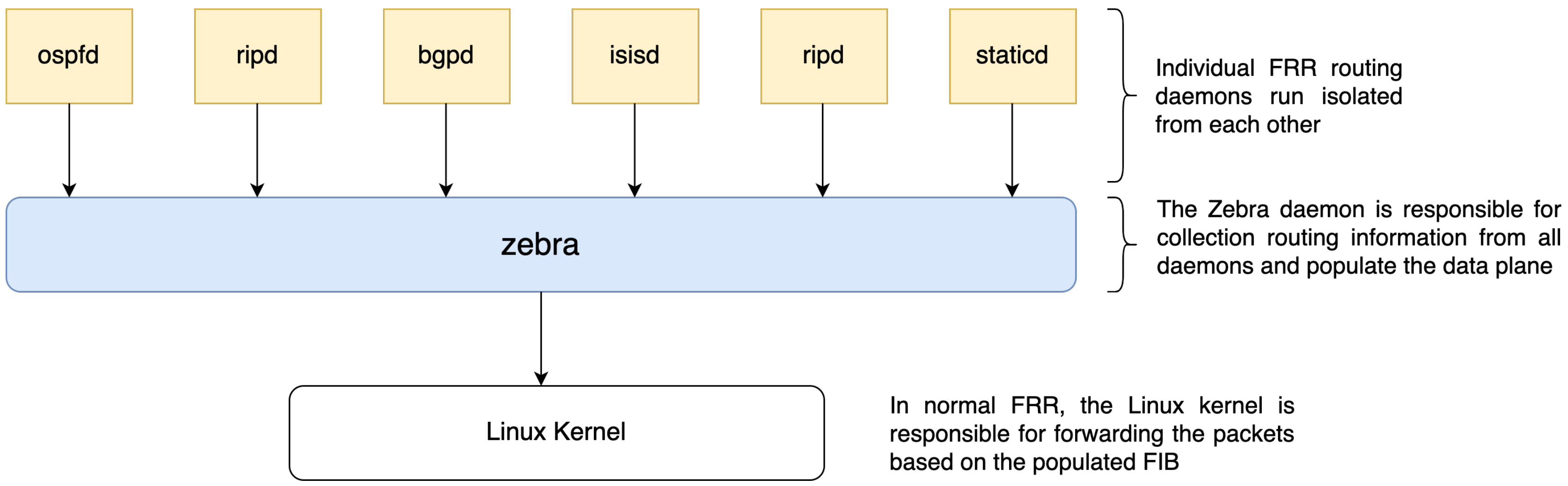
- A rapid, prototyping tool which helps create emulated networks inside a single OS kernel
- Uses the Linux kernel and network stack, nodes can be in the kernel or custom network namespaces.
- Our control plane routers run as custom Mininet nodes.

- p4-utils :

- Used to build and develop P4 networks. It is an extension to Mininet.
- The P4 data plane is implemented as a custom Mininet switch.

[1] http://mininet.org/
[2] https://github.com/nsg-ethz/p4-utils

# Control Plane : FRRouter [3]




FRR suite architecture


[3] https://frrouting.org/

# Data plane : P4 bmv2 Switch [4]

A P4 data plane consists of a :

- Parser :

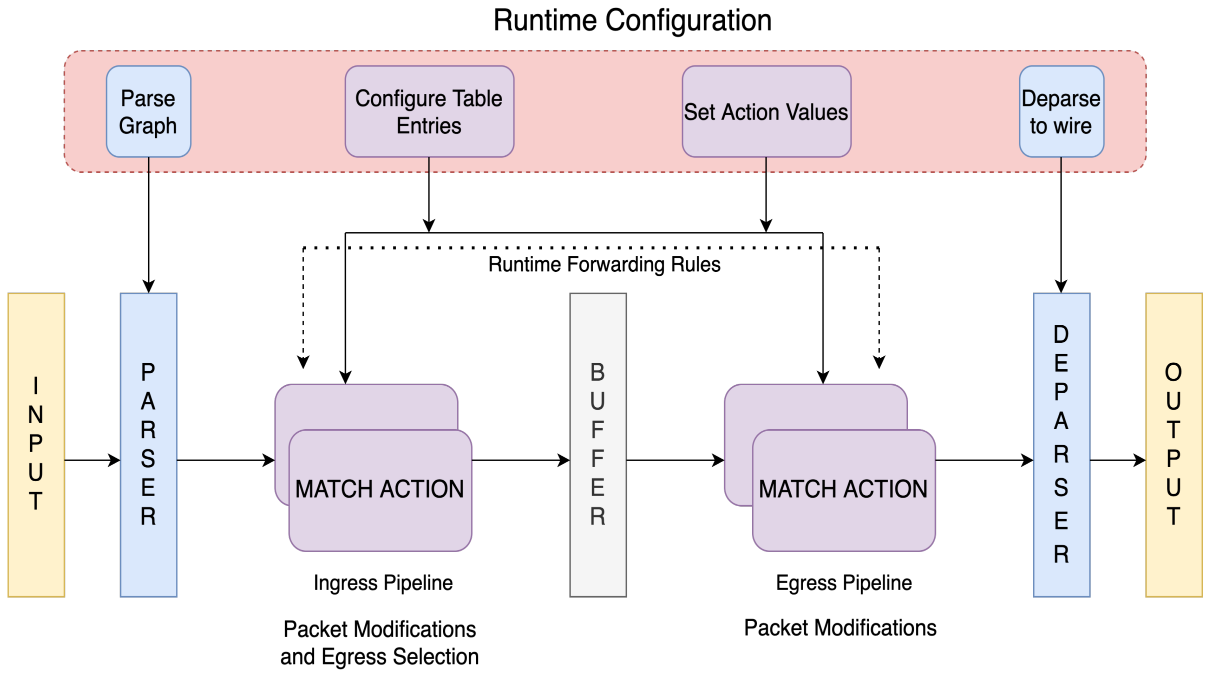
Extract information from the packet header

- Match-Action :

Match on fields to set actions to decide forwarding

- Deparser :

Add parsed headers back at the end




P4 data plane pipeline

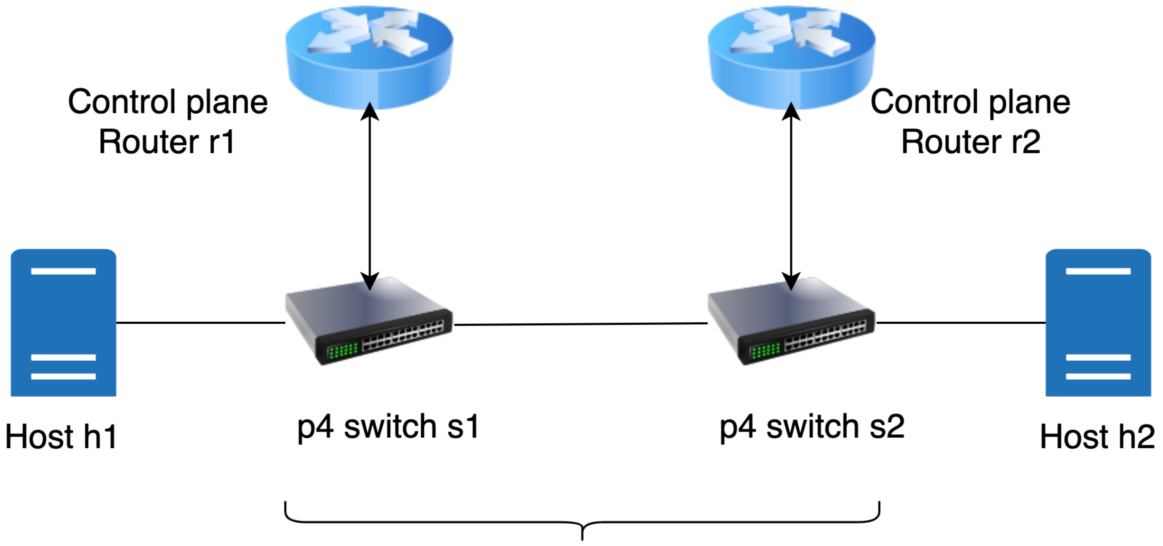
[4] https://p4.org/index.html

# Initial Super-Node design :

- FRR control plane
- P4 data plane

The control planes routers do not have real links to each other, they have non-connected fake interfaces.

The P4 data planes act as "transparent forwarding" nodes for the control planes.




Super-Node for routing and forwarding. Each such Super-Node consists of an FRR control plane and p4 data plane.

# However there is a problem..

- FRR routers do not send packets on interfaces with no real links.
- FRR routers drop packets if not received on the intended interface.

Two options :

- Add dummy links to nothing , capture packets and reinject to correct interface.

  (This approach is very slow)
- Add dummy links to the P4 data plane, no need to reinject packets.

  (Fast but more interfaces on the P4 data plane)

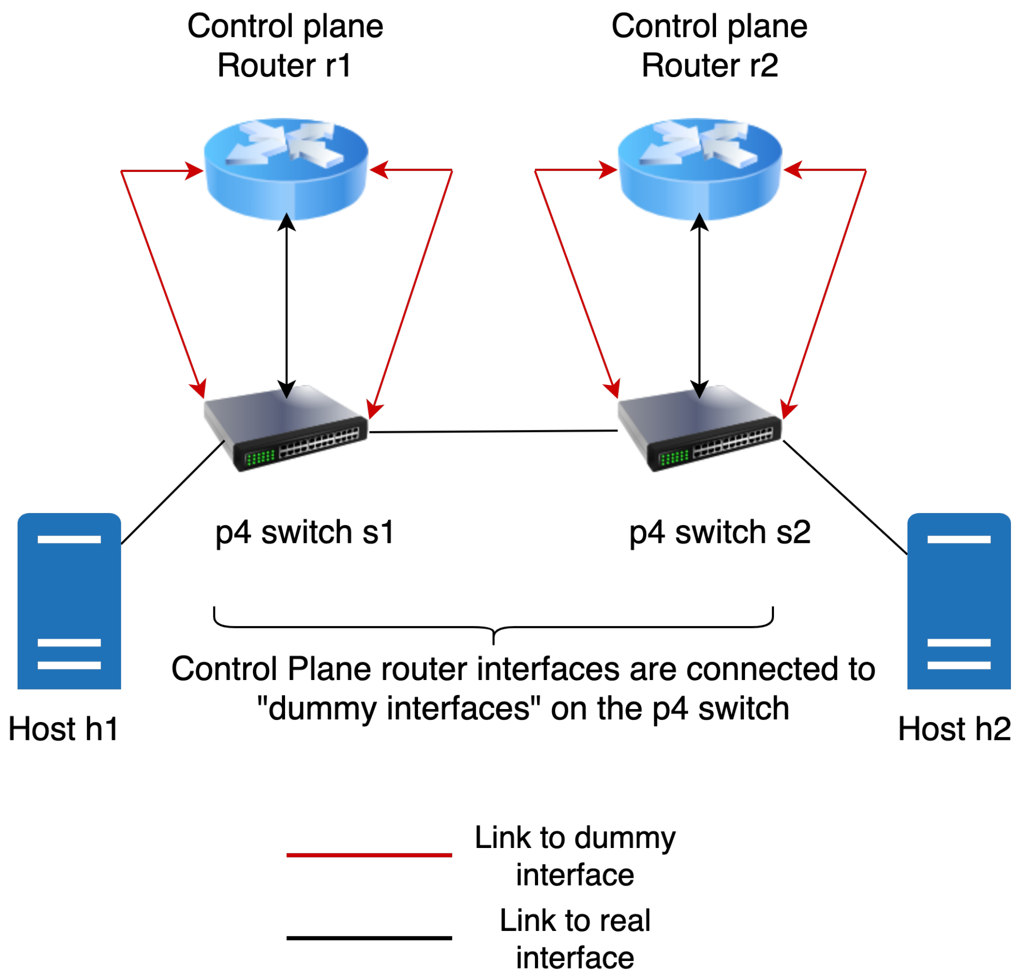
We choose the second option for our design.

# Final Super-Node design :

- Control planes routers have dummy links to each P4 switch.

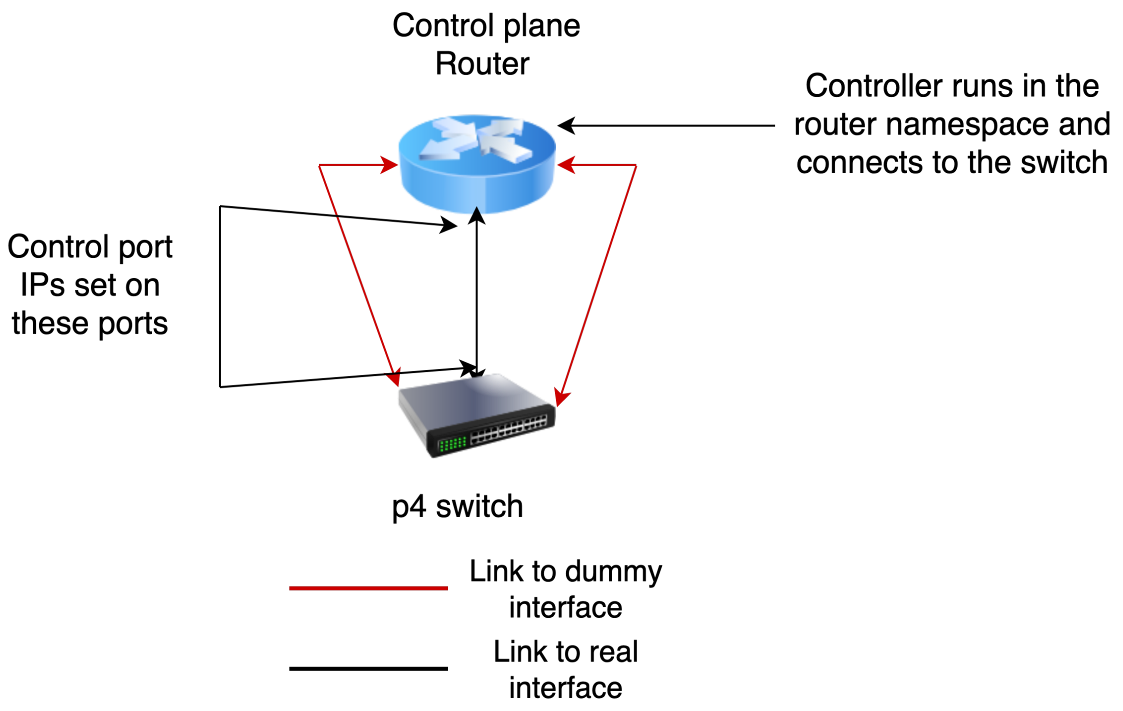
- The FRR control plane router is fooled to believe it has a real connection to the other.

- Control plane routers send and receive the correct packets.



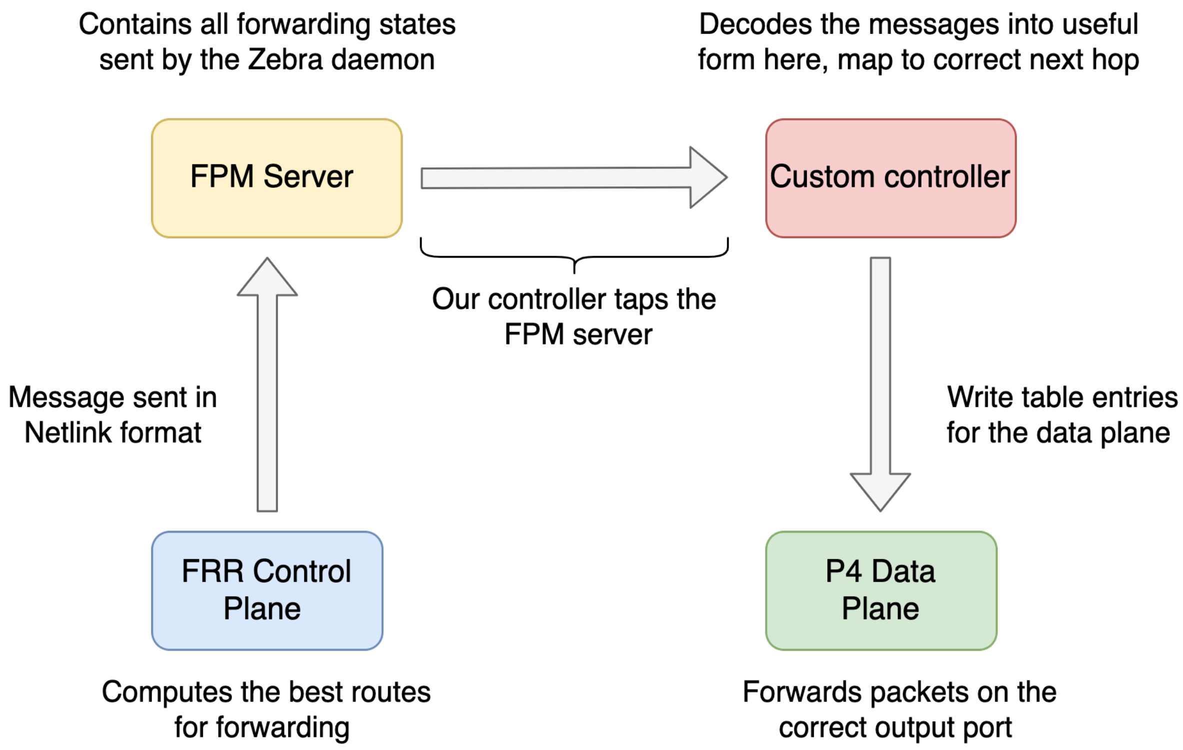
# Handling data plane updates :

- A Python controller running in each control plane router namespace.
- A P4 program running in the P4 data plane.
- The P4 data plane forwards control plane packets from dummy links to the correct output interface. (for OSPF, BGP etc.)
- Control plane routers now calculate best routes to all destinations.



# Handling data plane updates :

- FRR has an FPM server which collects a complete copy of the forwarding table.
- FPM by itself doesn't push entries into the data plane.
- Our controller is dynamic: updates routes into the data plane as soon as it arrives.
- Our programmable data plane ensures: only IPv4 packets are forwarded using FPM routes.



# Understanding FPM messages :

## Sample decoded FPM message :

{'family': 2, 'dst_len': 24, 'src_len': 0, 'tos': 0, 'table': 254, 'proto':

11, 'scope': 0,'type': 1, 'flags': 0, 'attrs': [('RTA_DST', '10.2.0.0'),

('RTA_PRIORITY', 11), ('RTA_GATEWAY', '10.0.1.2'), ('RTA_OIF', 26715)],

'header': {'length': 60, 'type': 24, 'flags': 1025, 'sequence_number': 0,

'pid': 0}}

Message to reach host h2 from Super-Node r1-s1 (no multipath configured).

## What our controller does:

Check family :

2 > Means IPv4 route

Check destination prefix :

24 > Inside AS, 8 > To other AS

Extract destination IP prefix :

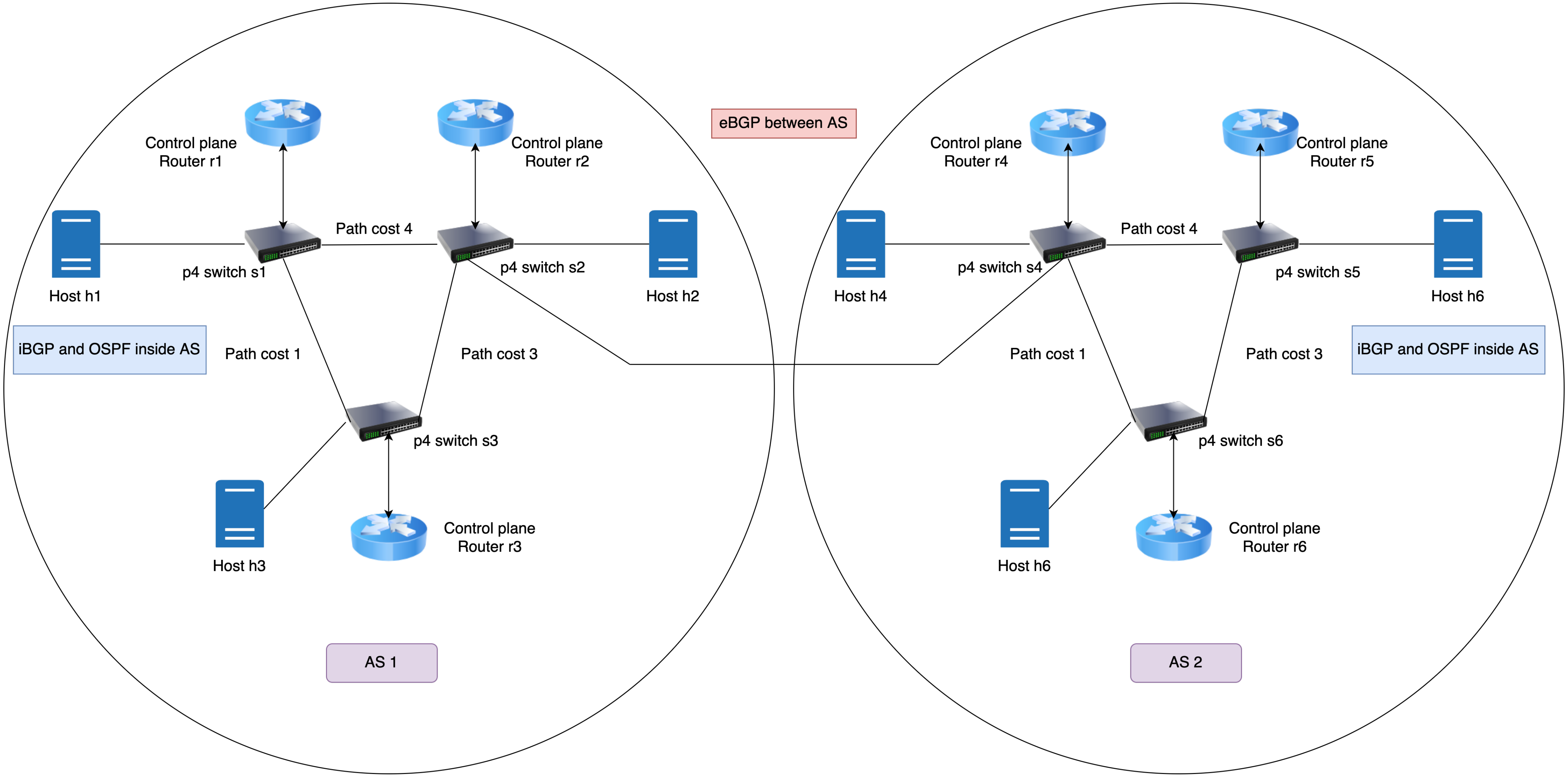
10.2.0.0/24

Map to correct output port :

26715 > 2

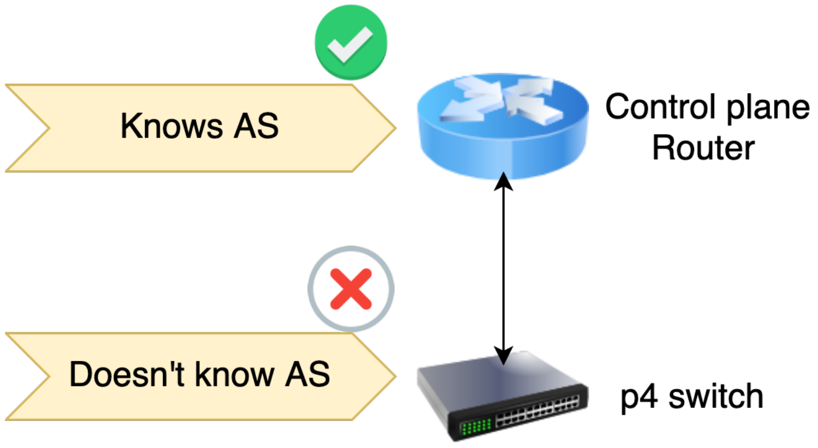
Write correct table entry for IPv4 forwarding

# Testing Topology :



# Feature : Running OSPF and BGP

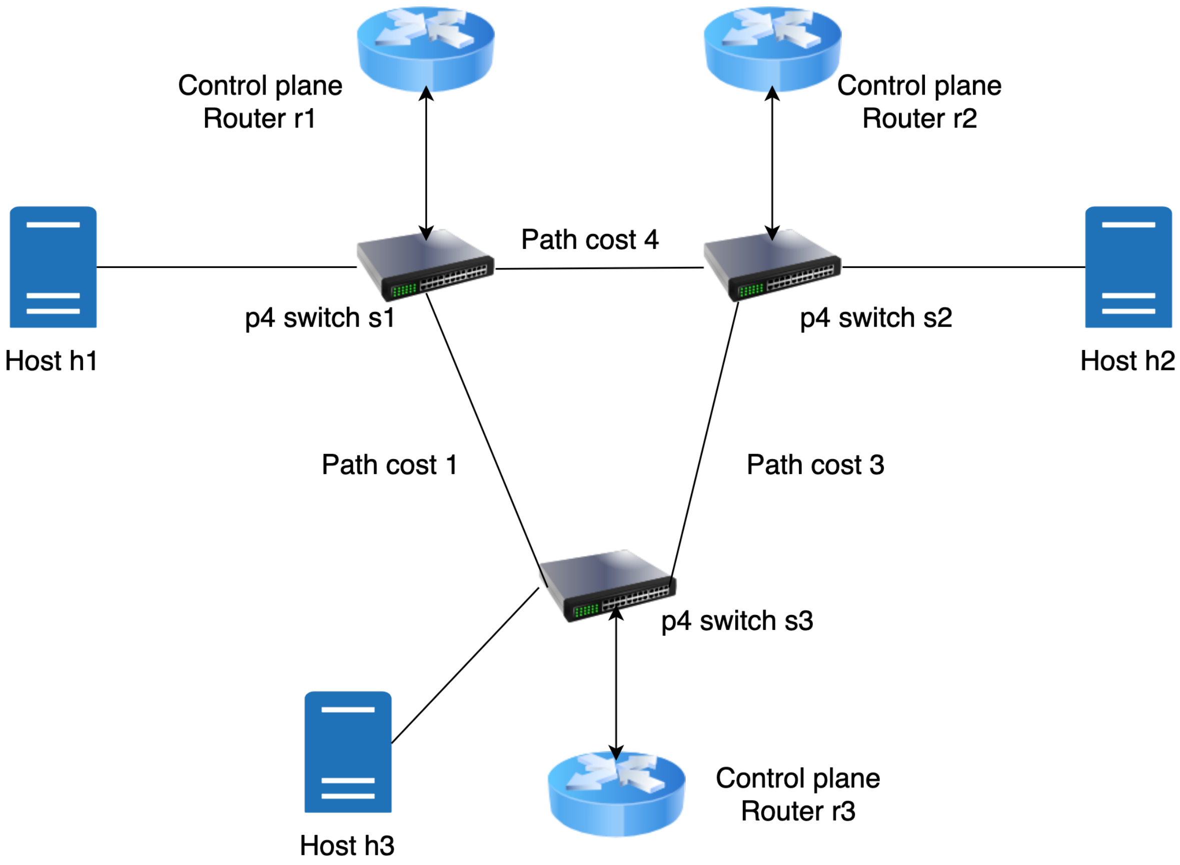
- AS - only known to control plane routers, not to the P4 data plane
- Super-Nodes at the border :  run eBGP
- Interior Super-Nodes : run iBGP and OSFP
- eBGP learnt routes are distributed over iBGP (preferred) or OSPF.
- Sustained TCP connection over the eBGP peers for connected BGP state
- OSPF – updates internal AS routes
- BGP – updates external AS routes



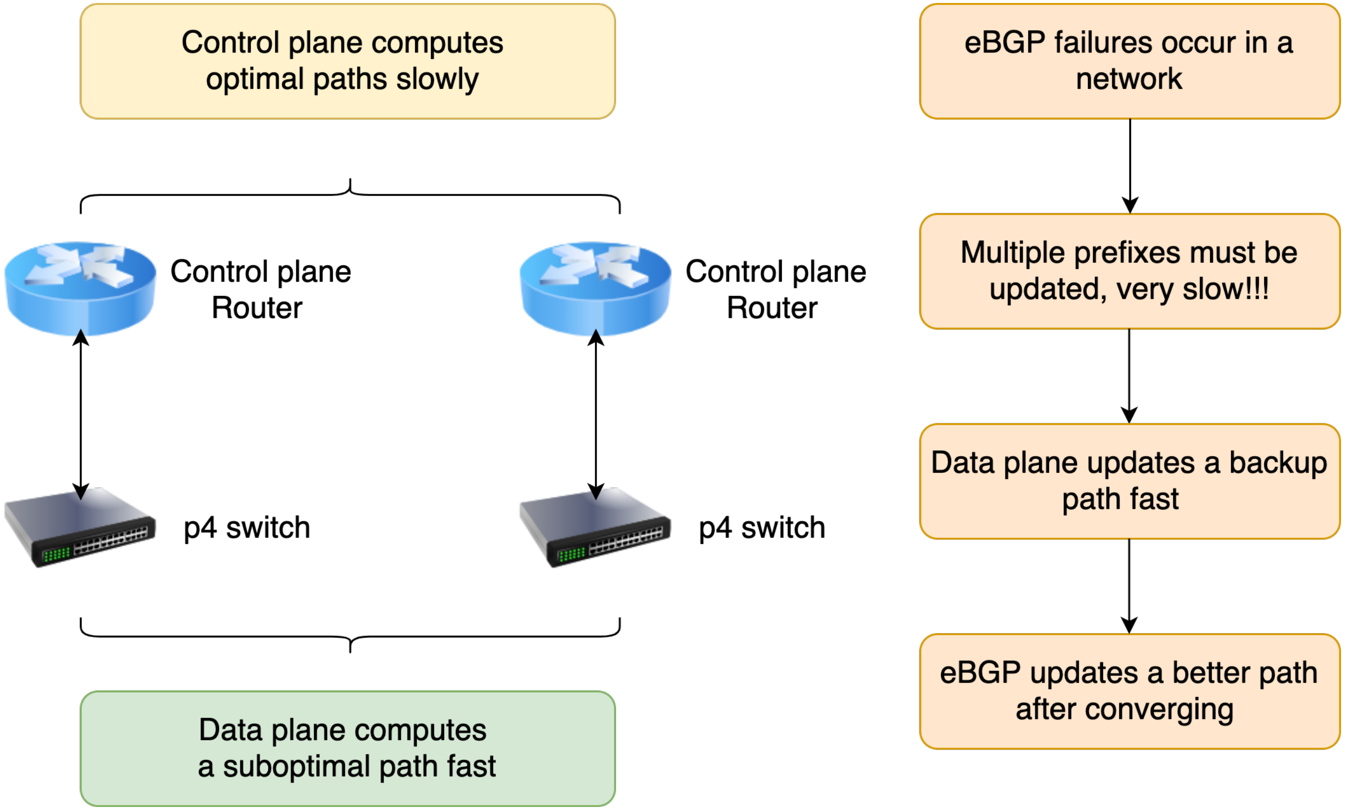

Super-Node setup

# Feature : Load Balancing using ECMP

- FRR identifies multipath routing on equi-cost paths.
- P4 program in the data plane calculates ECMP hash values on the standard 5-tuple.
- Controller checks for multi-path and populates the ECMP table for multi-path forwarding.
- Different ECMP groups for different IP destination prefixes (per AS) in order to have multi-path across AS.



# What can we do now?



# Outlook :

- Limited to adding dummy links externally via Linux :
  Currently no support for multiple, parallel links via Mininet or p4-utils
- Need to rely on dummy links :
  FRR doesn’t send packets on non-connected interfaces
- Cannot stop FRR from populating the kernel data plane
  P4 data planes can perform “transparent forwarding” on control plane packets
- Control plane must forward a lot of packets via the P4 data plane
  Programmability ensures complete control to have correct forwarding

# Time for questions

Siddhant Ray
MSc. in Electrical Engineering and Information Technology
sidray@student.ethz.ch

D-ITET
ETH Zurich