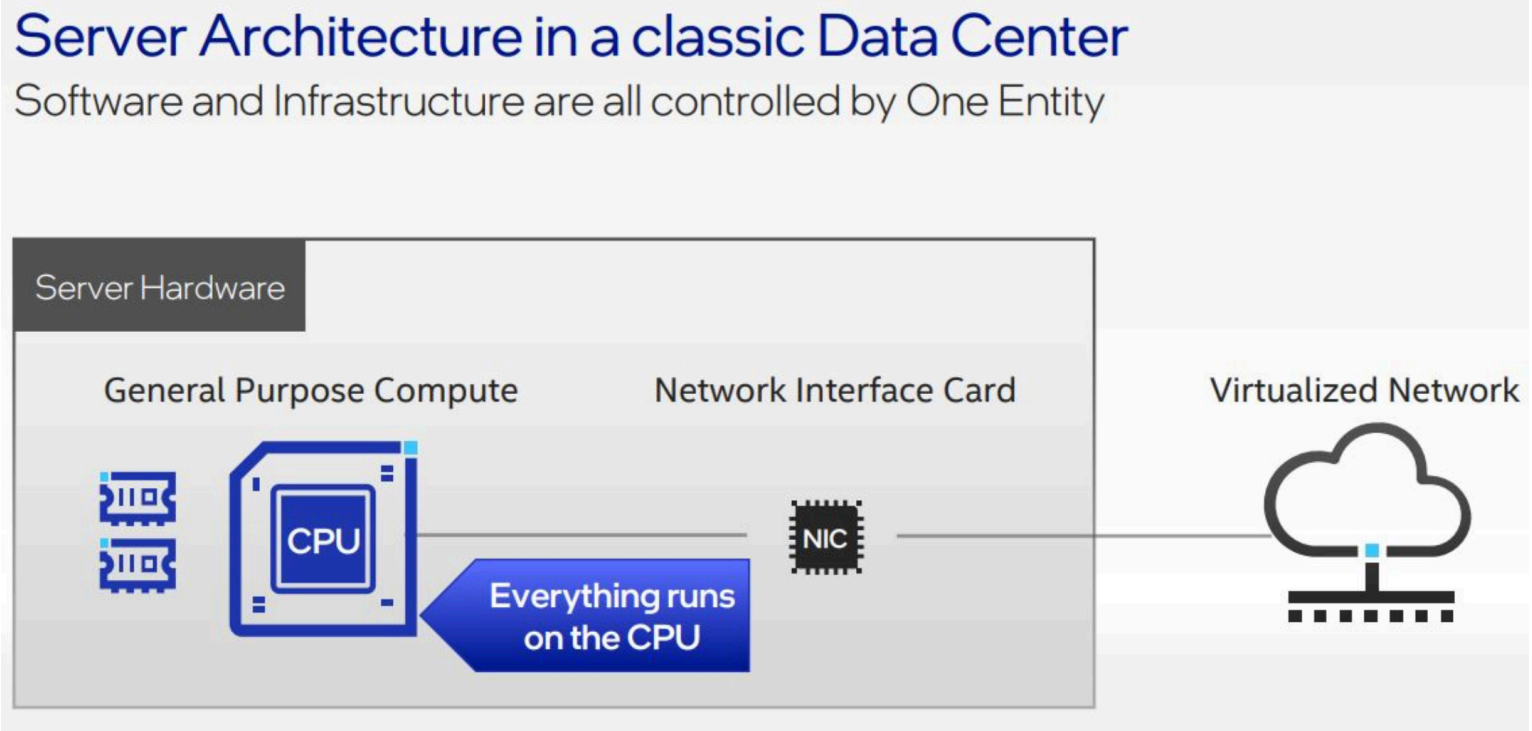
THE INTEL INFRASTRUCTURE PROCESSING STRATEGY

Software Networking

Politecnico
di Torino

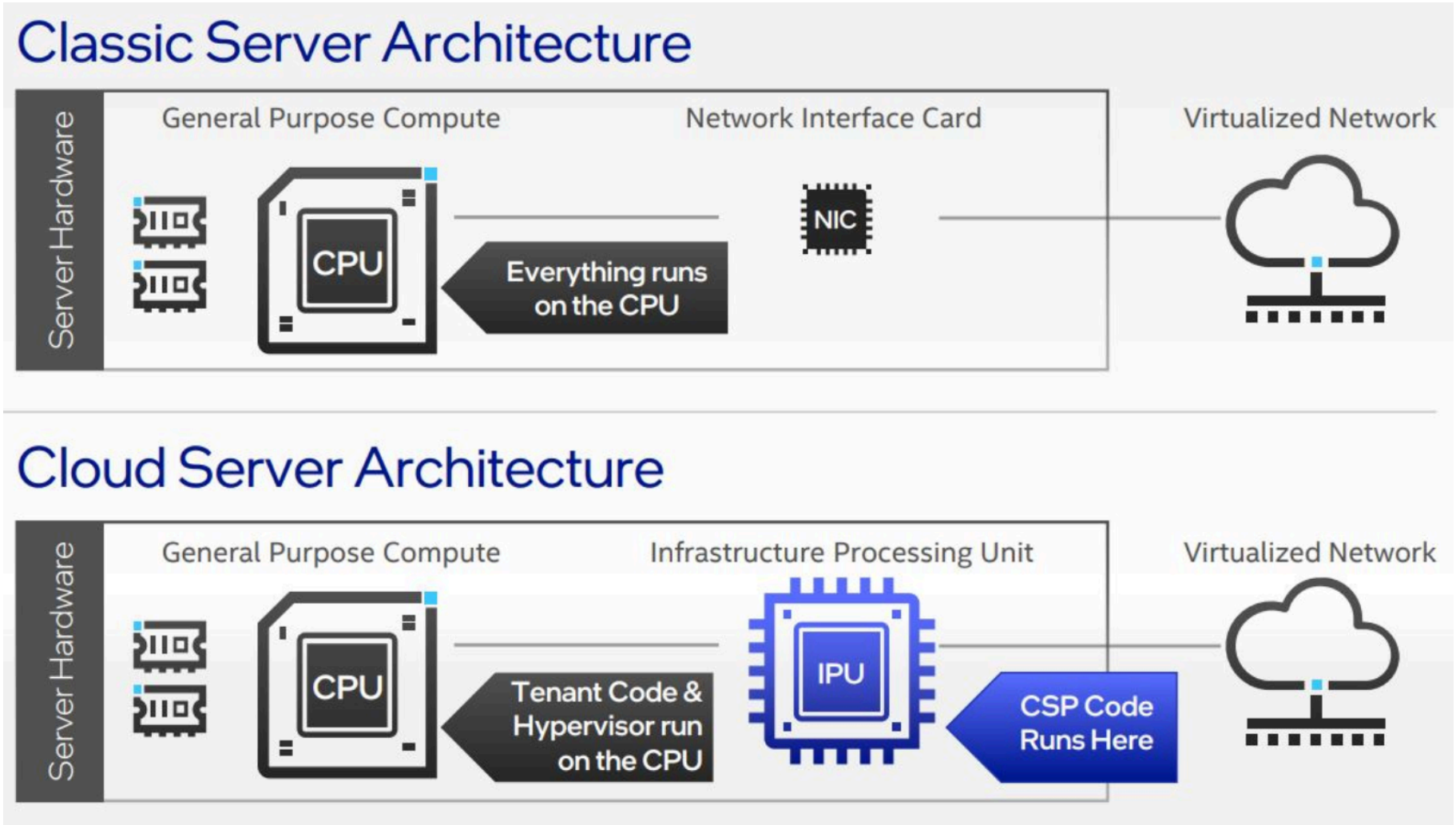
Introduction

CLASSIC DATA CENTER



[1] https://community.intel.com/t5/Blogs/Tech-Innovation/Data-Center/The-IPU-A-New-Strategic-Resource-for-Cloud-Service-Providers

Intel© Processing Unit



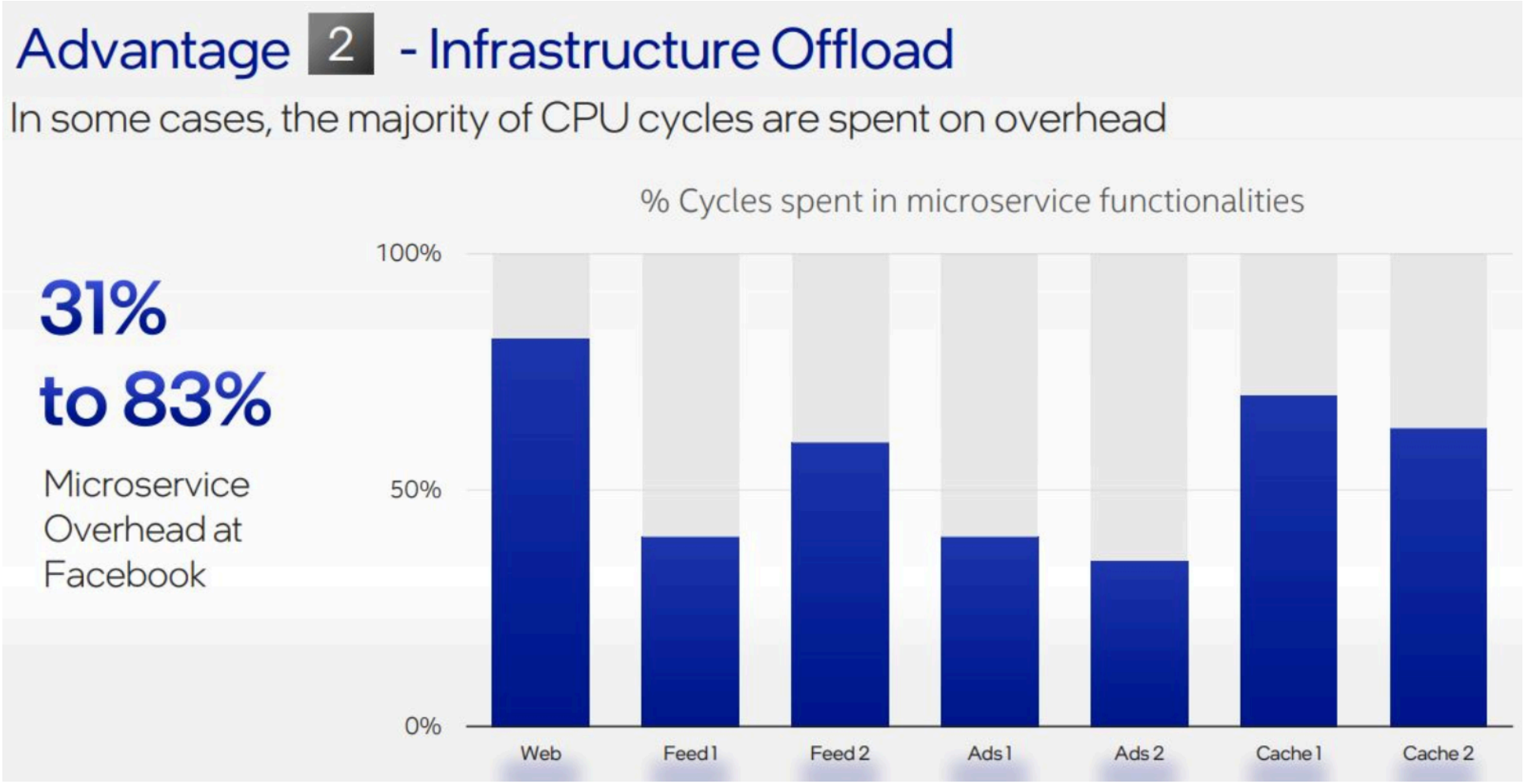
[2] https://community.intel.com/t5/Blogs/Tech-Innovation/Data-Center/The-IPU-A-New-Strategic-Resource-for-Cloud-Service-Providers

IPU ≠ SmartNIC

«A SmartNIC is a programmable network adapter that can accelerate infrastructure applications, however, unlike an IPU it does not provide offload capability to run the entire infrastructure stack and therefore does not give the service provider an extra layer of security and control, enforced in hardware.»

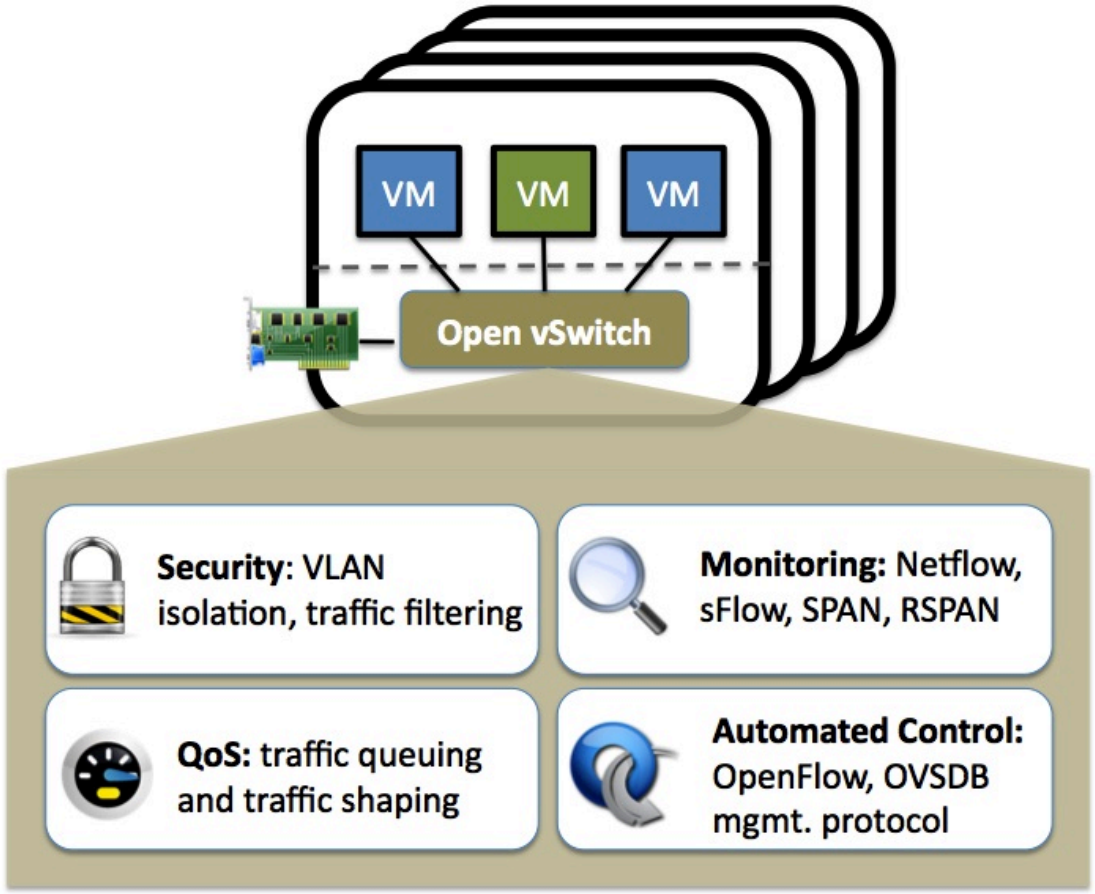
[3] IPU Based Cloud Infrastructure White Paper

Microservices overhead



[4] https://community.intel.com/t5/Blogs/Tech-Innovation/Data-Center/The-IPU-A-New-Strategic-Resource-for-Cloud-Service-Providers

Open vSwitch



The device used in the Intel solution is the P4 OvS. Open vSwitch is a multilayer software switch licensed under the open source Apache 2 license.

Open vSwitch is well suited to function as a virtual switch in VM environ- ments. In addition to exposing standard control and visibility interfaces to the virtual networking layer, it was designed to support distribution across multiple physical servers. Open vSwitch supports multiple Linux-based virtualization technologies including KVM, and VirtualBox .

The bulk of the code is written in platform-independent C and is easily ported to other environments.

Main Components

The main components of this distribution are:

- **ovs-vswitchd**, a daemon that implements the switch, along with a com- panion Linux kernel module for flow-based switching.
- **ovsdb-server**, a lightweight database server that ovs-vswitchd queries to obtain its configuration.
- **ovs-dpctl**, a tool for configuring the switch kernel module.
- Scripts and specs for building RPMs for Red Hat Enterprise Linux and deb packages for Ubuntu/Debian.
- **ovs-vsctl**, a utility for querying and updating the configuration of ovs- vswitchd.
- **ovs-appctl,** a utility that sends commands to running Open vSwitch dae- mons.

P4 Programming

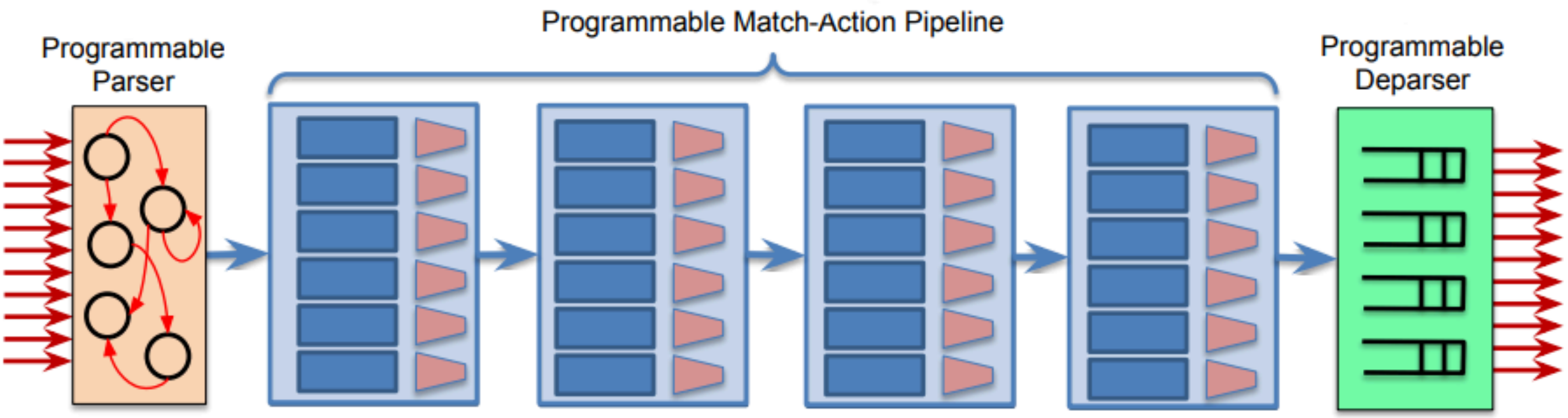
Motivation

- Domain Specific Language
- Protocol Agnostic Language
- Single code base for multiple target

It brings several development improvements and increases maintainability, security and portability.

It removes some security issues that come with the non-specific domain languages, like C/C++

Architecture



P4 and eBPF

There are a bunch of projects working on the interoperability between P4 and eBPF

- p4c-ebpf
- p4c-xdp
- p4c-ubpf

P4 is more restricted than eBPF but it guarantees a better portability and security

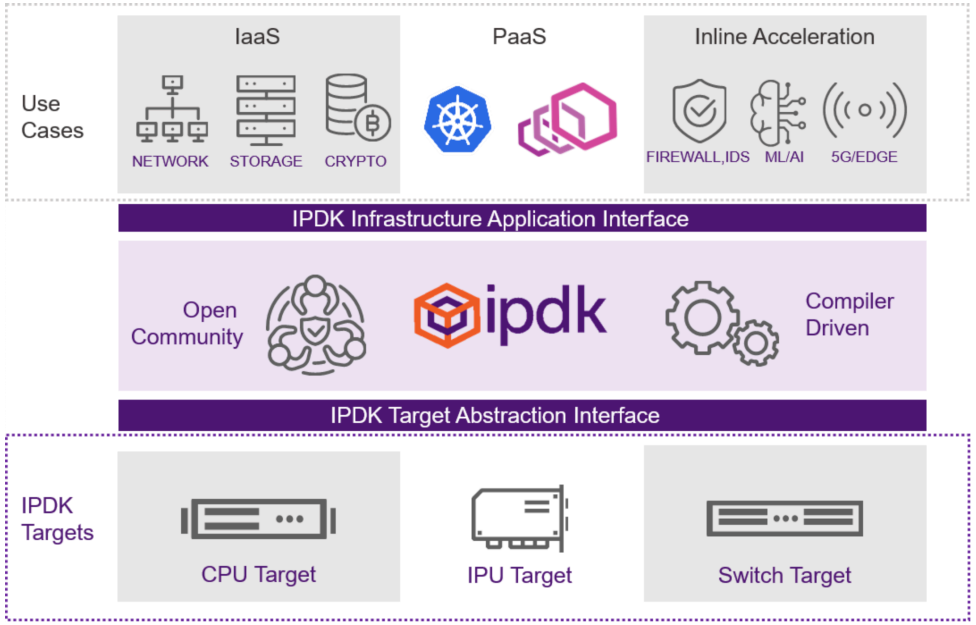
eBPF is more flexible than P4 so cross compile eBPF to P4 is more challenging

IPDK

What is?

Target Agnostic Open-Source Development Kit which runs on CPU, IPU, DPU, DPDK

It creates an abstraction layer between the target and the implementation to offload



What is?

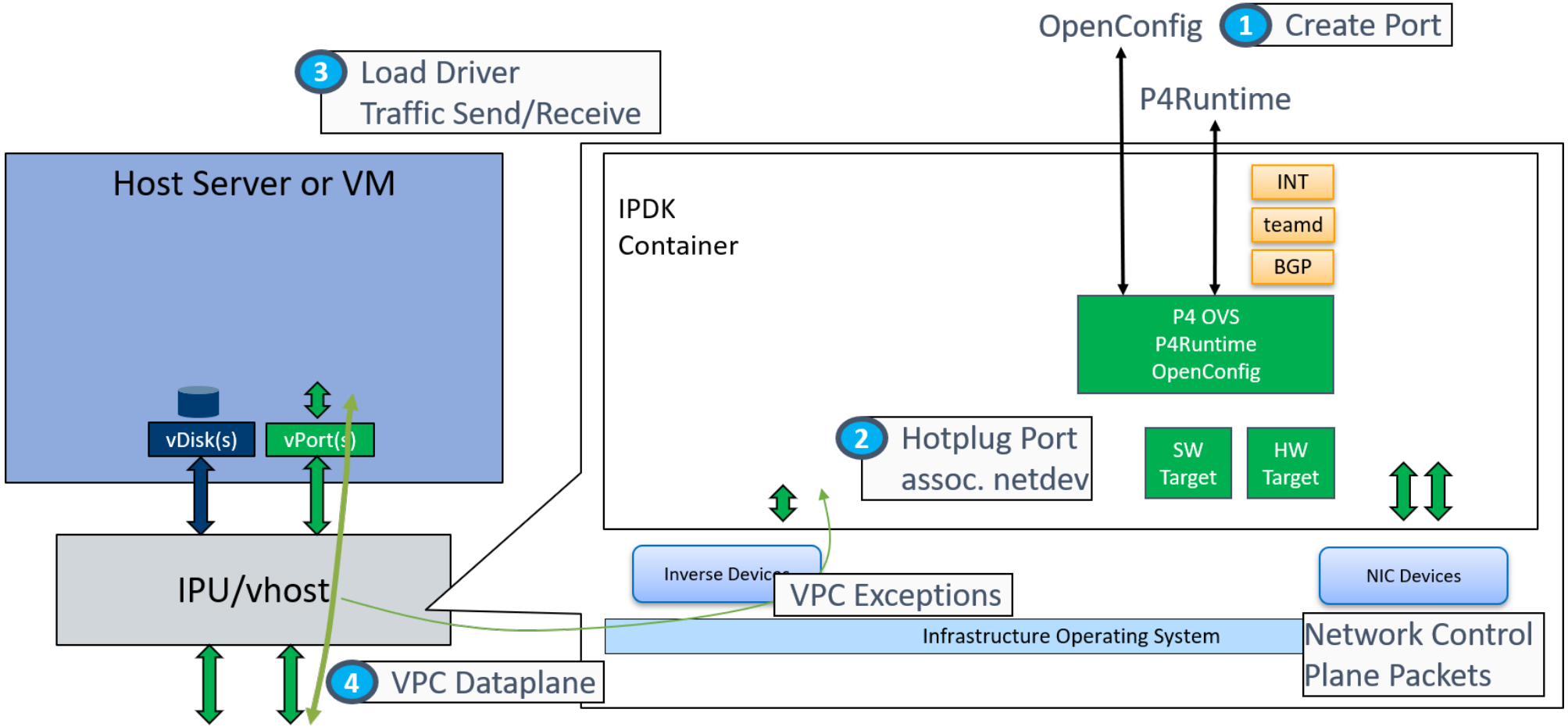
It provides us with a good number of recipe ready to use to build Virtual Networking and Virtual Storage

In the Development Kit we can find some CLIs which help us to perform common action like create interfaces, add/remove rules and manage the target

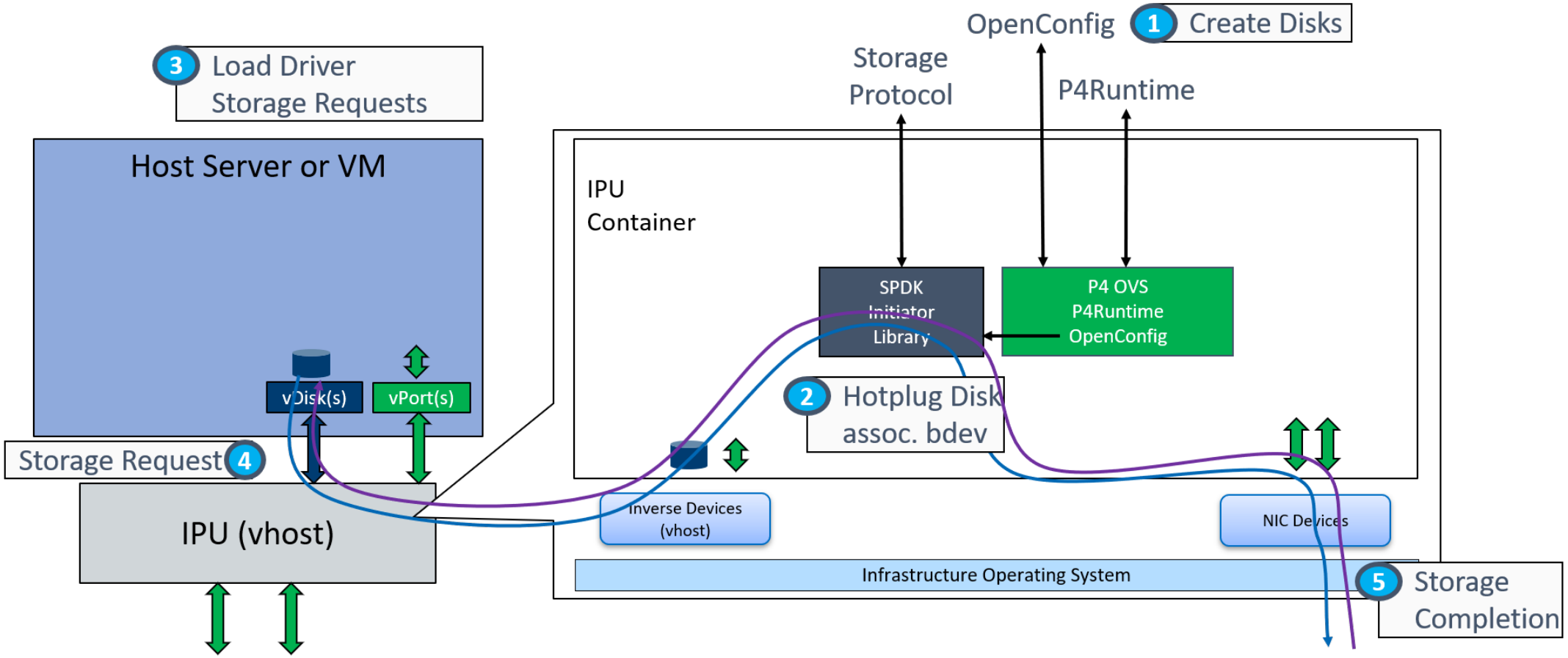
Currently only *virtio-net* is the virtual networking device while there are 3 different virtual storage devices:

- virtio-blk
- virtio-scsi
- NVMe

Virtual Networking

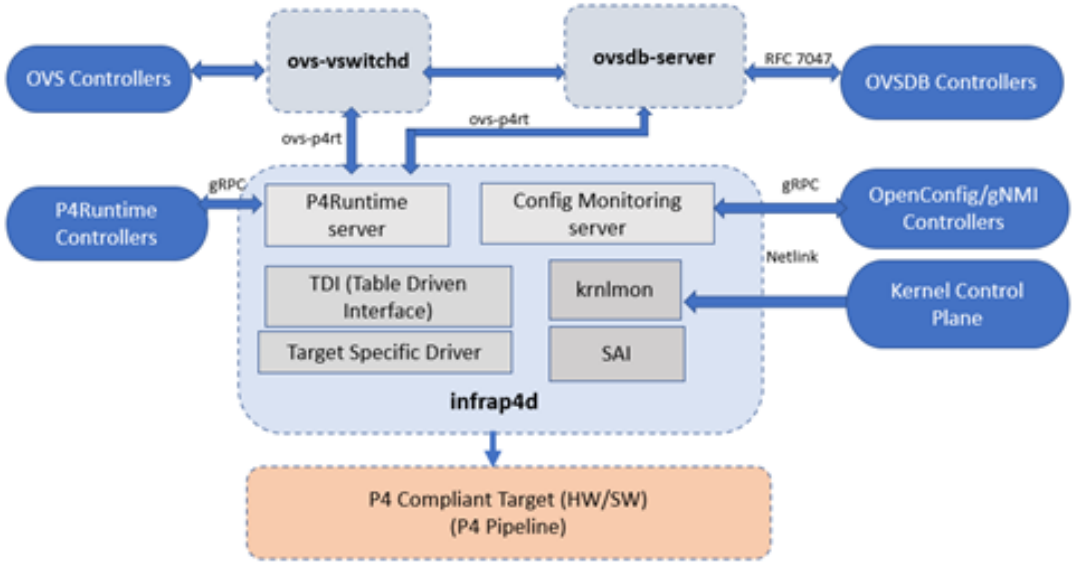


Virtual Storage



Internals

The main process is infrap4d (Stratum) which wraps some gRPC servers to manage the target



Internals

We always find 2 gRPC servers in the offloaded solution, in each target:

- **P4Runtime**: queryable by ovs-p4rt CLI to manage the data plane
- **gNMI**: gRPC Network Management Interface to manage the network configuration like vPorts

Each target may have additional servers, for example DPDK includes ovs-p4rt that is used to add/modify/delete P4 forwarding table's entries.

There are other components:

- **TDI - Table Driven Interface**: Set of APIs that enable configuration and management of P4 programmable devices. Each device can choose to use its own backends. Stratum uses this interface to talk to the target-specific driver

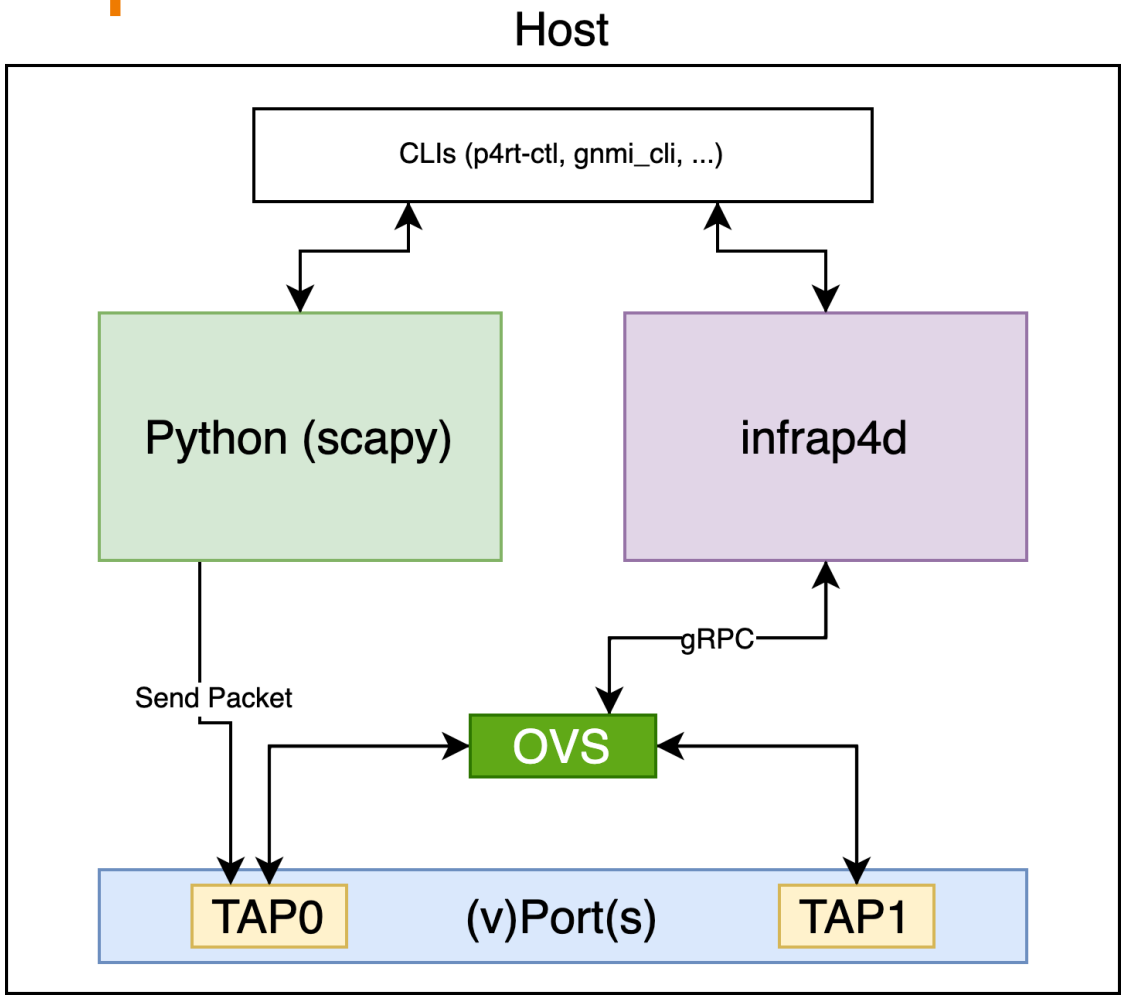
- **krnlmon - Kernel Monitor**: receives RFC 3549 messages from the Linux Kernel over a Netlink socket when changes are made to the kernel networking data structures

- **SAI - Swtich Abstraction Interface**: Switch Abstraction Interface (SAI) defines a vendor-independent interface for switching ASICs.

Example

The example will show:

- A P4 program to perform IPv4 forwarding
- Infrap4d running
- How to create vPorts with gNMI CLI
- How to create forwarding rules using p4rt-clt CLI
- How to send packets in a test environment with scapy



Politecnico
di Torino