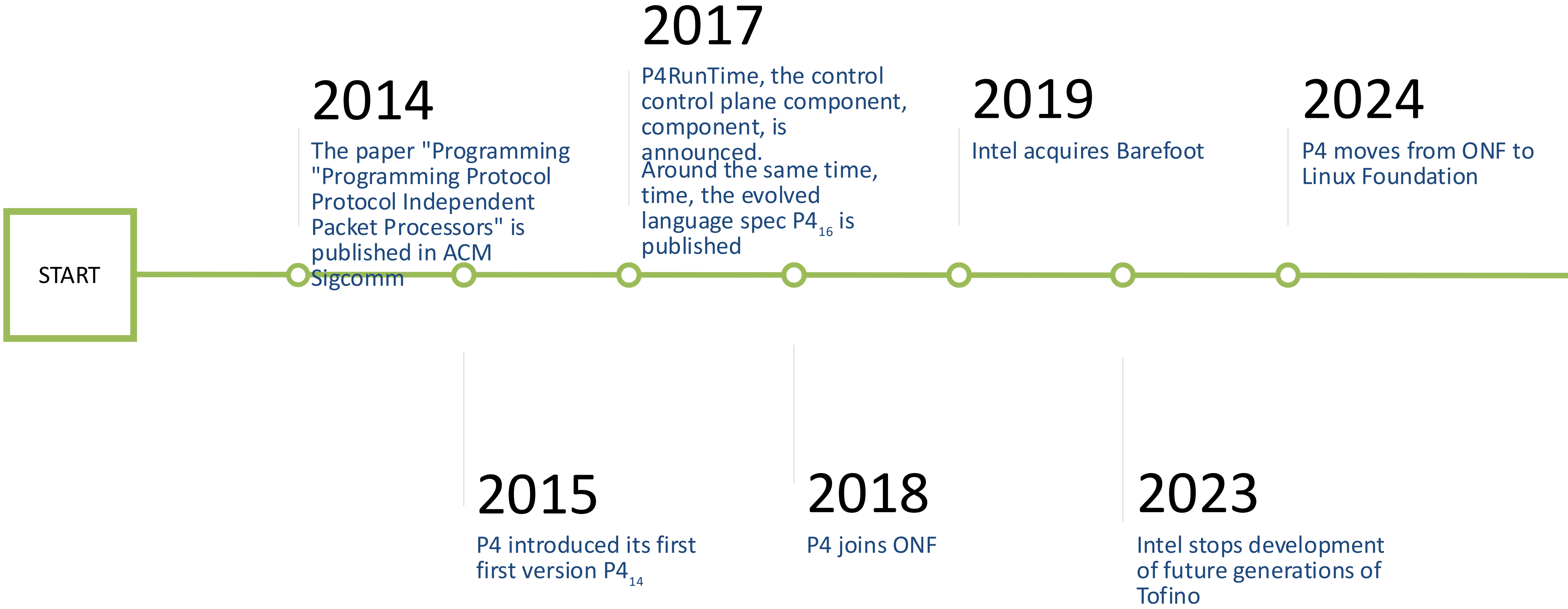
OCTOBER 3
P4
WORKSHOP 2024

The Past, Present and Future of P4

Deb Chatterjee



START

2014

The paper "Programming "Programming Protocol Protocol Independent Packet Processors" is published in ACM Sigcomm

2015

P4 introduced its first first version $P4_{14}$

2017

P4RunTime, the control control plane component, component, is announced.
Around the same time, time, the evolved language spec $P4_{16}$ is published

2018

P4 joins ONF

2019

Intel acquires Barefoot

2023

Intel stops development of future generations of Tofino

2024

P4 moves from ONF to Linux Foundation

Cascade Glacier

Programmable vSwitch Offload

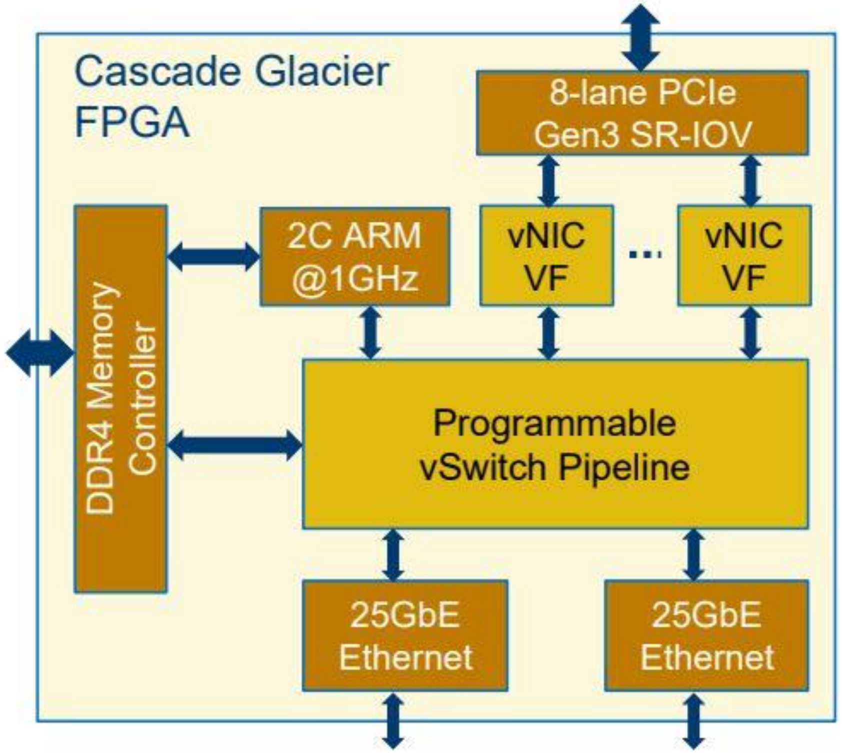
- *Customer vSwitches & Open vSwitch*
- 128Gb Internal Switch Fabric
- Millions of flows @ 12.5 Mpps x2
- Programmable encap/decap & NAT
- Connection tracking & ACLs
- Pipeline generated from P4

Embedded CPU Cores

- *vSwitch slow path & NIC management*

Virtio-net Hardware Offload

- *Supports Existing Linux & DPDK VMs*
- Up to 40Gb/sec, 12.5 Mpps
- 128 VFs, Multiple queues/VF
- TSO, CSUM, transmit rate limiters
- Live migration between HW & SW



Arria-10 25GbE Board

- Production by Q2'18
- Dual 25G SFP28
- PCIe gen3 x8
- DDR4 Memory
- <45W, passive heatsink

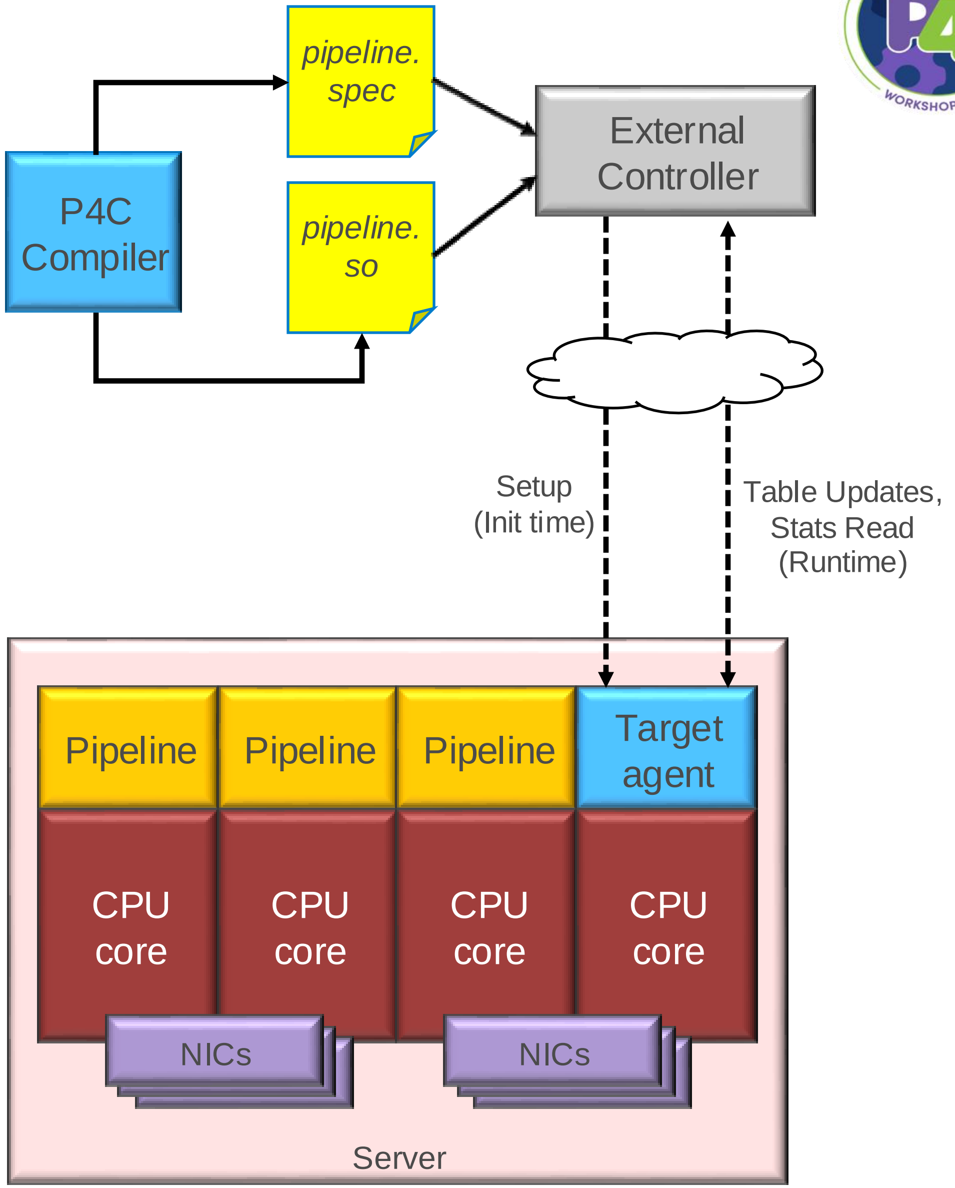
P4-DPDK

Step 1: Offline

- The P4C compiler (i.e. p4c-dpdk) translates the input *pipeline.p4* file into an intermediate representation *pipeline.spec* (plain text file) or *pipeline.so* (binary file).

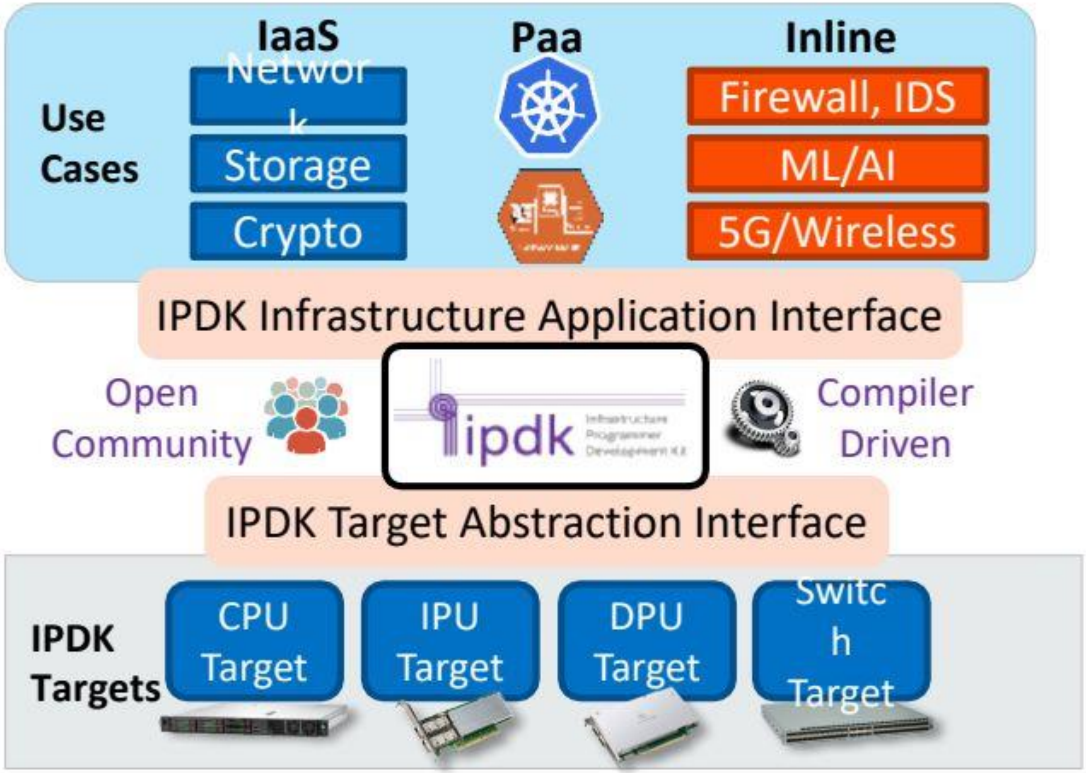
Step 2: Run-time

- The external controller connects to the target agent to load the P4 blob: in the baseline “interpreted” mode, this is the *pipeline.spec* text file; in the optimized “compiled” mode, this is the *pipeline.so* binary file.
- The target agent creates the P4 objects for each pipeline based on the P4 blob and maps each pipeline to a CPU core.
- Each CPU core executes one or multiple pipelines by running the associated instructions for each input packet. Each pipeline is single threaded.
- The external controller performs table updates and reads the statistics.



IPDK, with P4 as the pillar for everything

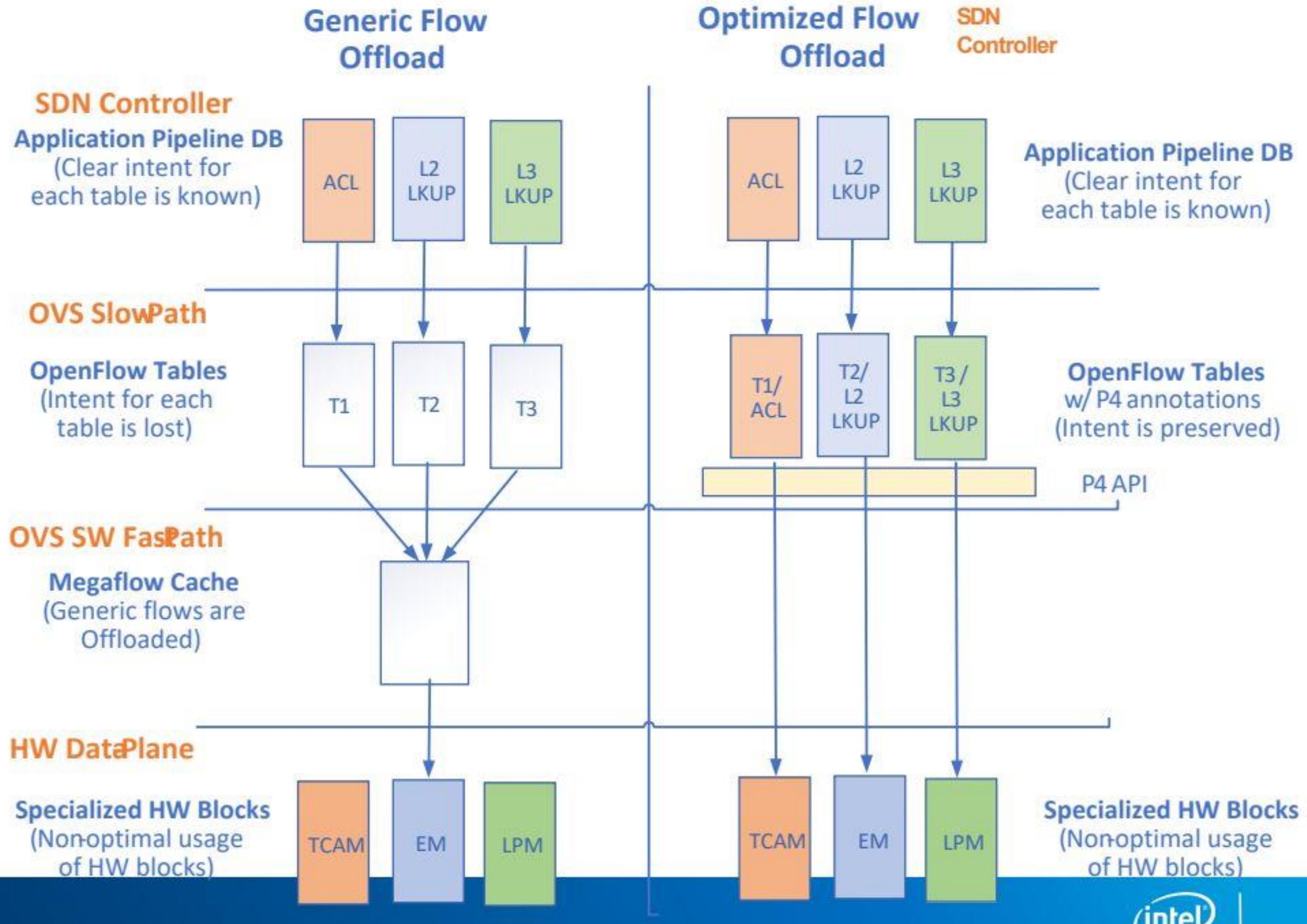
1. **Infrastructure-as-a-Service**
   Virtual Networking, Storage & Crypto
   Across VMs, containers & bare metal
2. **Platform-as-a-Service**
   Container Networking (Kubernetes)
   Sidecars (Envoy, MongoDB)
3. **Inline Acceleration**
   Firewall, IDS, Network Telemetry
   5G/Wireless Infrastructure, AI/ML



IPDK.io : Infrastructure Programmer Development Kit
Collaborate with the community on Github & Slack

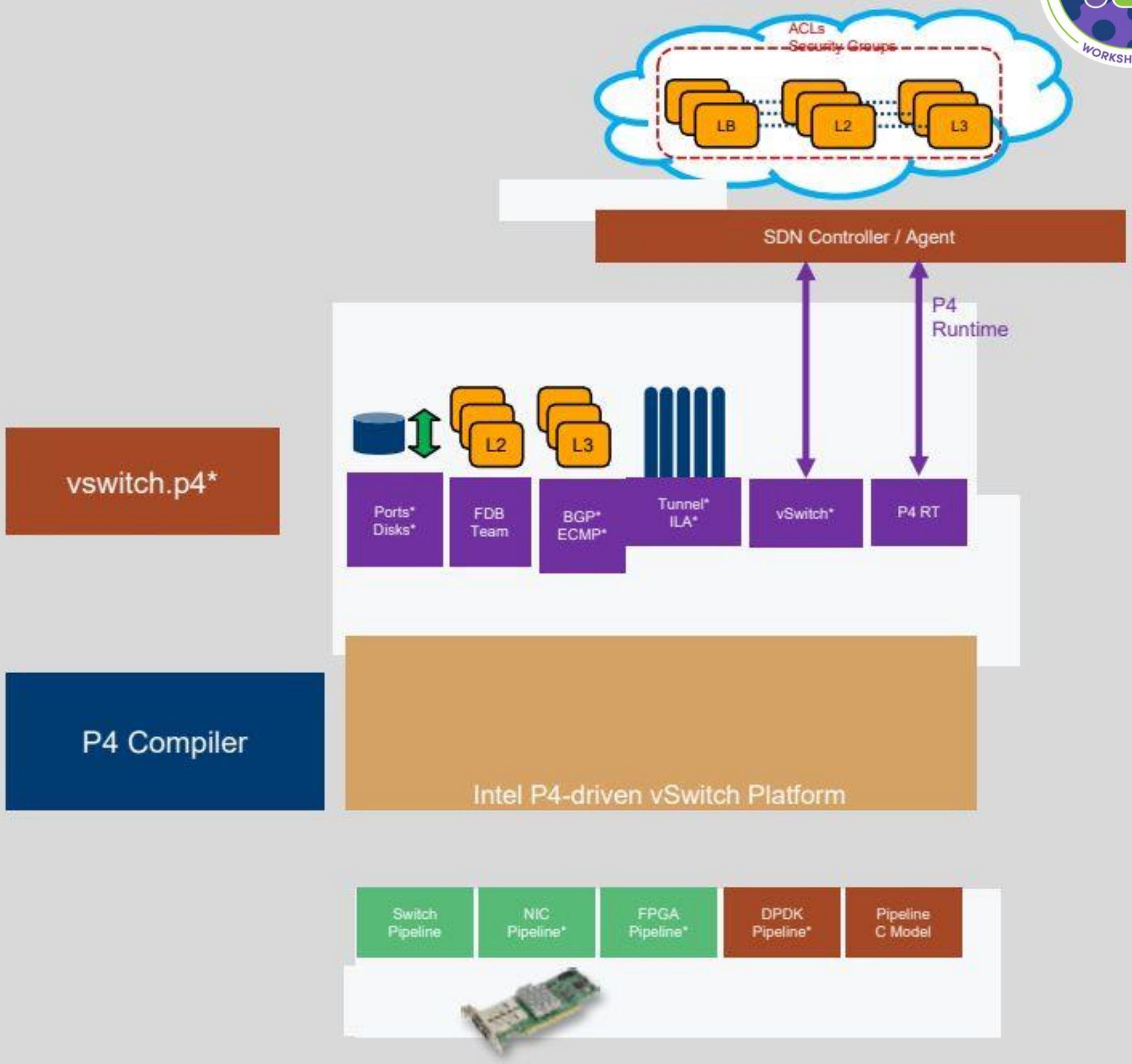
OVS HW offload challenges

- OVS Megaflow cache - great SW optimization idea, can be slow, potential of DDOS attack
- The reactive behavior requires high slow path performance; maintaining high flow set up rate is a challenge
- The aggregation followed by disaggregation -> the intent is lost
- Better -> skip the caches and offload directly to the HW tables
- Solves the other associated problems



P4-driven vSwitch platform

- As HW vendor, we need to support many platforms
  - Switch, NIC, FPGA, SW
- As HW vendor, we need to support many vSwitches
  - Custom vSwitches not upstreamed
  - Windows GFT, Vmware NSX-T, VPP and many others
- P4-driven vSwitch platform
  - Is common, extensible, and programmable
  - Any specific vSwitch can be supported by simply adding a thin shim layer



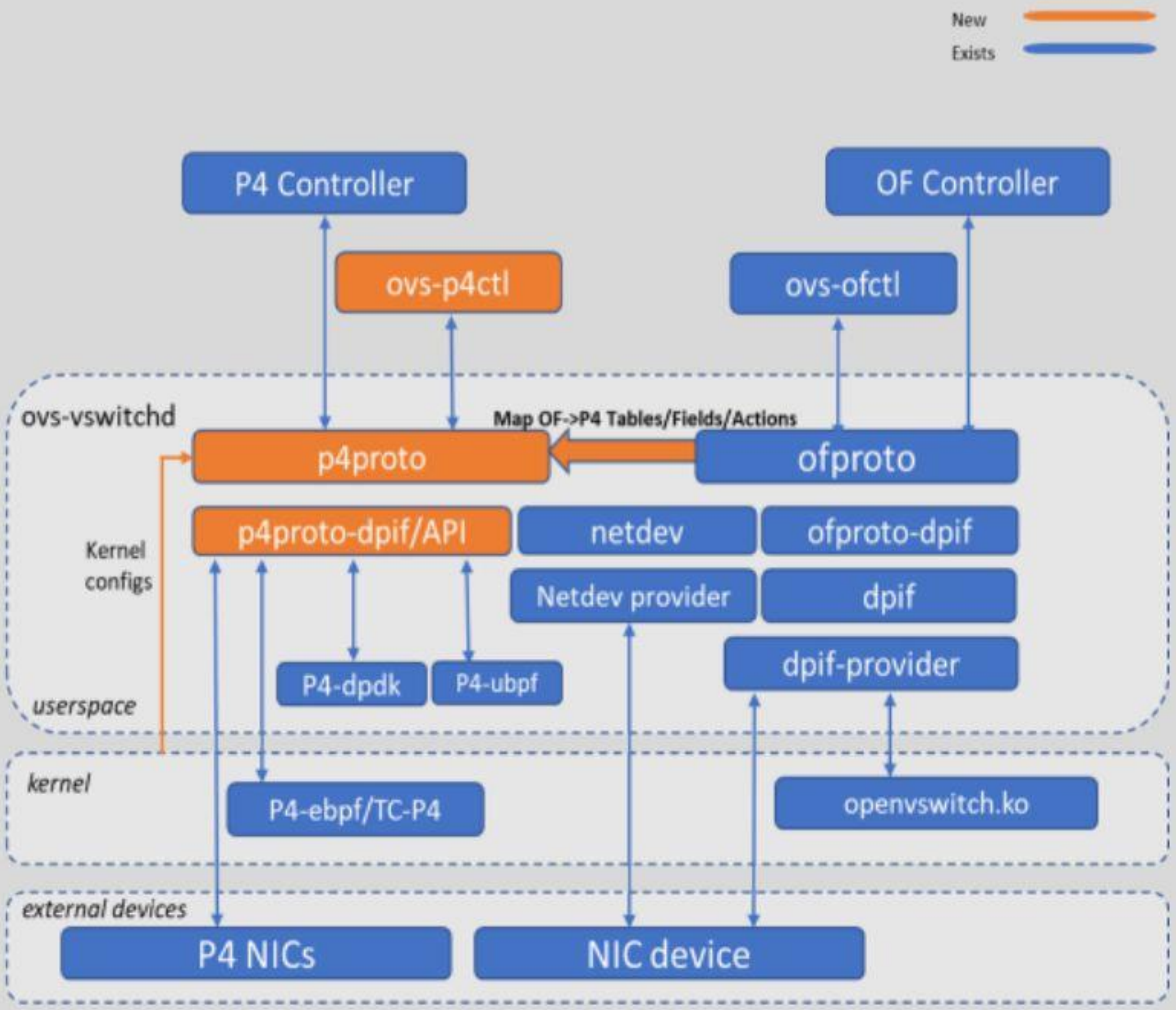
P4-driven vSwitch platform

Control planes:

- OVS – Maps OVS configuration to P4 Tables (E.g. Vxlan)
- P4Runtime + Openconfig – Configures P4 tables explicitly (E.g. Container load-balancing)
- Kernel – Maps Kernel configurations (via SAI) to P4 Tables (E.g. ECMP w/ FRR)
- All three control planes can used to program the same P4 target.
- Multiple P4Runtime clients can connect and program different P4 pipelines

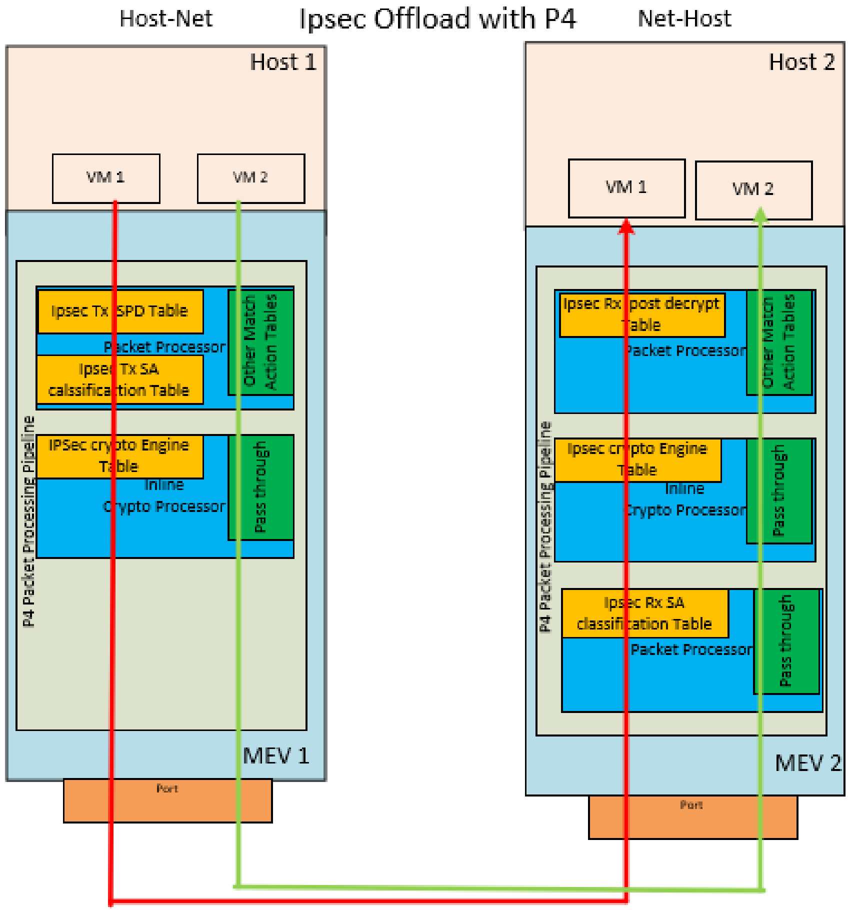
Data Planes:

- Physical NICs (Tofino, Intel P4 NICs etc)
- P4-DPDK (userspace)
- P4-ebpf (kernel)



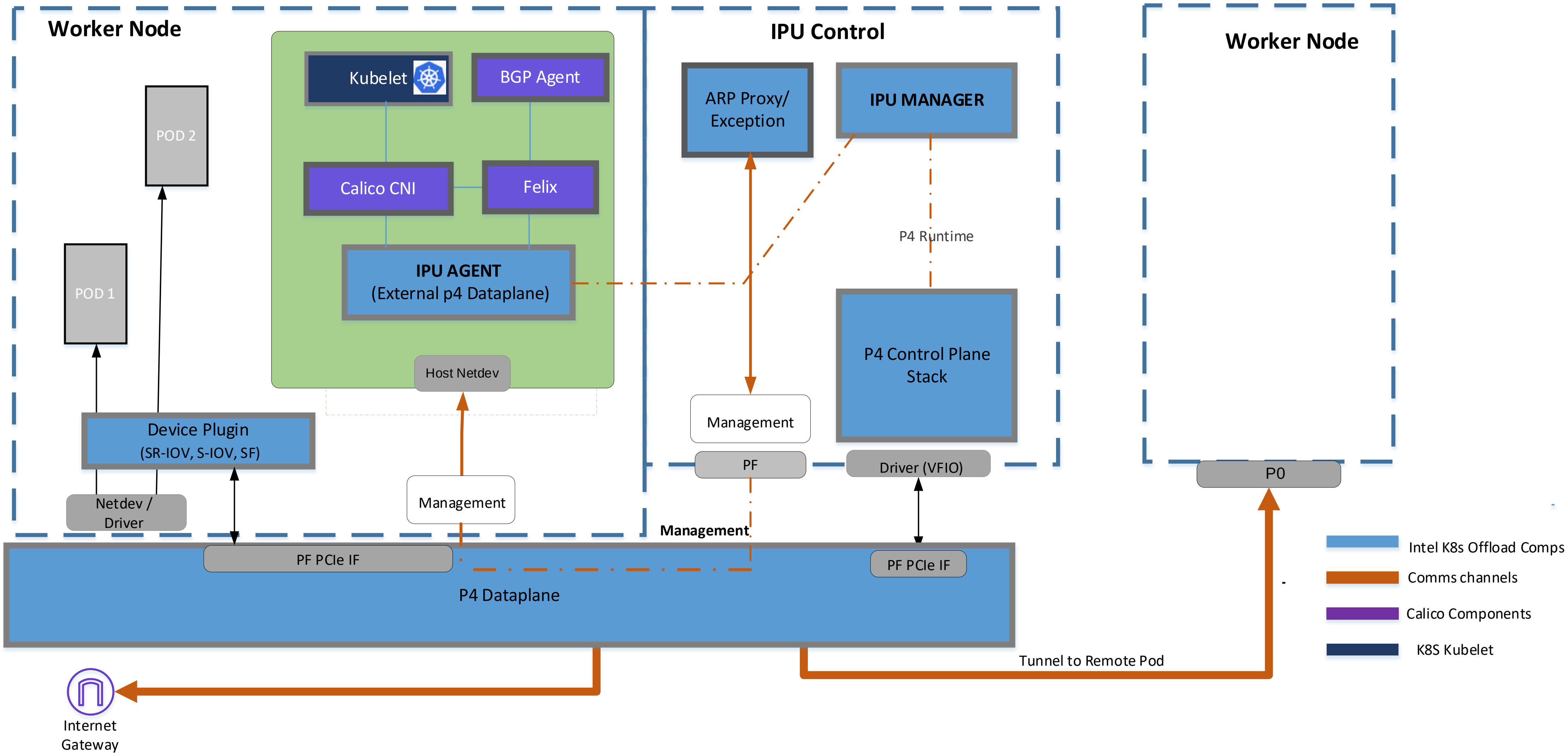
P4-programmable IPSec

- Both Tunnel and Transport mode IPSec traffic, along with non-IPSec traffic, can be offloaded simultaneously on the same NIC.
- All traffic passes through the inline crypto processor, and the P4 match-action tables determine which traffic to encrypt or decrypt, while passing the rest as clear traffic.
- P4 was very useful in supporting features such as custom VXLAN tags


Host-Net
Ipsec Offload with P4
Net-Host
Host 1
VM 1
VM 2
Ipsec Tx SPD Table
Packet Processor
Ipsec Tx SA calssification Table
Other Match Action Tables
IPSec crypto Engine Table
Inline Crypto Processor
Pass through
P4 Packet Processing Pipeline
MEV 1
Port
Host 2
VM 1
VM 2
Ipsec Rx post decrypt Table
Packet Processor
Other Match Action Tables
Ipsec crypto Engine Table
Inline Crypto Processor
Pass through
Ipsec Rx SA classification Table
Packet Processor
Pass through
P4 Packet Processing Pipeline
MEV 2
Port

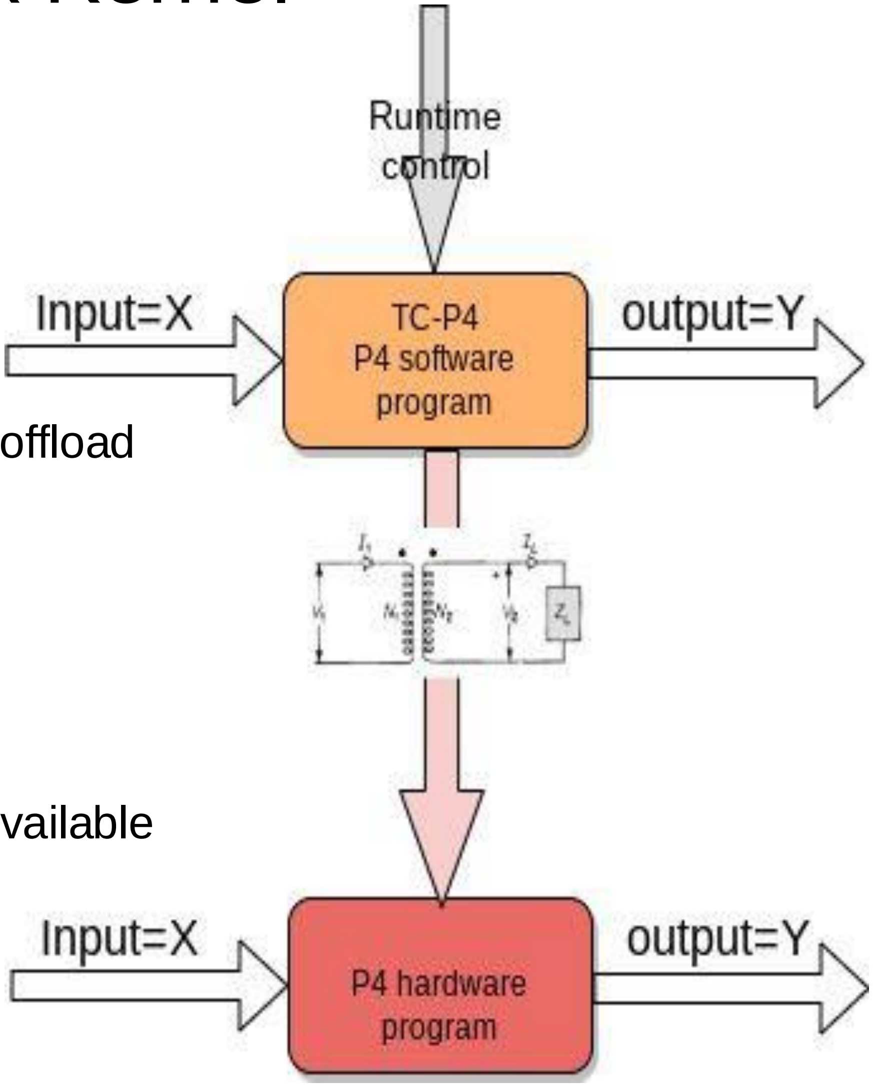
Green Line: Tx to Rx Clear Data Traffic
Red Line: Tx to Rx IPSec offloaded Data Traffic

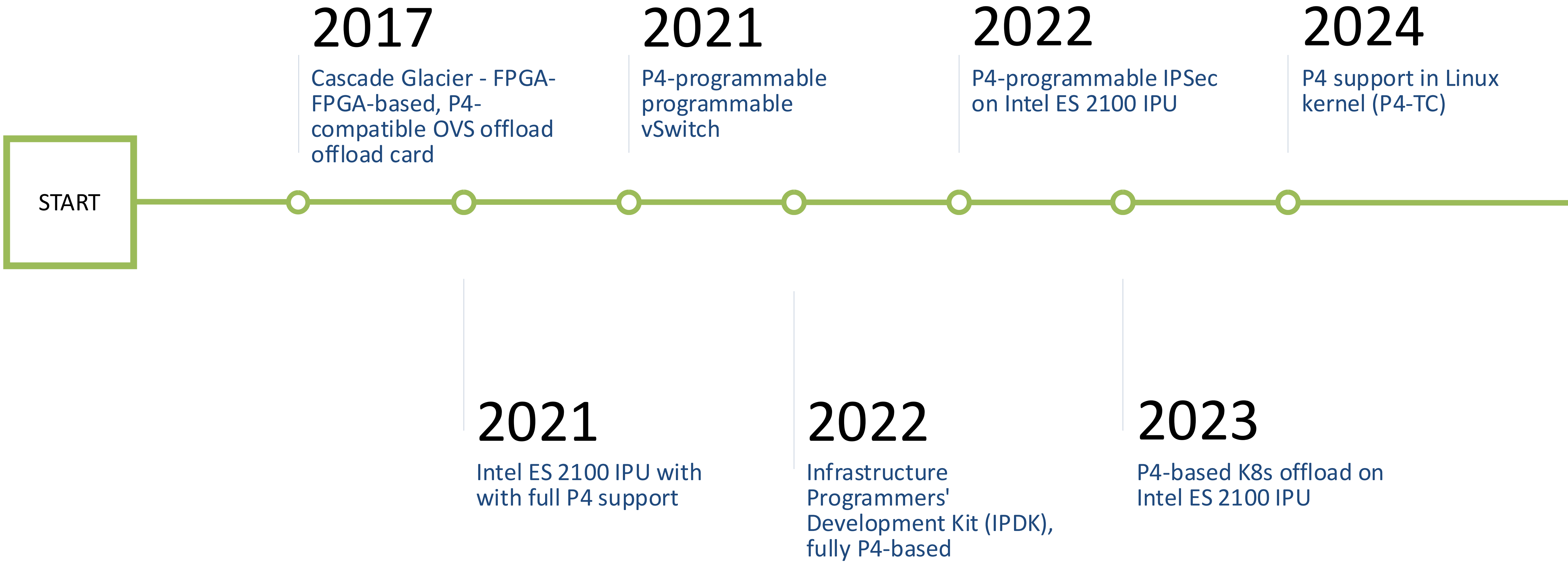
P4-enabled K8s offload



P4-TC: P4 support in Linux Kernel

- Datapath definition using P4
  - Generate the datapath for both s/w and vendor h/w
    - Functional equivalence between sw and hw
- P4 Linux kernel-native implementation
  - Kernel TC-based software datapath and Kernel-based HW datapath offload
    - Understood Infra tooling which already has deployments
  - Seamless software and hardware symbiosis
  - Functional equivalence whether offloading or s/w datapaths
    - Bare Metal, VMs, or Containers
  - Ideal for datapath specification
    - test in s/w container, VM, etc) then offload when hardware is available





START

2017
Cascade Glacier - FPGA-FPGA-based, P4-compatible OVS offload offload card

2021
Intel ES 2100 IPU with with full P4 support

2021
P4-programmable programmable vSwitch

2022
Infrastructure Programmers' Development Kit (IPDK), fully P4-based

2022
P4-programmable IPSec on Intel ES 2100 IPU

2023
P4-based K8s offload on Intel ES 2100 IPU

2024
P4 support in Linux kernel (P4-TC)

Our experience with P4

- The learning curve of P4 is possibly a myth!
- More than the expressibility of P4, it’s quite often the underlying limitations of the HW that becomes the limiting factor
- P4 leaves a lot of details to be figured out and changed at the compiler level, which is not a bad thing
- The language can benefit from having additional objects recognized, such as queues
- Perfect portability, defined as “taking P4 written for device A, and running it unmodified on device B” may be an elusive target
- New use cases, such as being able to support machine learning, is worth exploring. Some of these newer use cases will require support for not just abstraction of packets, but also abstraction of flows and beyond
- New P4 possibilities are endless. We should keep maintaining the momentum on P4!

Thank You