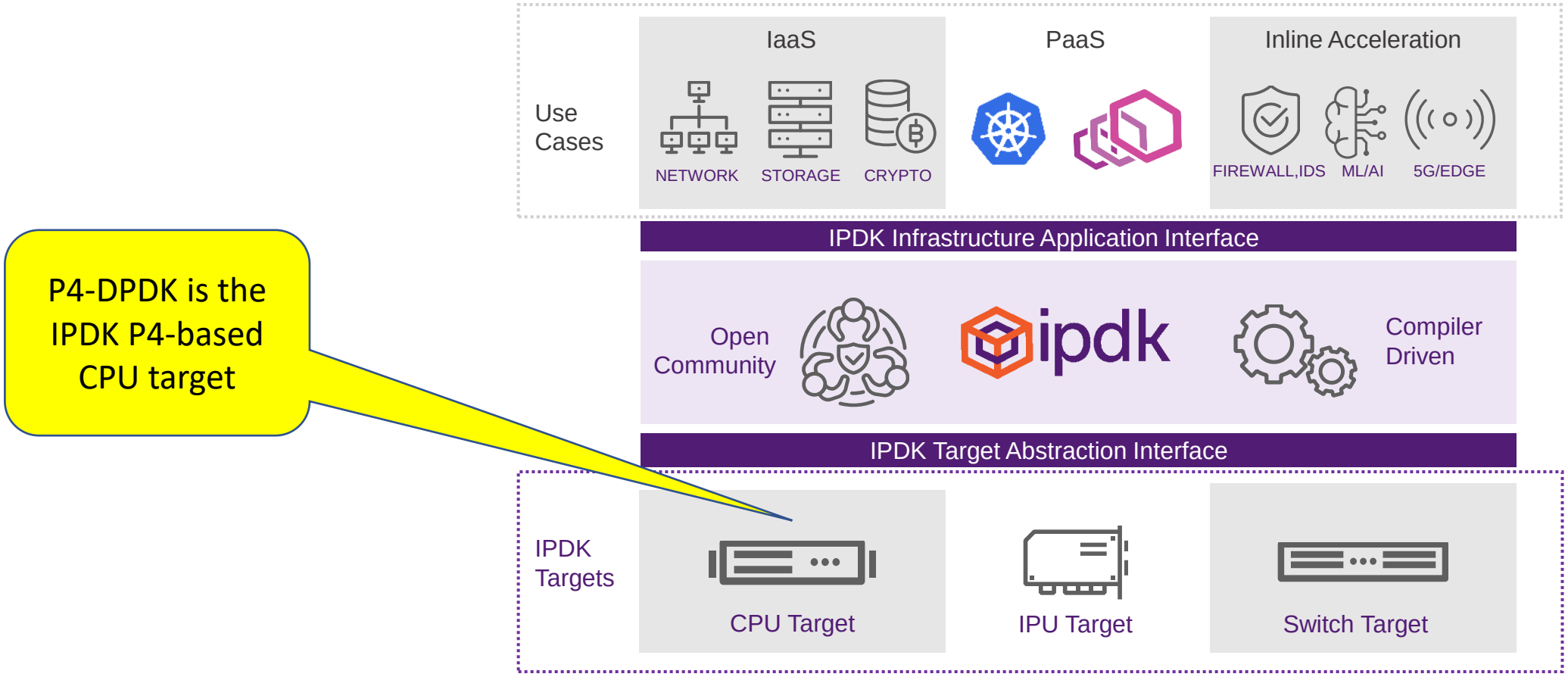
Develop your CPU network stack in P4

Cristian Dumitrescu, SW Architect, Intel

Agenda

1. P4-DPDK: What is it
2. P4-DPDK: What is it not
3. P4-DPDK Feature Update
4. P4-DPDK New Feature: Compiled Pipeline
5. Example Use-cases
6. Conclusions

IPDK.io


Use Cases
IaaS
NETWORK
STORAGE
CRYPTO
PaaS
Inline Acceleration
FIREWALL,IDS
ML/AI
5G/EDGE
IPDK Infrastructure Application Interface
Open Community
ipdk
Compiler Driven
IPDK Target Abstraction Interface
IPDK Targets
CPU Target
IPU Target
Switch Target
P4-DPDK is the IPDK P4-based CPU target

P4-DPDK: What is it

- Open-source framework to run P4 programs on multi-core CPUs.
- Goal: Develop better and faster SW switches and network stacks by combining the P4 language flexibility with the DPDK performance.
- The IPDK project uses P4-DPDK as the CPU target.
- Open-source:
  - P4 compiler back-end and TDI implementation on p4.org
  - P4 data plane engine on dpdk.org.

P4-DPDK is getting better, faster and more pervasive every year!

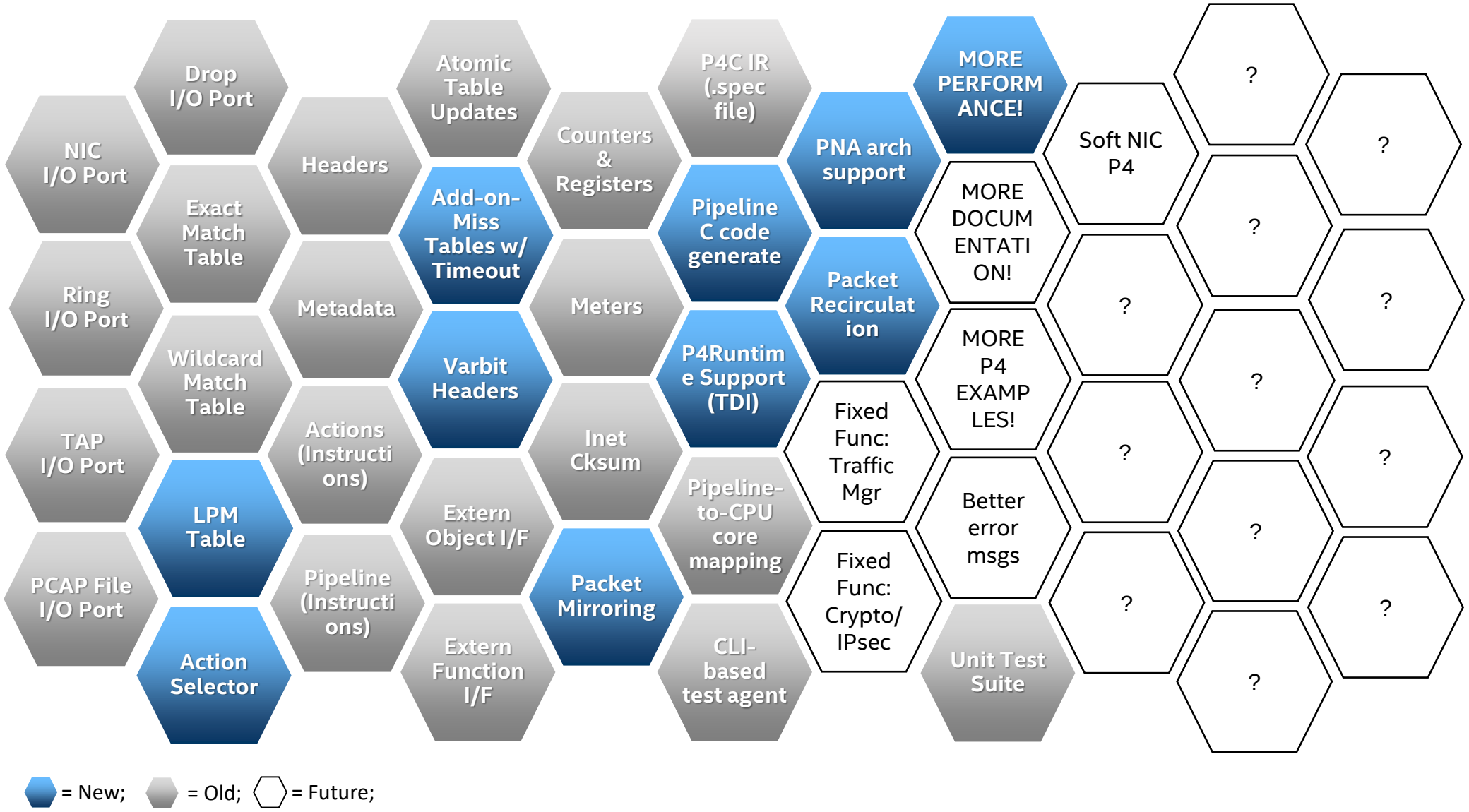
P4-DPDK: What is it (2)

Component	Open-source repository
DPDK P4 data plane engine	http://git.dpdk.org/dpdk/tree/lib/pipeline
P4C compiler back-end	https://github.com/p4lang/p4c/tree/main/backends/dpdk
Table Driven Interface (TDI)	https://github.com/p4lang/tdi
TDI implem. for P4-DPDK target	https://github.com/p4lang/p4-dpdk-target
P4Runtime server	https://github.com/stratum/stratum/tree/main/stratum/hal/lib/barefoot
IPDK	https://github.com/ipdk-io

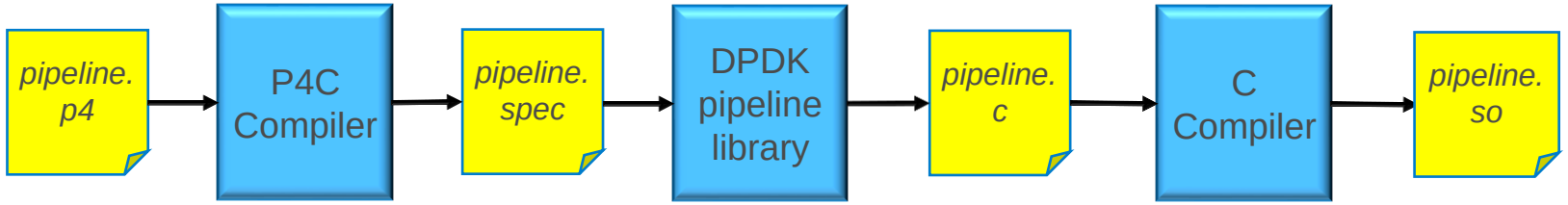
P4-DPDK: What it is not

- P4-DPDK is not a P4 language simulator, like BMv2. Performance is key for P4-DPDK!

P4-DPDK Feature Update (since P4 Workshop 2021)



New feature: Compiled pipeline



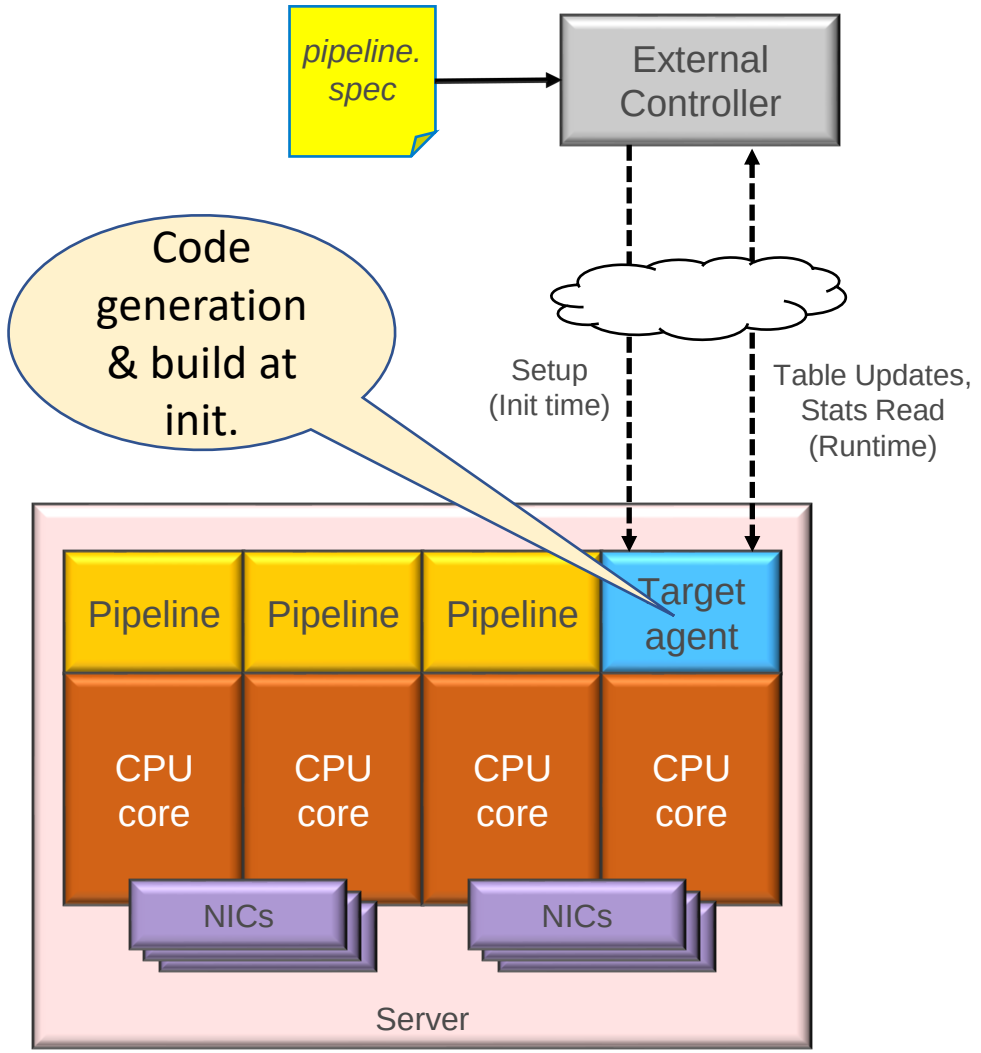
- Interpreted pipeline mode (runs the *pipeline.spec* file):
  - The plain text *.spec* file contains the P4 object definitions (headers, meta-data, actions, tables, etc) and subroutines (translated actions and control blocks).
  - The subroutines are made out of instructions from a predefined P4 “virtual machine” ISA. The instruction operands are the P4 objects.
  - Performance penalty: for every instruction, a function pointer is invoked (slow)!
- Compiled pipeline mode (runs the *pipeline.so* file):
  - The binary shared object *.so* file is built out of the *pipeline.c* file, which is generated from the *.spec* file.
  - The *.c* file contains a C function for every action and control block. The instructions are replaced by an inline call to their associated function.
  - Performance improvement (typ. 30-70%): achieved by the C compiler having visibility on the entire pipeline code, which it can now efficiently optimize!

Enable the compiled pipeline mode for even more performance!

New feature: Compiled pipeline (2)

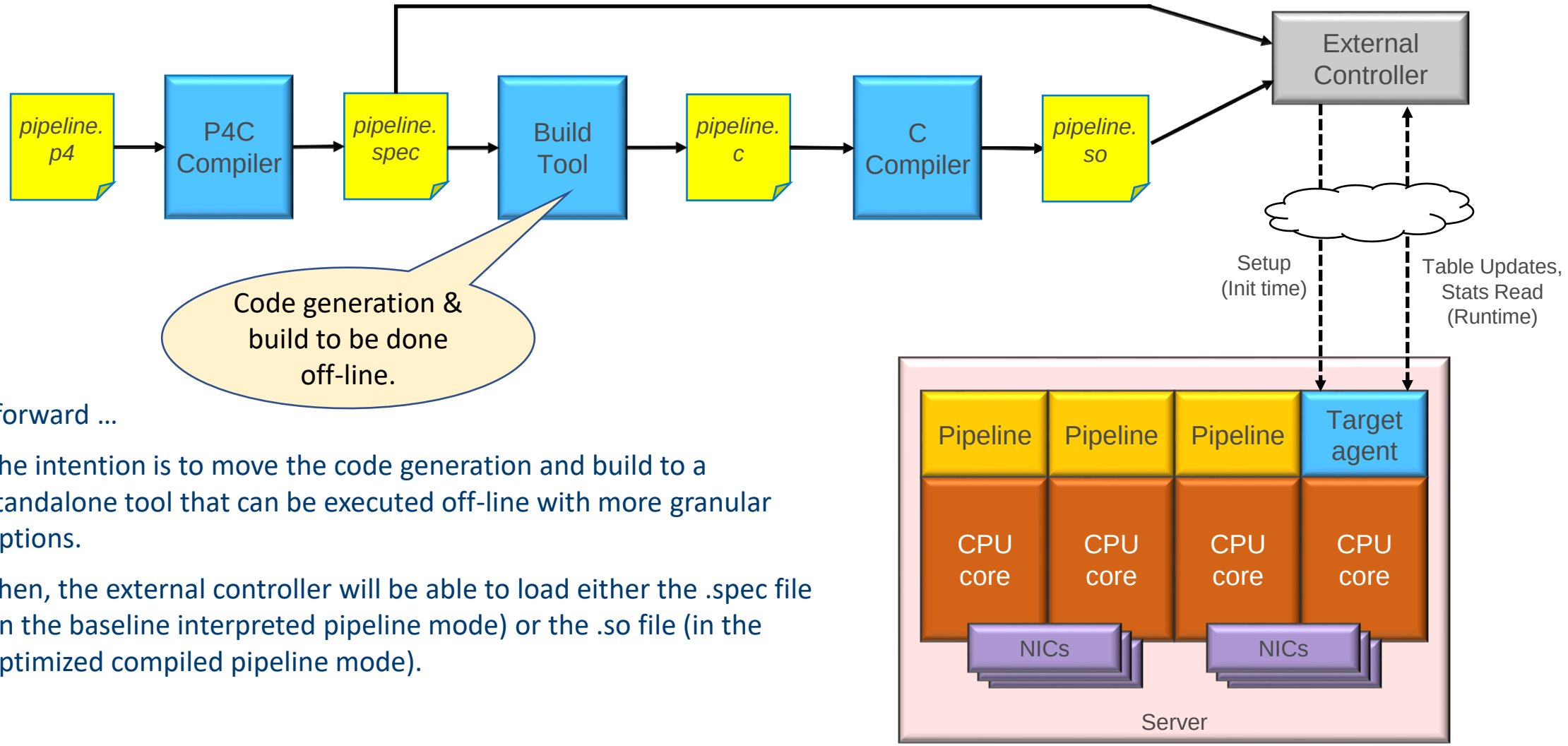
Currently, the compiled pipeline mode can be easily enabled at run-time simply by exporting the following environment variable:

export RTE_INSTALL_DIR=<*PATH_TO_DPDK_FOLDER*>



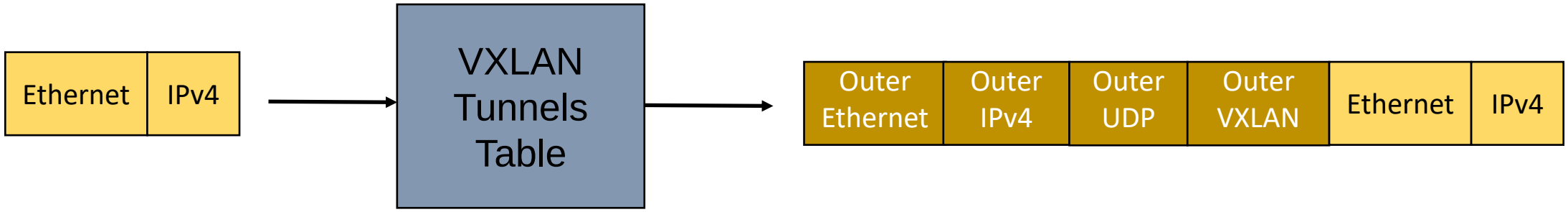
- Currently …
  - The external controller loads the *.spec* file to the target agent.
  - The code generation (*pipeline.c*) and build (*pipeline.so*) are executed silently under the hood by the target agent at initialization, if enabled; if failing for any reason (e.g. C compiler not installed), the execution reverts to the interpreted mode.

New feature: Compiled pipeline (3)



- Going forward …
  - The intention is to move the code generation and build to a standalone tool that can be executed off-line with more granular options.
  - Then, the external controller will be able to load either the .spec file (in the baseline interpreted pipeline mode) or the .so file (in the optimized compiled pipeline mode).

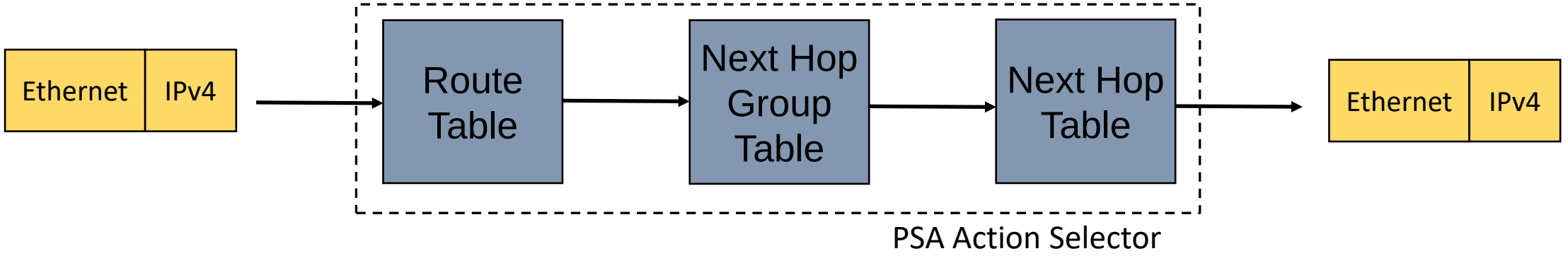
Use-case: VXLAN Encapsulation



Highlights:

- Exact match table lookup: 64K tunnels
- Complex action: push 50 bytes of headers from the table entry to the packet, update IPv4 and UDP length, update IPv4 checksum.

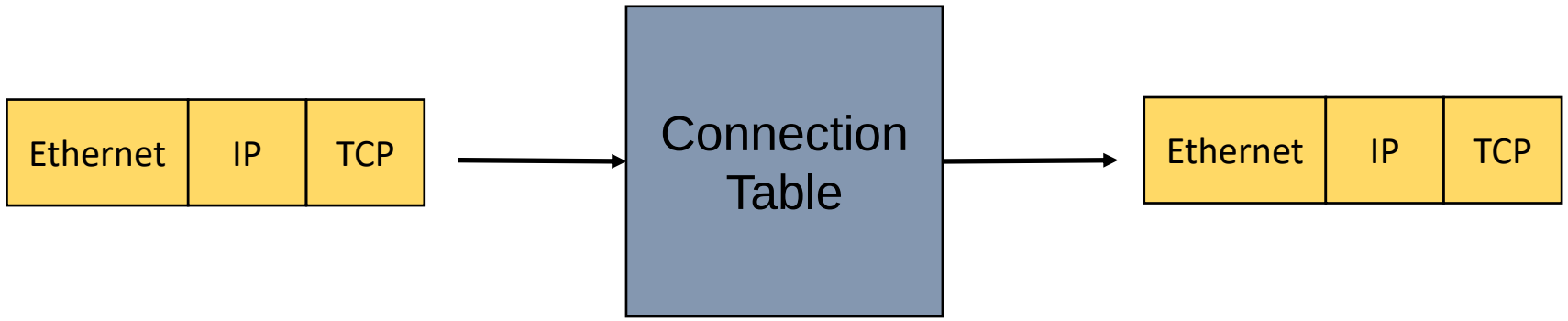
Use-case: Complex FIB with VRF and ECMP/WCMP



Highlights:

- Route Table: Virtual Routing and Forwarding (VRF) support. Key: vrf_id (exact match), ipv4_dst_addr (LPM match);
- Next Hop Group Table: Equal/Weighted Cost Multi-Path (ECMP/WCMP) support;
- Next Hop Table: Exact match.

Use-case: Connection Tracking



PNA add-on-miss table

Highlights:

- Connection table: PNA add-on-miss table that allows the data plane to modify the table entries without any control plane intervention. Entries automatically expire on timeout, unless hit and their timer rearmed.
- Default action: learn (conditionally) the missed flow.
- Regular action: rearm the hit entry timer or do nothing (conditionally).

P4 program:
https://github.com/p4lang/pna/blob/main/examples/pna-example-tcp-connection-tracking.p4

Conclusions

1. A lot of work has been done lately in P4-DPDK to enable more features, performance and use-cases.
2. P4-DPDK can be used to quickly develop complex CPU network stacks. One example is the P4-OVS project under IPDK.
3. P4-DPDK is becoming better, faster and more pervasive every year!

Thank You