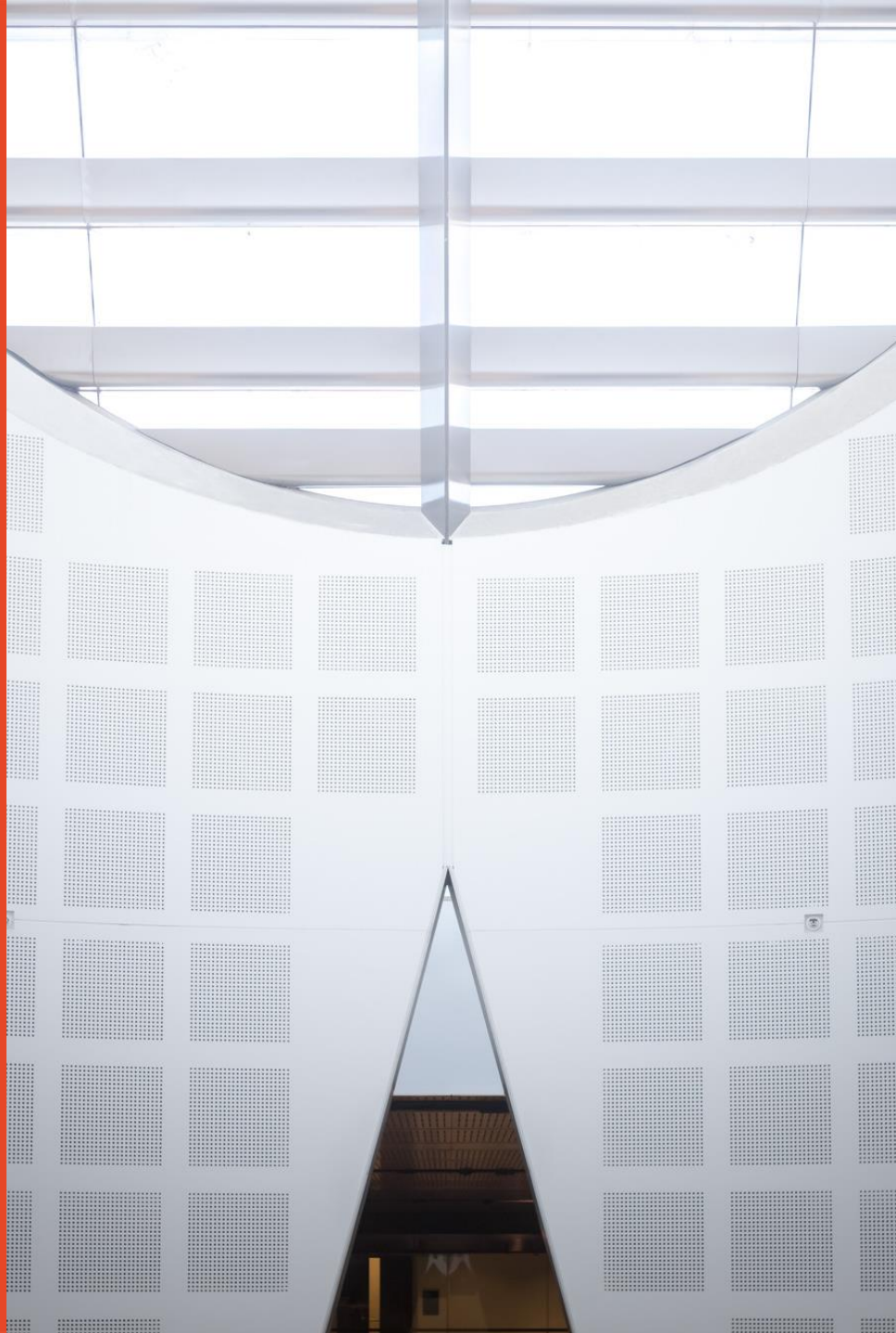
# Topic 8: P4 - Protocol Independent Data Plane

**Presented by**
Dong YUAN

School of Electrical and Computer Engineering

dong.yuan@sydney.edu.au

THE UNIVERSITY OF SYDNEY



# Desirable Features in SDN switches

- Configurable packet parser
  - Not tied to a specific header format
- Flexible match+action tables
  - Multiple tables (in series and/or parallel)
  - Able to match on all defined fields
- General packet-processing primitives
  - Copy, add, remove and modify
  - For both header fields and meta-data

# Protocol-Independent Data Plane

- Bosshart, Pat, et al. "P4: Programming protocol-independent packet processors." ACM SIGCOMM Computer Communication Review 44.3 (2014): 87-95.

# P4: Programming protocol-independent packet processors

- P4 is a high-level language for programming protocol-independent packet processors.

- P4 works in conjunction with SDN control protocols like OpenFlow.
  - OpenFlow explicitly specifies protocol headers on which it operates. This set has grown from 12 to 41 fields in a few years, increasing the complexity of the specification.

- P4 propose how OpenFlow should evolve in the future.

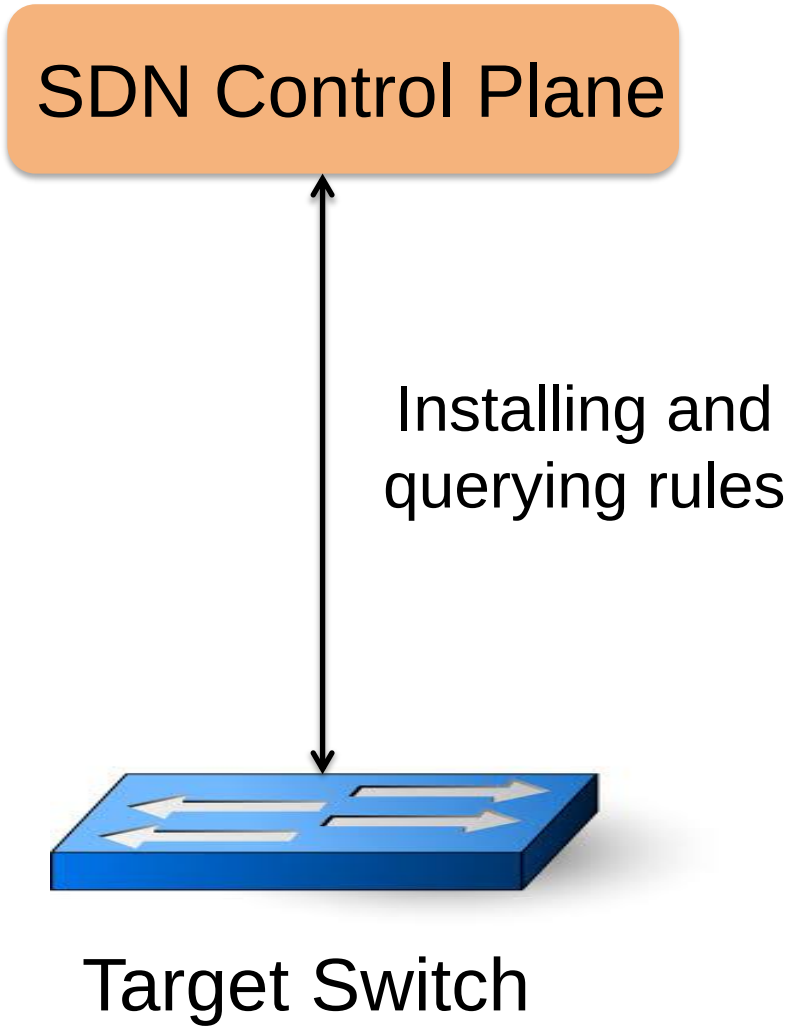
# Over Five Years…

Proliferation of header fields

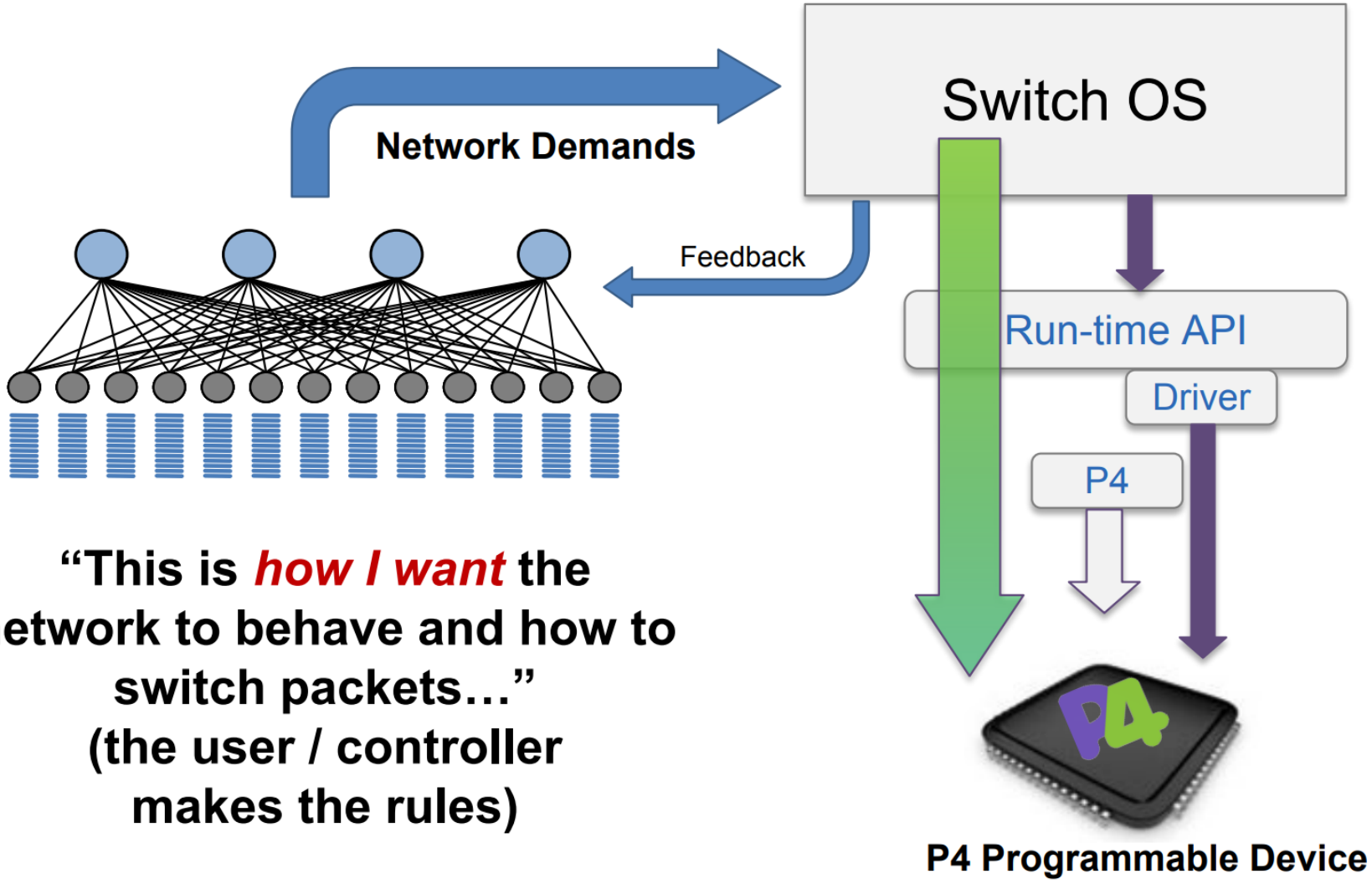
| Version | Date | # Headers |
|---|---|---|
| OF 1.0 | Dec 2009 | 12 |
| OF 1.1 | Feb 2011 | 15 |
| OF 1.2 | Dec 2011 | 36 |
| OF 1.3 | Jun 2012 | 40 |
| OF 1.4 | Oct 2013 | 41 |

- Multiple stages of heterogeneous tables, Still not enough (e.g., VXLAN, NVGRE, STT, …)
- Control and data not sufficiently decoupled
- No easy way to modify packet format

# "Classic" OpenFlow (1.x)



# Top-down design of a switch

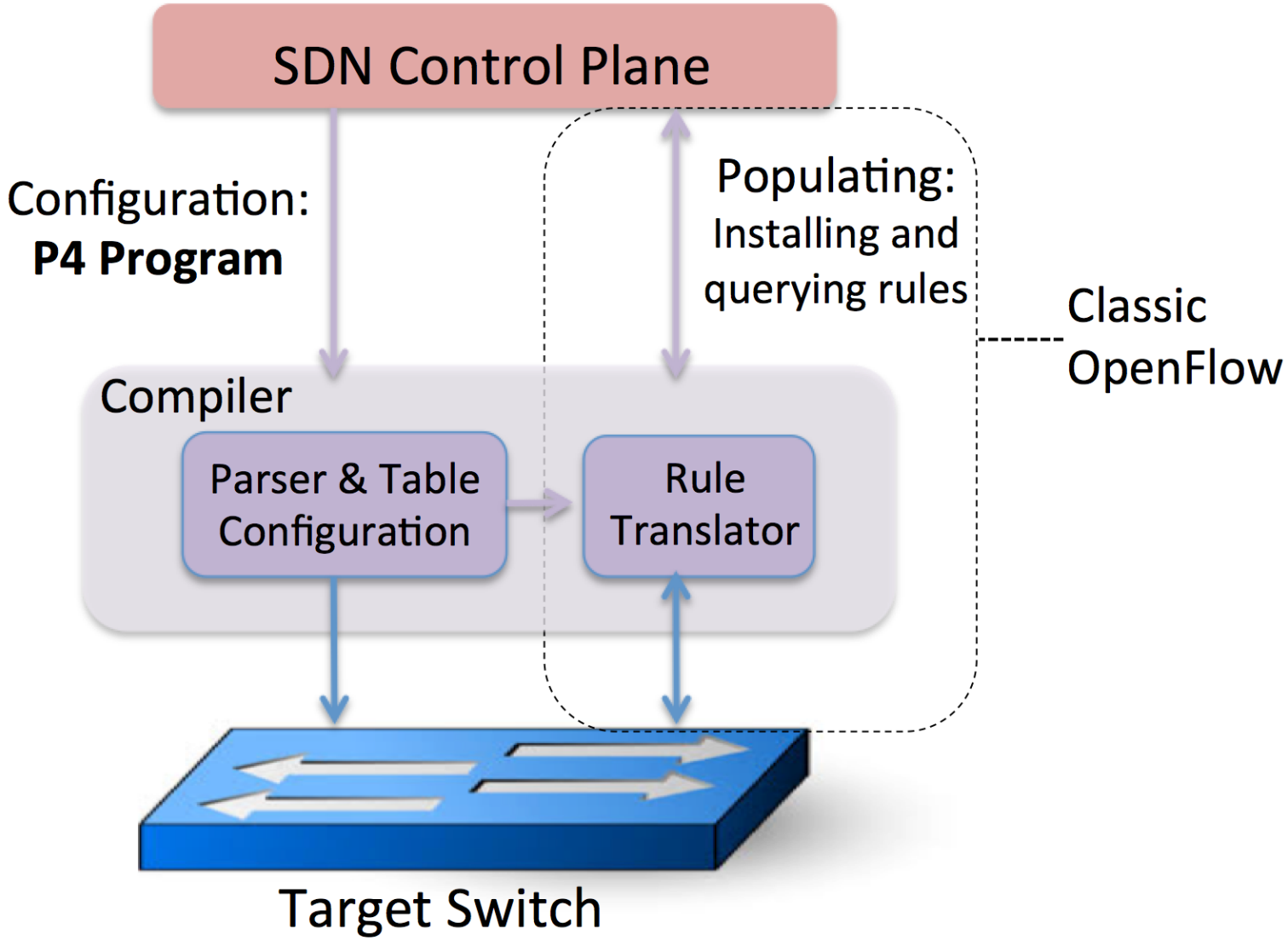



**"This is *how I want* the network to behave and how to switch packets…" (the user / controller makes the rules)**

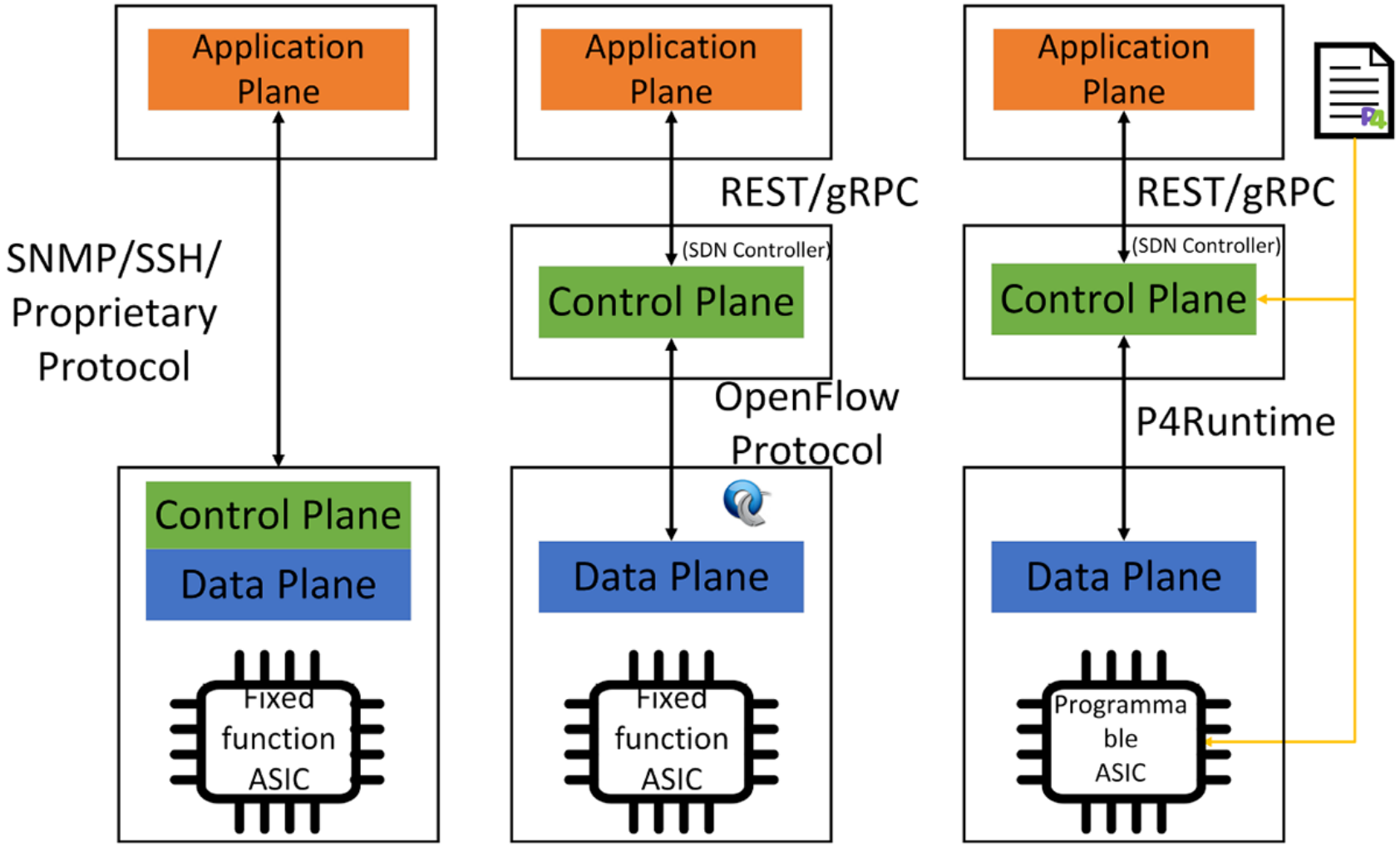


https://opennetworking.org/wp-content/uploads/2020/12/P4_D2_East_2018_01_basics.pdf

# "OpenFlow 2.0" with P4 language



# Traditional Networking v.s. Openflow v.s. P4



# Three goals

- Reconfigurability in the field: Programmers should be able to change the way switches process packets once they are deployed.

- Protocol independence: Switches should not be tied to any specific network protocols.
  - Configure a packet parser
  - Define a set of typed match+action tables

- Target independence: Programmers should be able to describe packet processing functionality independently of the specifics of the underlying hardware.
  - Program without knowledge of switch details
  - Rely on compiler to configure the target switch

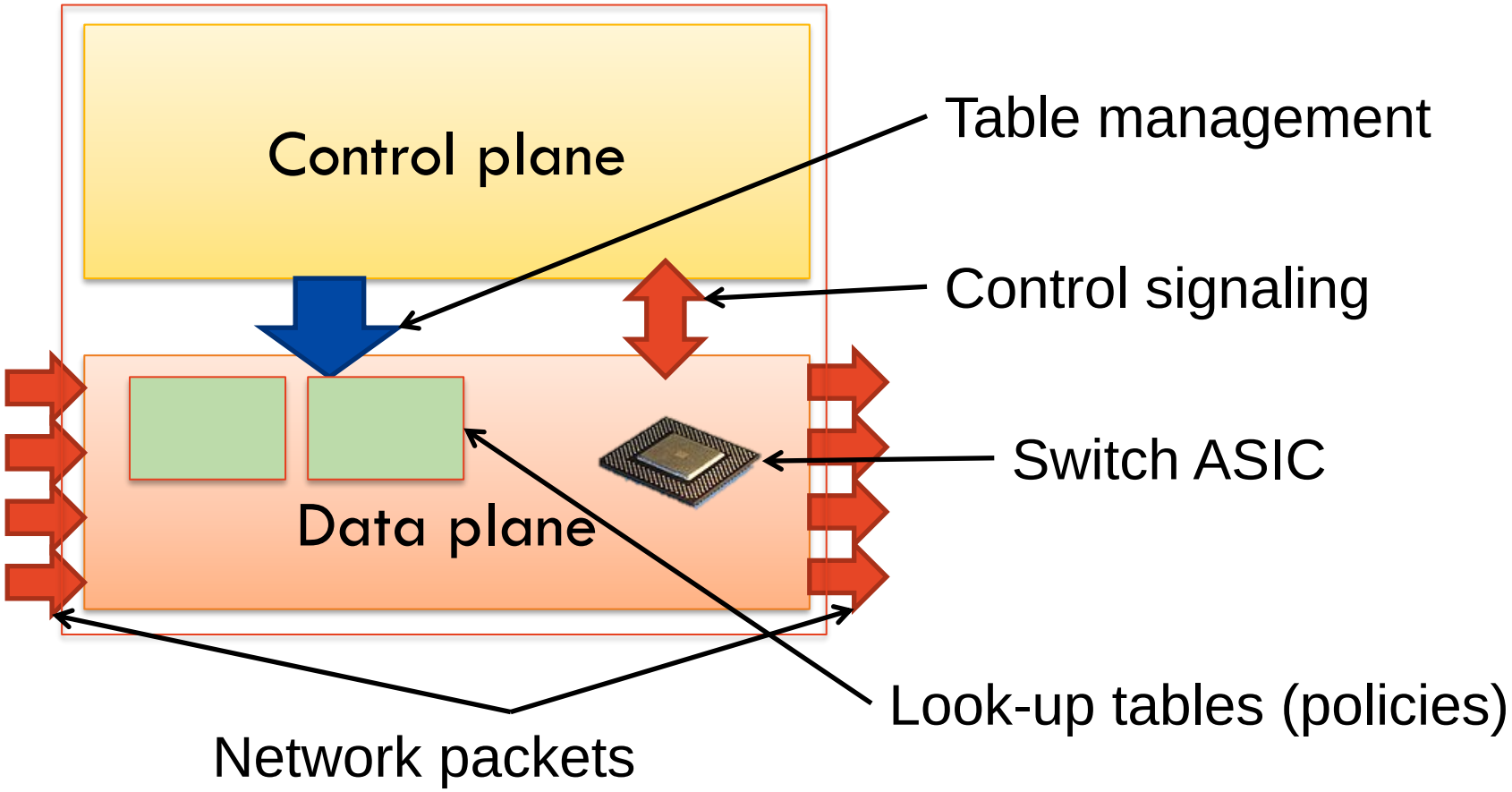
# P4 Community


- http://github.com/p4lang
- http://p4.org

- Mailing lists
- Workshops / P4 developer days

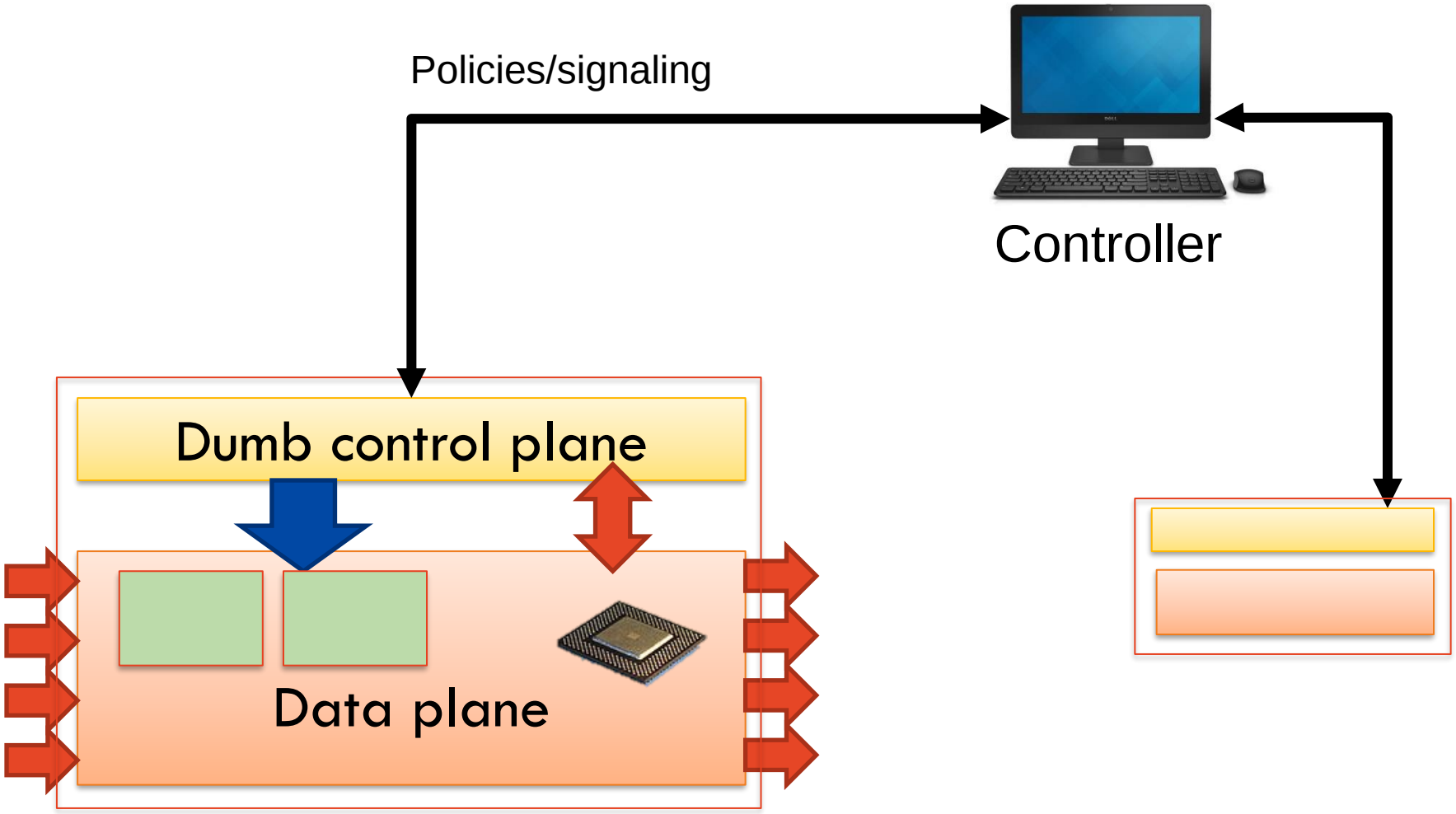
- $P4_{16}$ - Most recent revision
  - Similar to C; strongly typed
  - Spec: https://p4.org/p4-spec/docs/P4-16-v1.2.0.pdf
  - Reference compiler implementation (Apache 2 license): http://github.com/p4lang/p4c

# Traditional switch architecture



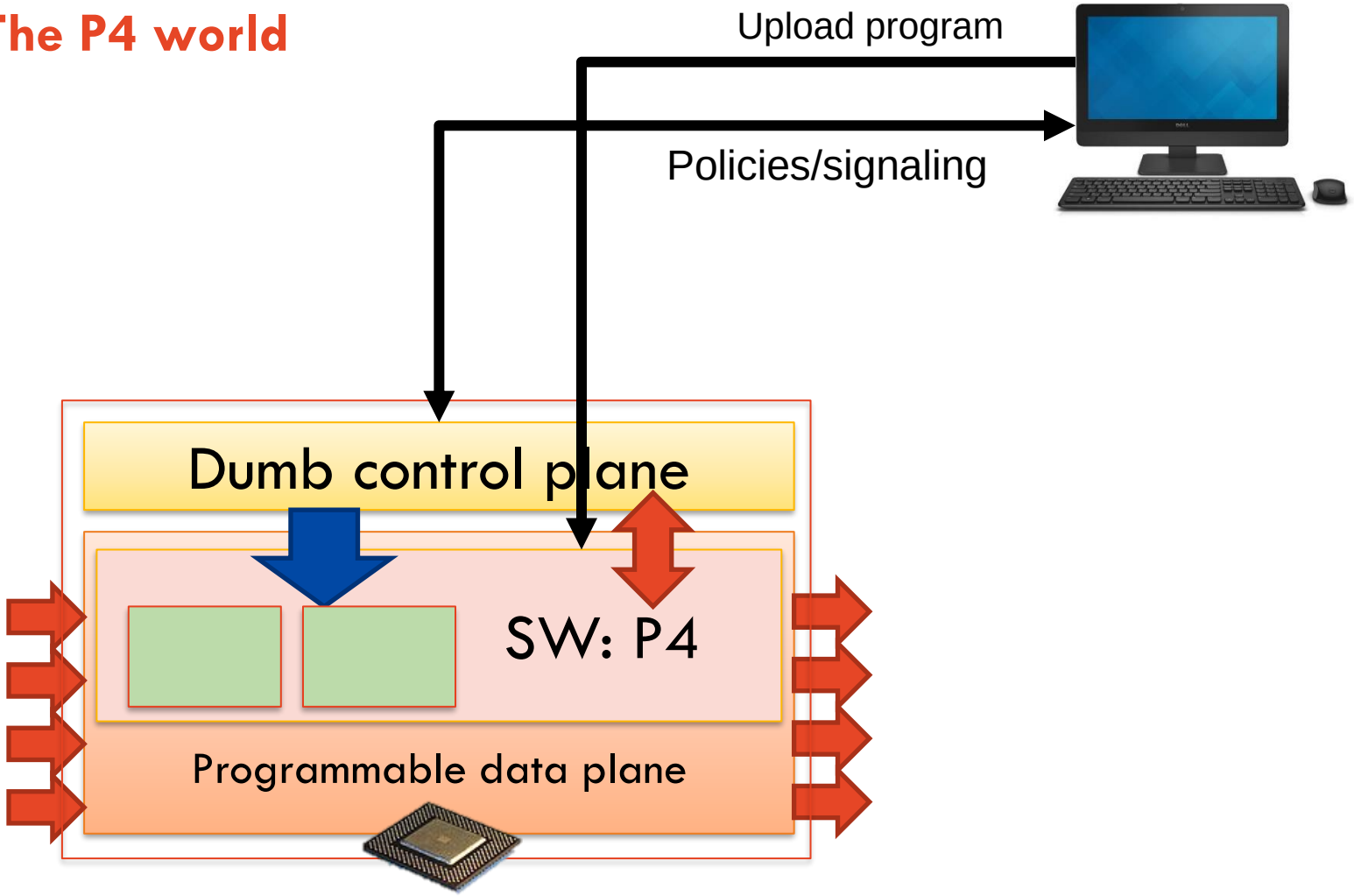

Source: Mihai Budiu, VMware Research

# Software-Defined Networking



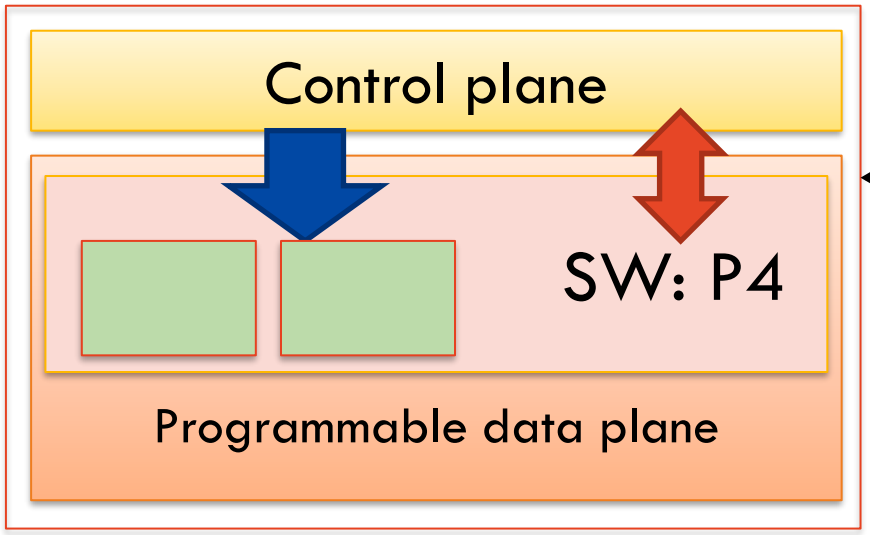

Source: Mihai Budiu, VMware Research

# The P4 world




Source: Mihai Budiu, VMware Research

# Not just for switches!



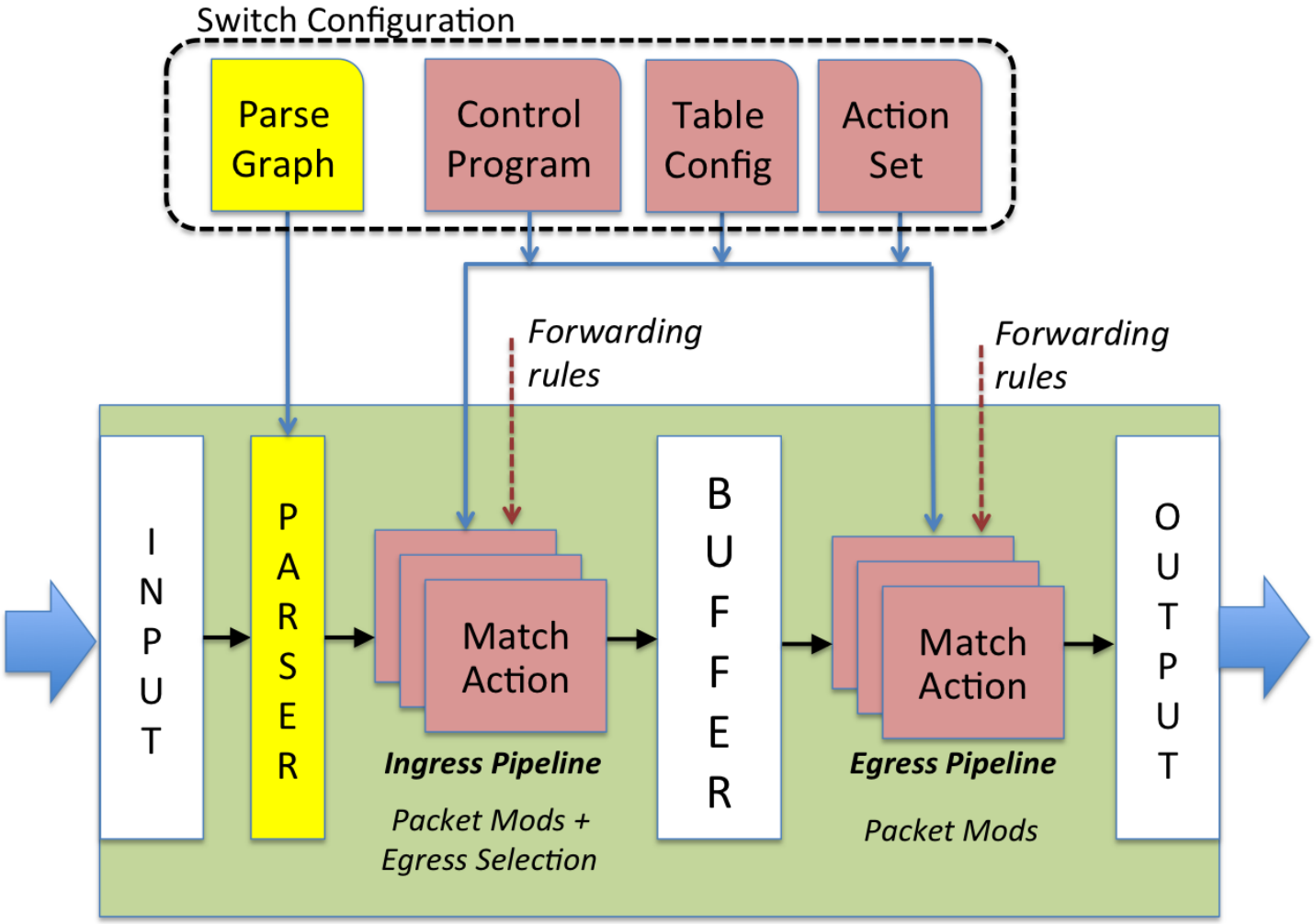

- Programmable switches
- FPGA switches
- Programmable network cards
- Software switches
- Hypervisor switches
- You name it…

Source: Mihai Budiu, VMware Research

# P4 vs. Openflow

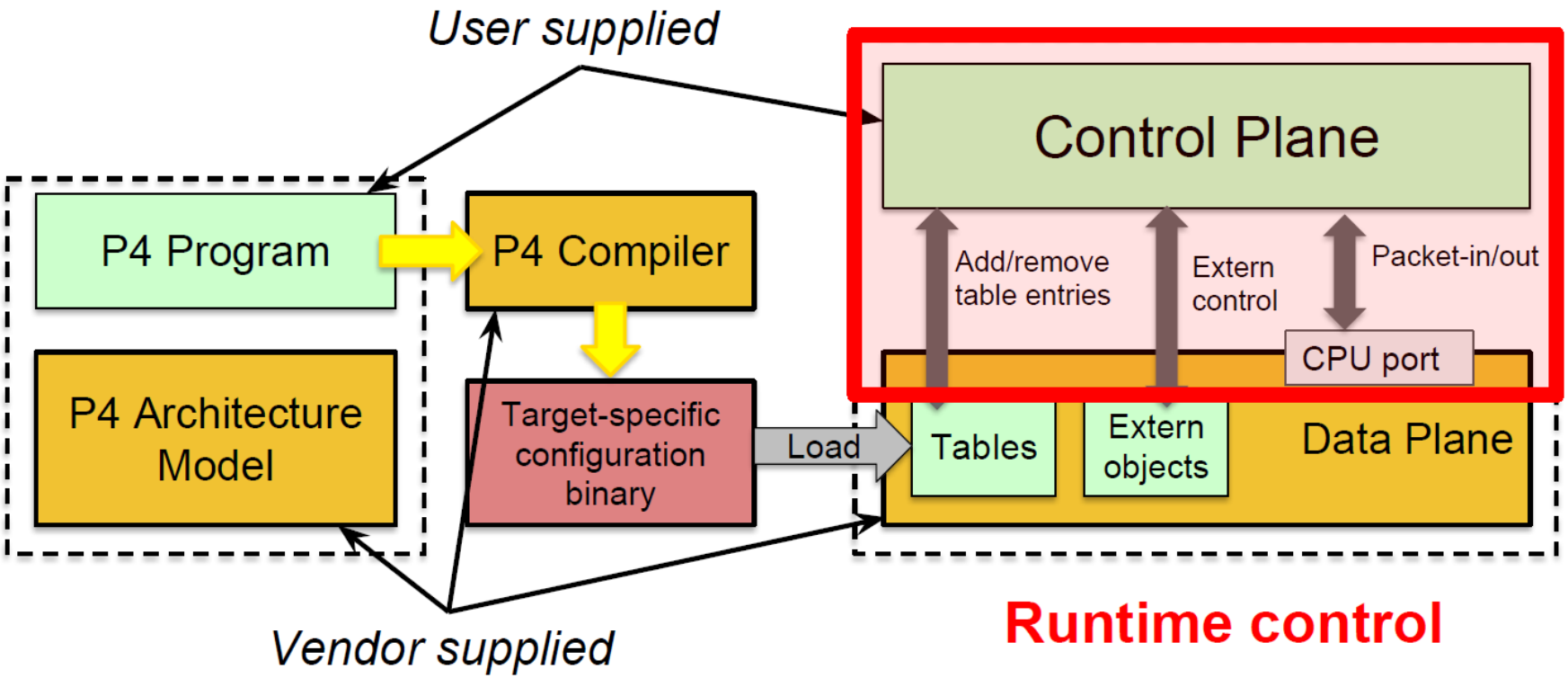
- OpenFlow assumes a fixed parser, whereas P4 supports a programmable parser to allow new headers to be defined.
- OpenFlow assumes the match+action stages are in series, whereas in P4 they can be in parallel or in series.
- P4 assumes that actions are composed from protocol-independent primitives supported by the switch.

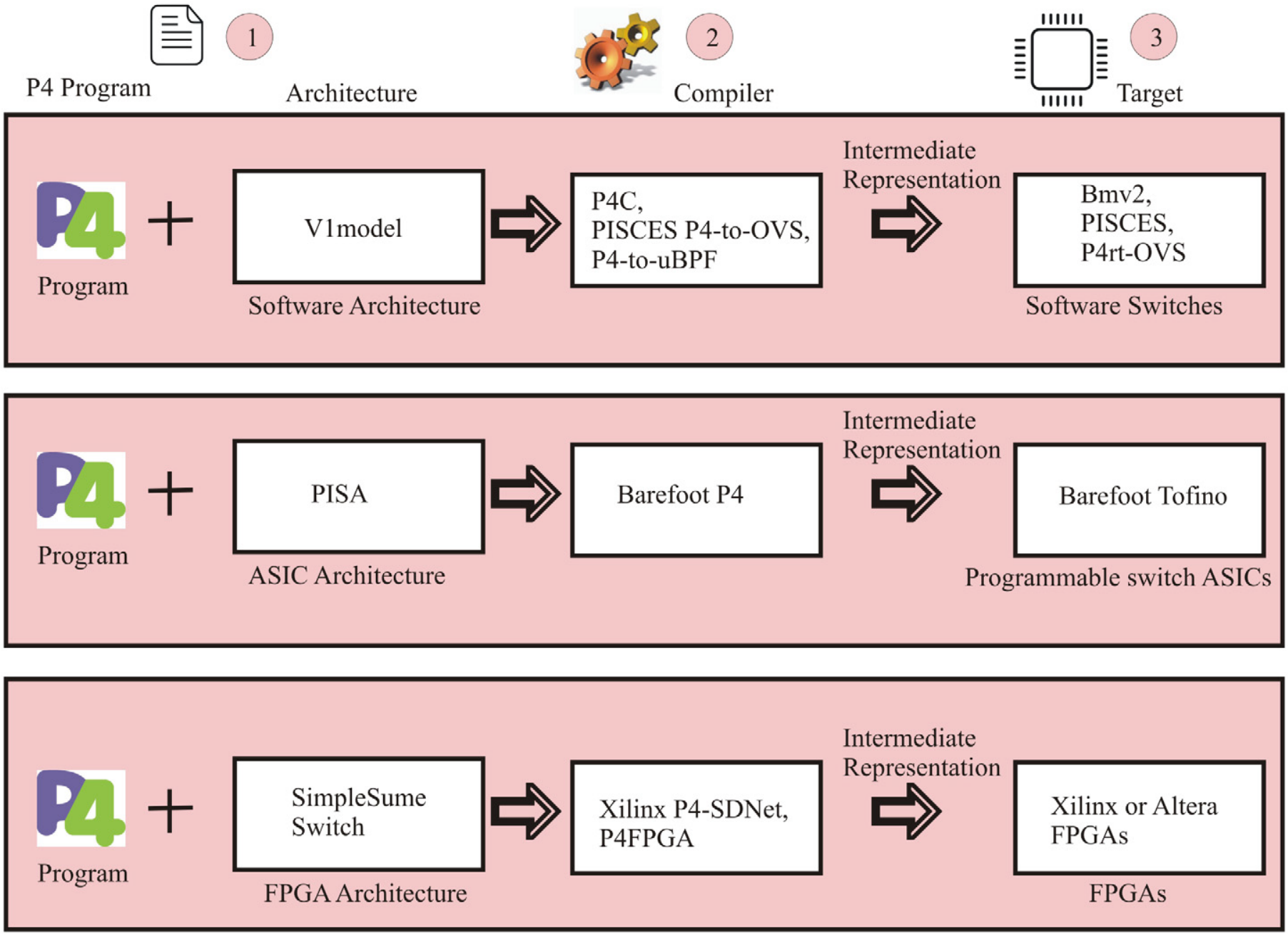
# P4 - Packet forwarding model



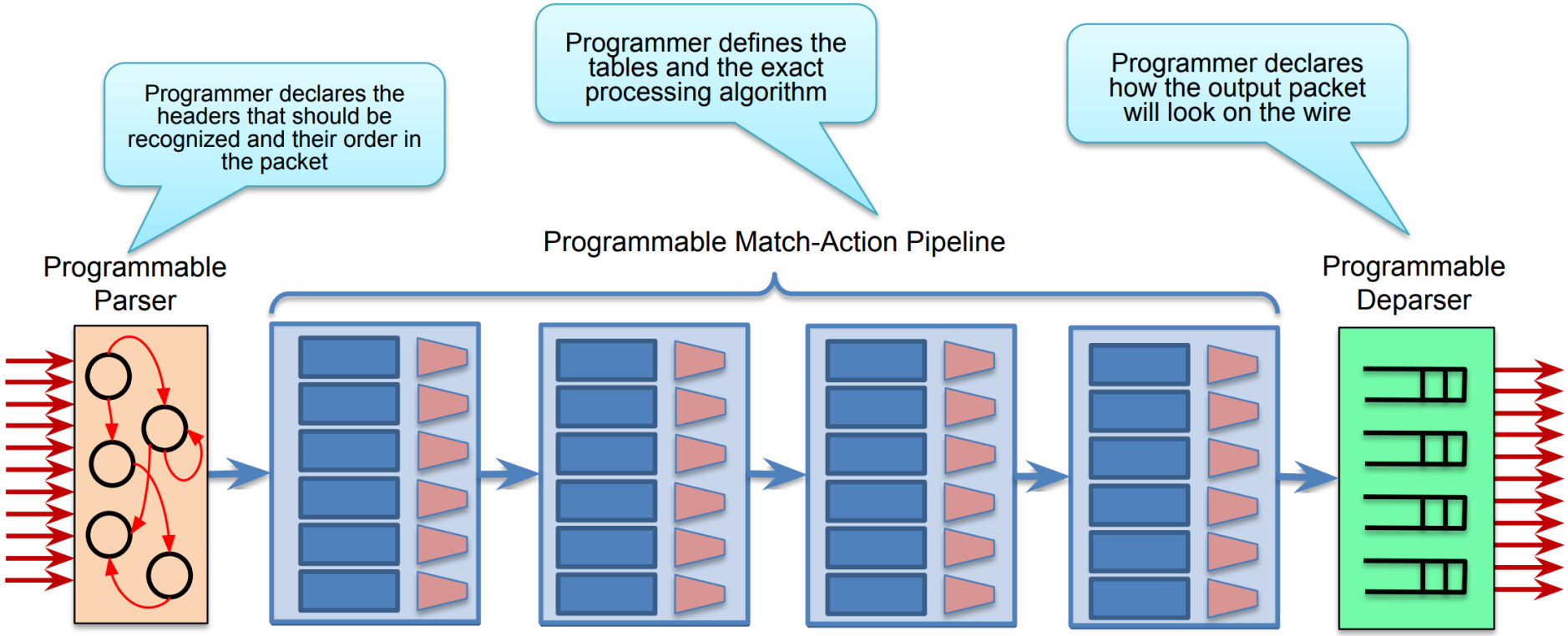
# P4 Language Concepts

- Headers: A header definition describes the sequence and structure of a series of fields.
- Parsers: A parser definition specifies how to identify headers and valid header sequences within packets.
- Tables: Match+action tables are the mechanism for performing packet processing.
- Actions: P4 supports construction of complex actions from simpler protocol-independent primitives.
- Control Programs: The control program determines the order of match+action tables that are applied to a packet.


User supplied
Control Plane
P4 Program
P4 Compiler
Add/remove table entries
Extern control
Packet-in/out
CPU port
P4 Architecture Model
Target-specific configuration binary
Load
Tables
Extern objects
Data Plane
Vendor supplied
Runtime control


1
P4 Program
Architecture
2
Compiler
3
Target
Program
V1model
Software Architecture
P4C,
PISCES P4-to-OVS,
P4-to-uBPF
Intermediate
Representation
Bmv2,
PISCES,
P4rt-OVS
Software Switches
Program
PISA
ASIC Architecture
Barefoot P4
Intermediate
Representation
Barefoot Tofino
Programmable switch ASICs
Program
SimpleSume
Switch
FPGA Architecture
Xilinx P4-SDNet,
P4FPGA
Intermediate
Representation
Xilinx or Altera
FPGAs
FPGAs

# PISA: Protocol-Independent Switch Architecture

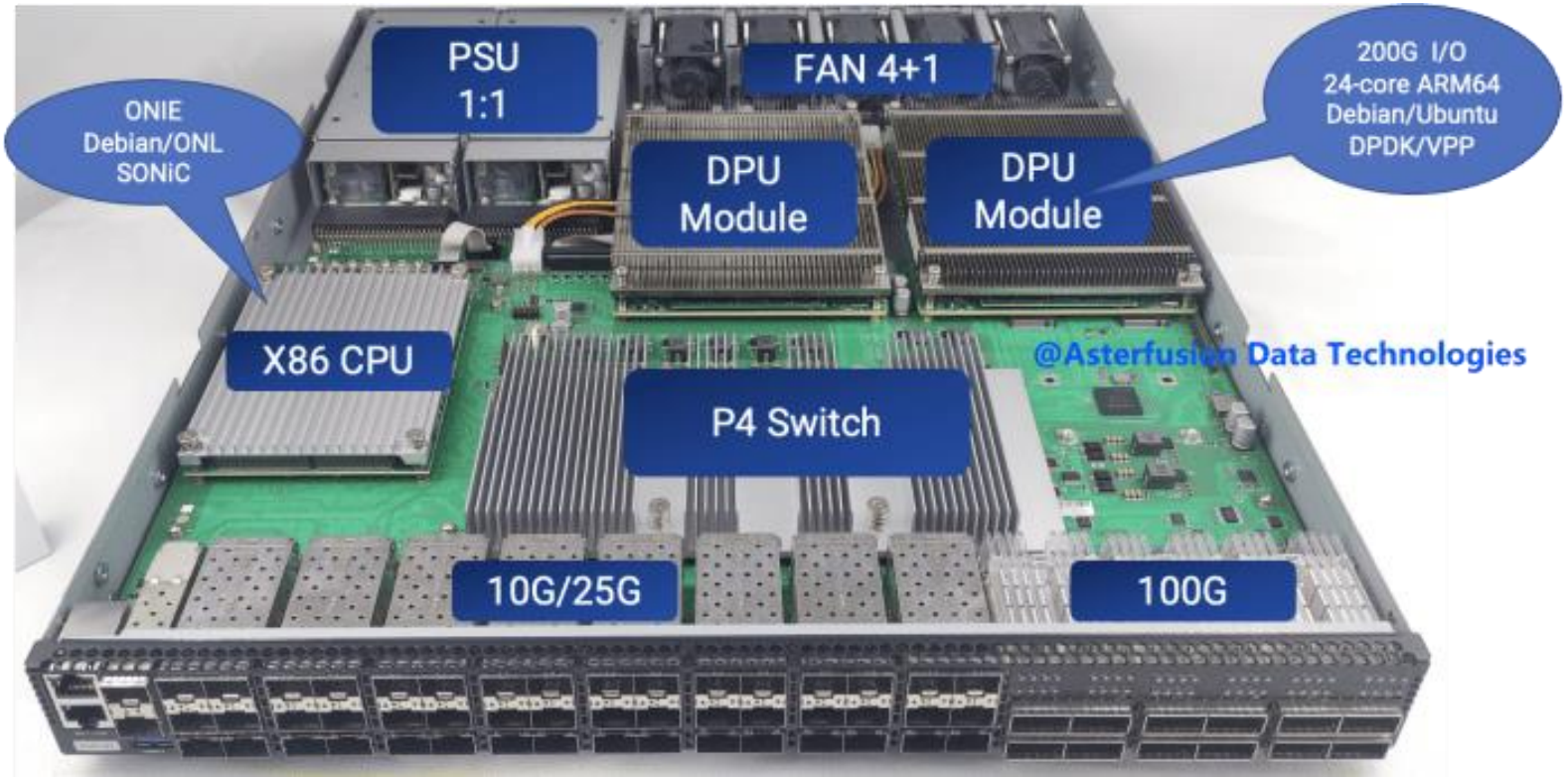




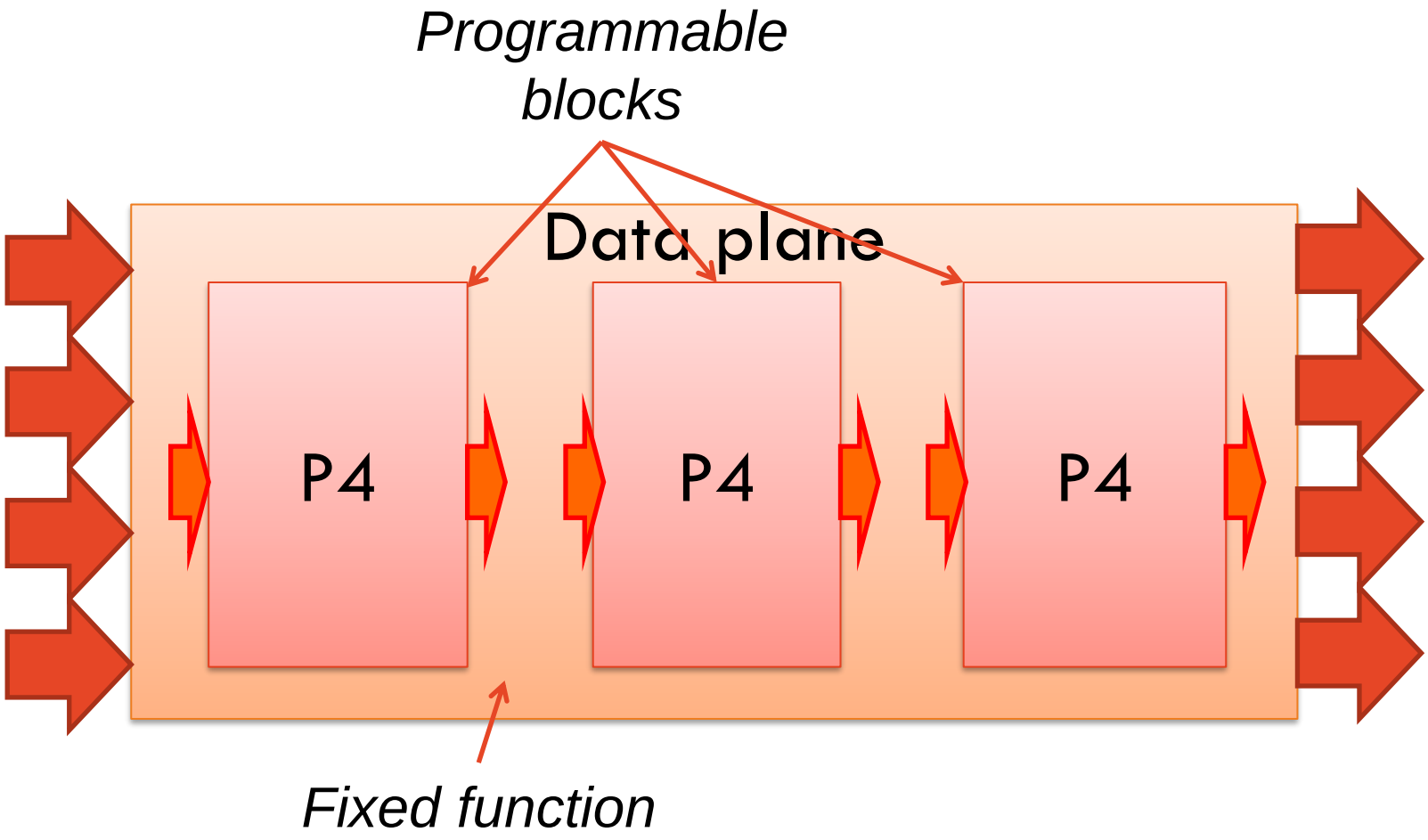

https://opennetworking.org/wp-content/uploads/2020/12/P4_D2_East_2018_01_basics.pdf

# Inside a P4 switch

– Asterfusion X312P-48Y-T P4 Programmable switch



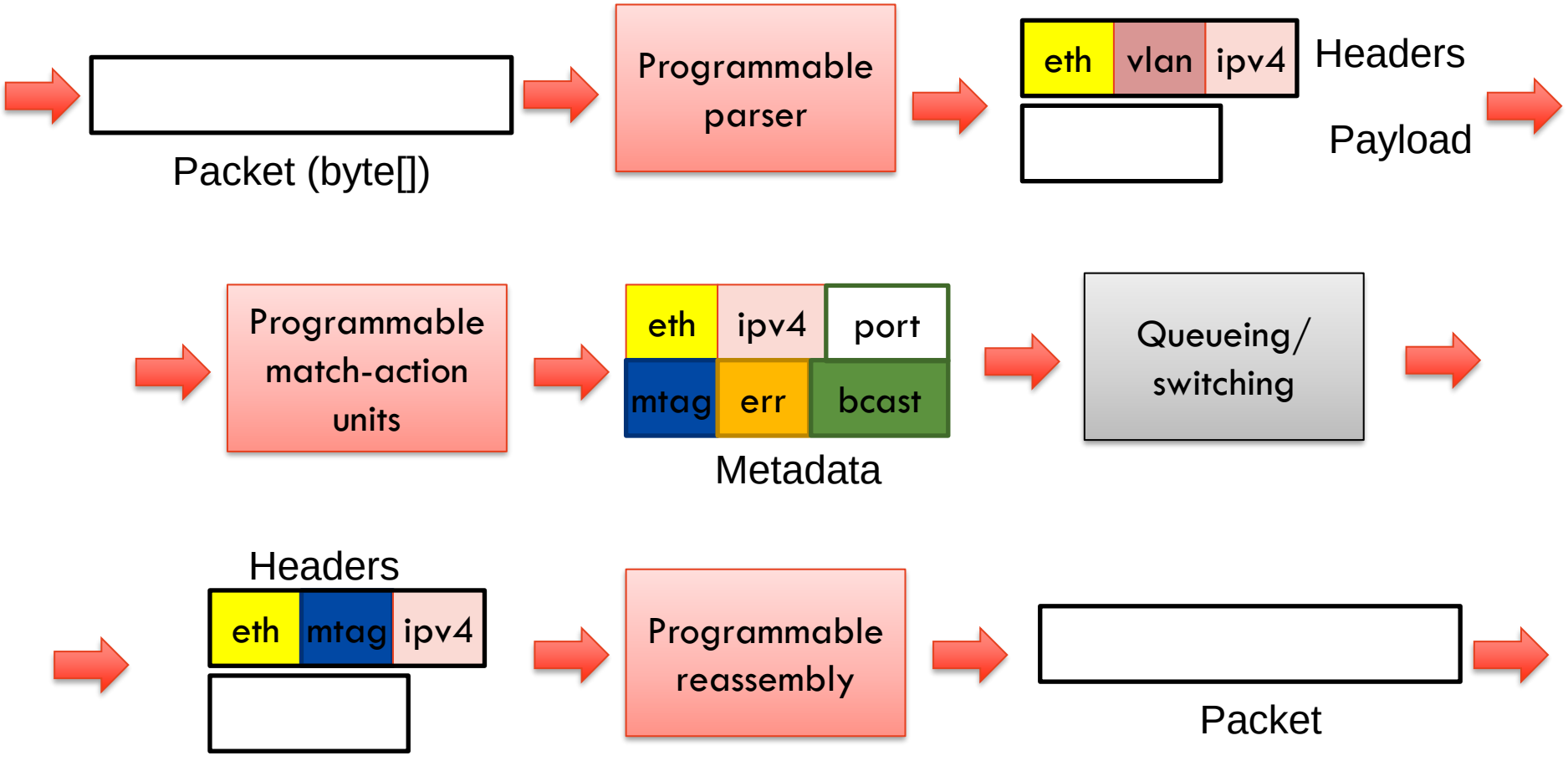
# $P4_{16}$ data plane model



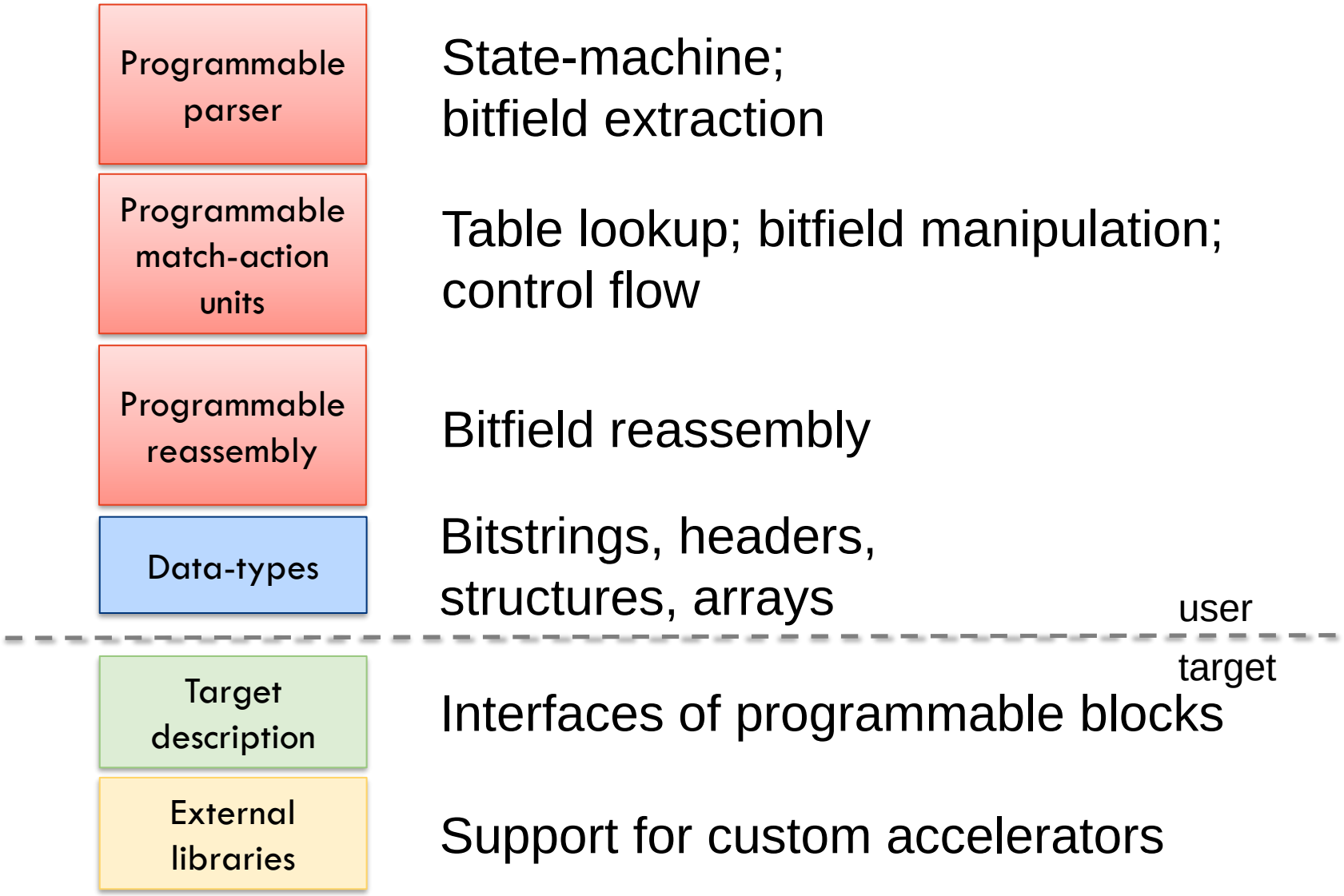
# P4 Compiler

- Parser
  - Programmable parser: translate to state machine
  - Fixed parser: verify the description is consistent
- Control program
  - Target-independent: table graph of dependencies
  - Target-dependent: mapping to switch resources
- Rule translation
  - Verify that rules agree with the (logical) table types
  - Translate the rules to the physical tables

# Example packet processing pipeline



# Language elements




No: memory (pointers), loops, recursion, floating point, complex data structures

# P4 Standard Metadata

- **ingress_port** - the port on which the packet arrived
- **egress_spec** - the port to which the packet should be sent to
- **egress_port** - the port on which the packet is departing from (read only in egress pipeline)

```
struct standard_metadata_t {
    bit<9>  ingress_port;
    bit<9>  egress_spec;
    bit<9>  egress_port;
    bit<32> clone_spec;
    bit<32> instance_type;
    bit<1>  drop;
    bit<16> recirculate_port;
    bit<32> packet_length;
    bit<32> enq_timestamp;
    bit<19> enq_qdepth;
    bit<32> deq_timedelta;
    bit<19> deq_qdepth;
    bit<48> ingress_global_timestamp;
    bit<32> lf_field_list;
    bit<16> mcast_grp;
    bit<1>  resubmit_flag;
    bit<16> egress_rid;
    bit<1>  checksum_error;
}
```



https://opennetworking.org/wp-content/uploads/2020/12/P4_D2_East_2018_01_basics.pdf

# P4 Program Template

```
#include <core.p4>
#include <v1model.p4>
/* HEADERS */
struct metadata { ... }
struct headers {
  ethernet_t   ethernet;
  ipv4_t       ipv4;
}
/* PARSER */
parser MyParser(packet_in packet,
          out headers hdr,
          inout metadata meta,
          inout standard_metadata_t smeta) {
  ...
}
/* CHECKSUM VERIFICATION */
control MyVerifyChecksum(in headers hdr,
                         inout metadata meta) {
   ...
}
/* INGRESS PROCESSING */
control MyIngress(inout headers hdr,
                  inout metadata meta,
                  inout standard_metadata_t std_meta) {
  ...
}
```

```
/* EGRESS PROCESSING */
control MyEgress(inout headers hdr,
                 inout metadata meta,
                 inout standard_metadata_t std_meta) {
  ...
}
/* CHECKSUM UPDATE */
control MyComputeChecksum(inout headers hdr,
                          inout metadata meta) {
  ...
}
/* DEPARSER */
control MyDeparser(inout headers hdr,
                   inout metadata meta) {
  ...
}
/* SWITCH */
V1Switch(
  MyParser(),
  MyVerifyChecksum(),
  MyIngress(),
  MyEgress(),
  MyComputeChecksum(),
  MyDeparser()
) main;
```



https://opennetworking.org/wp-content/uploads/2020/12/P4_D2_East_2018_01_basics.pdf

# P4 Domains of Application - INT

- The parser and the de-parser are two important features of the language.
  - If used properly, then one can define custom protocols that meet the requirements and constraints of many environments.

- In-band Network Telemetry (INT)
  - INT is a framework designed to allow the collection and reporting of network state, by the data plane, without requiring intervention or work by the control plane in collecting and delivering the state from the data plane.
  - Packets may contain header fields that are interpreted as "telemetry instructions" by network devices.
  - The instructions may be programmed in the network data plane to match on particular network flows and to execute the instructions on the matched flows.

INT: https://github.com/p4lang/p4-applications/blob/master/docs/INT_v2_1.pdf

# P4 Domains of Application - INT

- Network Monitoring
  - Process statistics collections or perform certain actions when an event is triggered.
- Traffic Engineering
  - Load Balancing, Routing, Congestion Control, etc.
- Function Offloading
  - Simple applications can be to offloaded to data plane.
- Cross-cutting Domains
  - In-network computation, industrial control systems, etc.
- Security
  - Data plane firewall, DoS detection, etc.
- Target-specific Optimizations
  - P4 can be compiled in multiple target-specific configuration formats.

Liatifis, Athanasios, et al. "Advancing sdn from openflow to p4: A survey." *ACM Computing Surveys* 55.9 (2023): 1-37.

# P4 Portable NIC Architecture

- New application of P4

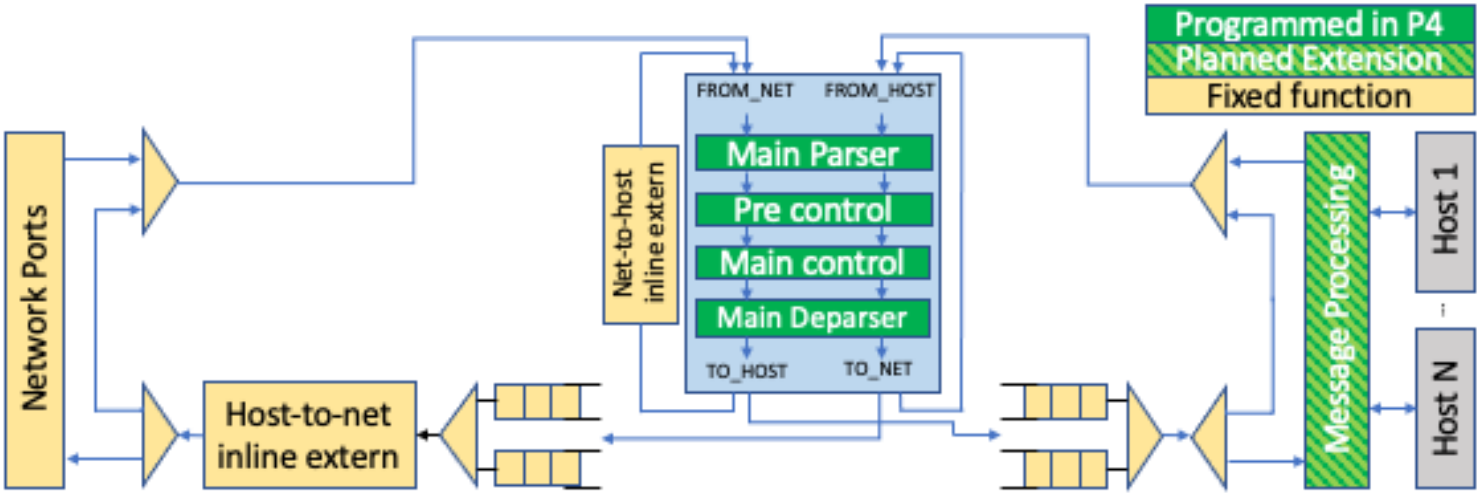
- A P4 program comprises an architecture, which describes the structure and capabilities of the pipeline, and a user program, which specifies the functionality of the programmable blocks within that pipeline.

- The Portable NIC Architecture (PNA) is an architecture that describes the structure and common capabilities of network interface controller (NIC) devices that process packets going between one or more interfaces and a host system.



https://p4.org/p4-spec/docs/PNA.html

# P4 Portable NIC Architecture




The Portable NIC Architecture (PNA) Model has four programmable P4 blocks and several fixed-function blocks. The behavior of the programmable blocks is specified using P4.

The network ports, packet queues, and (optional) inline extern blocks are fixed-function blocks that can be configured by the control plane, but are not intended to be programmed using P4.



https://p4.org/p4-spec/docs/PNA.html

# P4runpro: Enabling Runtime Programmability for RMT Programmable Switches

- Current programmable ASICs, a.k.a., Reconfigurable Match-Action Table (RMT) switches, can only specify data plane programs at compile-time and are unable to update the switch data plane in a runtime manner.
  - When a running program needs to be changed, the operator has to reprovision the switch, causing disruptions to traffic and suspending unrelated switch programs.
  - This limitation stems from P4's limited level of abstraction, i.e., it only defines a single data plane context. As a result, all switch programs defined by P4 are bound to and share the same hardware resources.

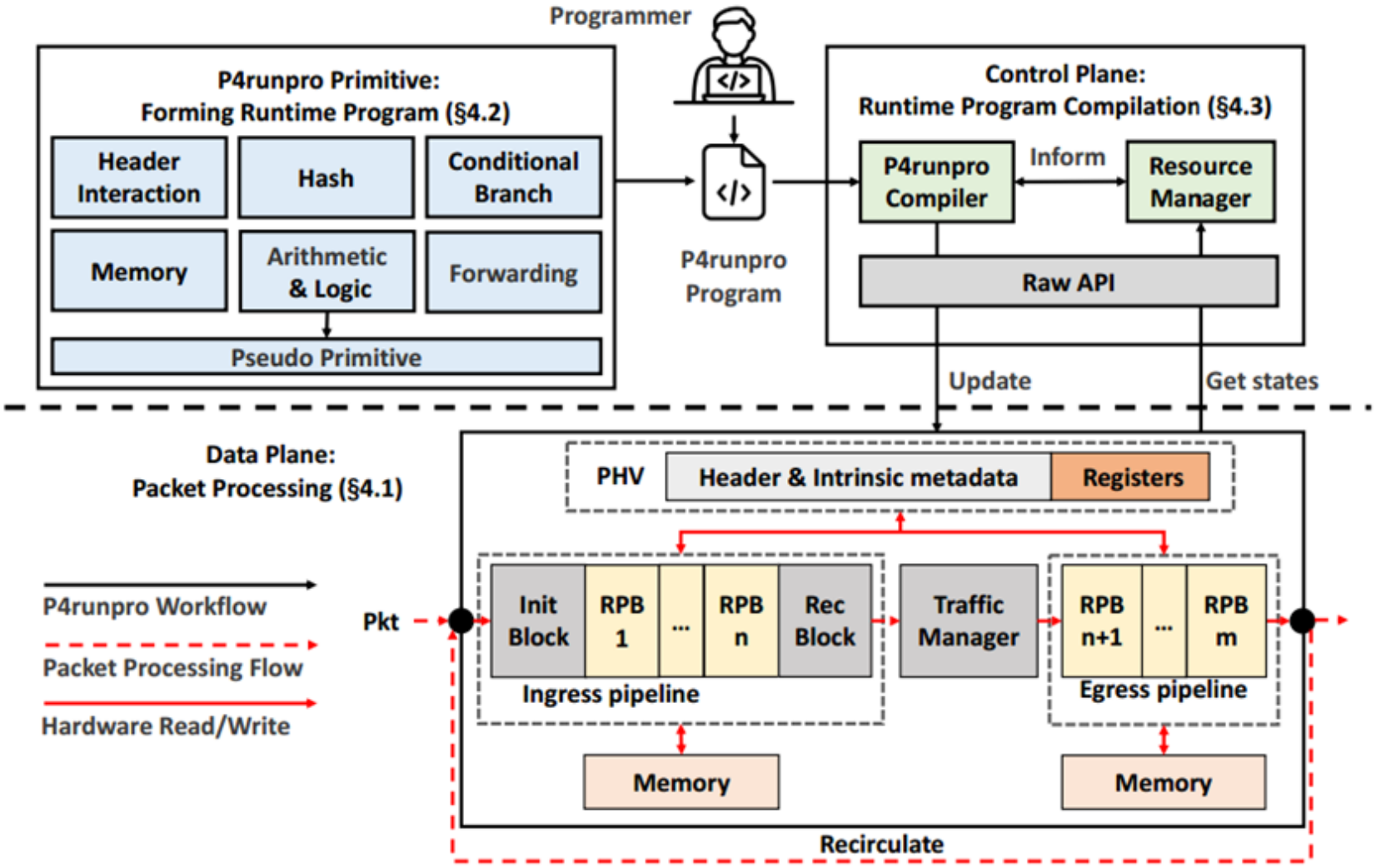
- P4runpro enables runtime programmability for RMT ASICs, which allows for data plane updates without the need for switch reprovisioning, ensuring no disruptions to ongoing traffic and programs.

# P4runpro – Core Idea

Decouple the implementation of switch programs from hardware resources.

- Identify and abstract a set of atomic operations from a variety of heterogeneous switch programs.
  - Design P4runpro primitives and pseudo primitives as runtime programming interfaces.
- These atomic operations are pre-installed at compile-time and can then be flexibly combined into various target programs at runtime.
  - P4runpro compiler translate the input programs into entries and consistently update them to the data plane.
  - The key of the compiler is an efficient resource allocation scheme.

# P4runpro - Architecture



# Limitations of P4

- The core P4 language is very small
  - Highly portable among many targets
  - But very limited in expressivity
- Accelerators can provide additional functionality
  - May not be portable between different targets
  - Under construction: library of standard accelerators
- A Workshop for P4 applications
  - https://opennetworking.org/2022-p4-workshop-gated/



Source: Mihai Budiu, VMware Research

# P4 is not…

- Active networking: a way for packets to inject new code
- Programming the control plane: that is Software-Defined Networking
- A tool for third parties to program the network
- A language for:
  - distributed computations
  - network middleboxes (NFV)
  - network operating systems

Source: Mihai Budiu, VMware Research

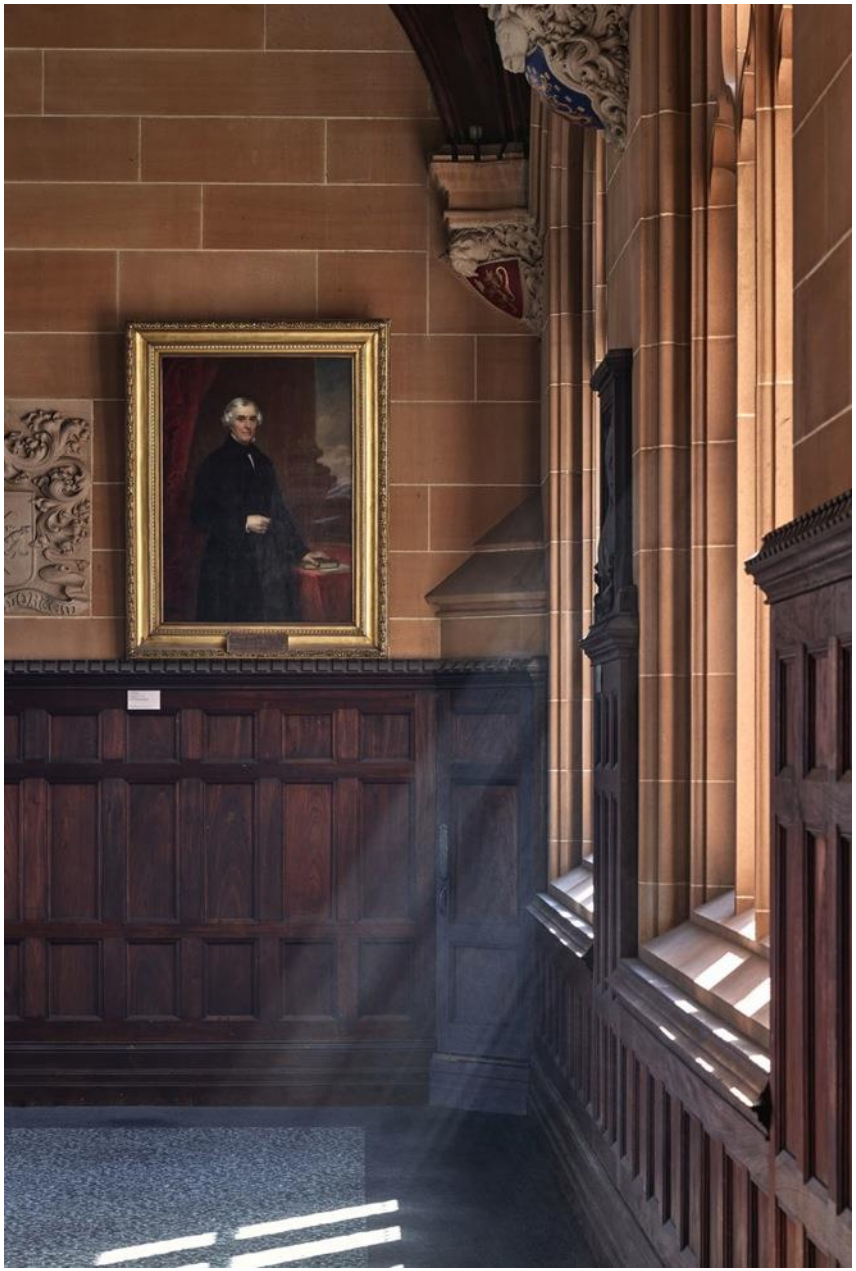
# Thank you!

References:

https://p4.org/p4-spec/docs/PNA.html

https://www.sdxcentral.com/sdn/?c_action=num_ball

https://www.opennetworking.org/



THE UNIVERSITY OF SYDNEY