ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Introduction to P4

Khanh Nam Chu

ONE LOVE. ONE FUTURE.

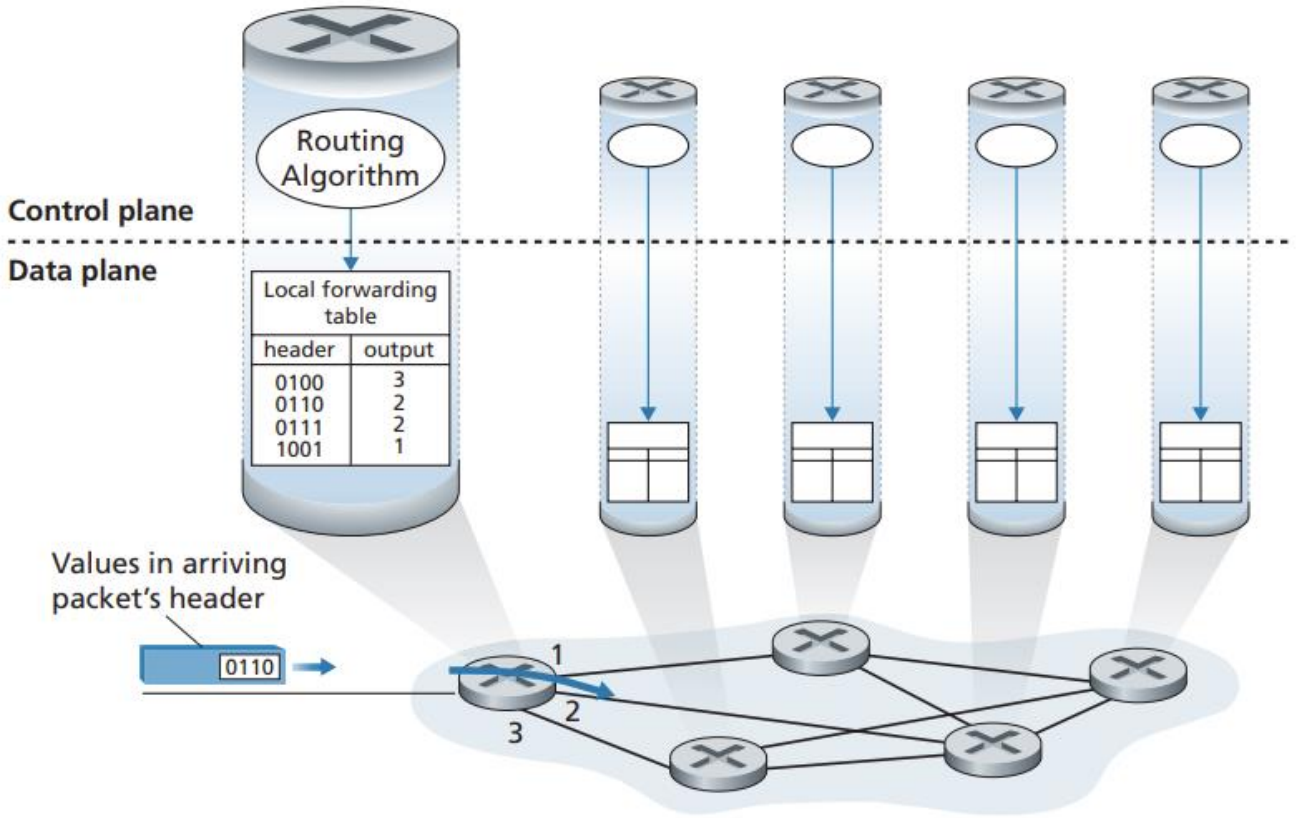
CONTENTS

1. SDN

Traditional Network:

- The control plane and data plane are installed within a packet switch.
- The control plane runs different algorithms to figure out what the network looks like, what paths are available.
- The control plane tells the data plane how to forward packets accordingly.

→ Require manually configuration, complex, and not effective.

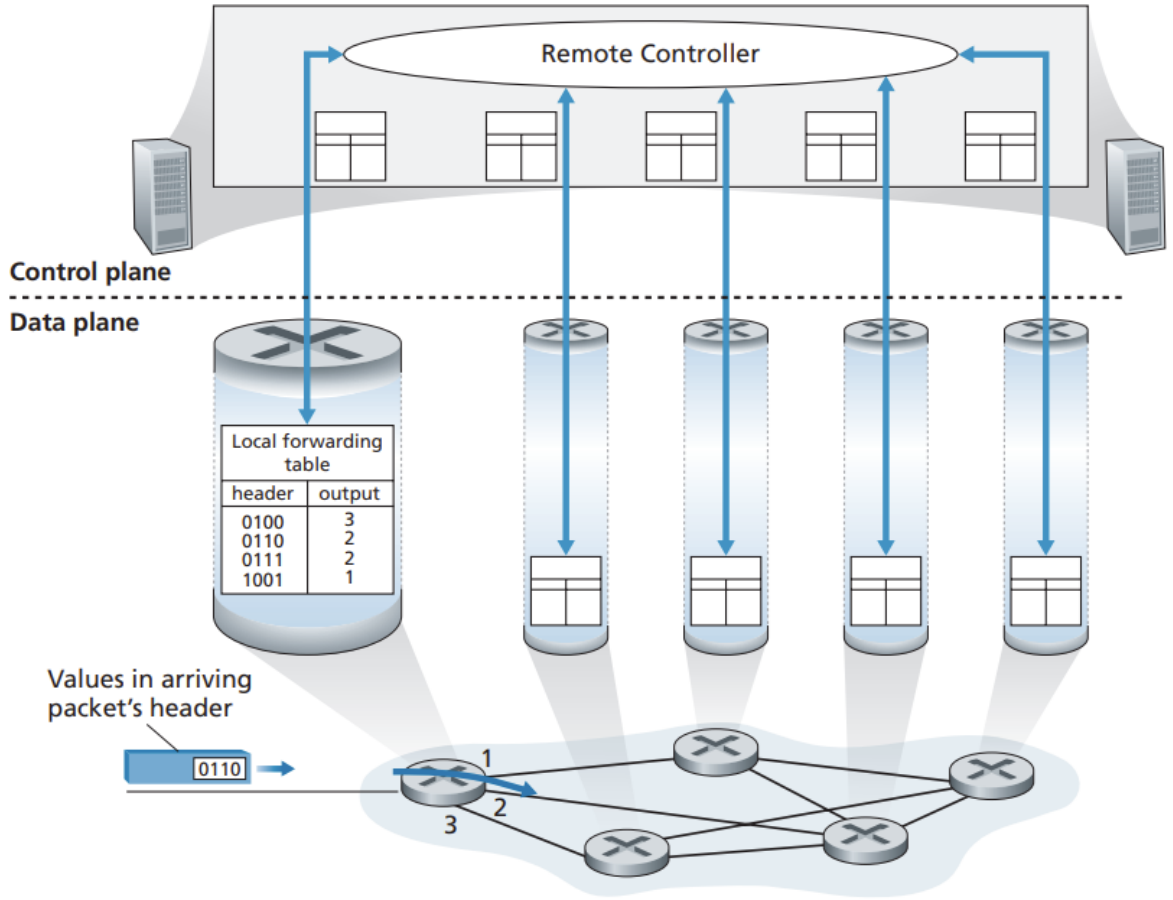


1. SDN

Software-Defined Network:

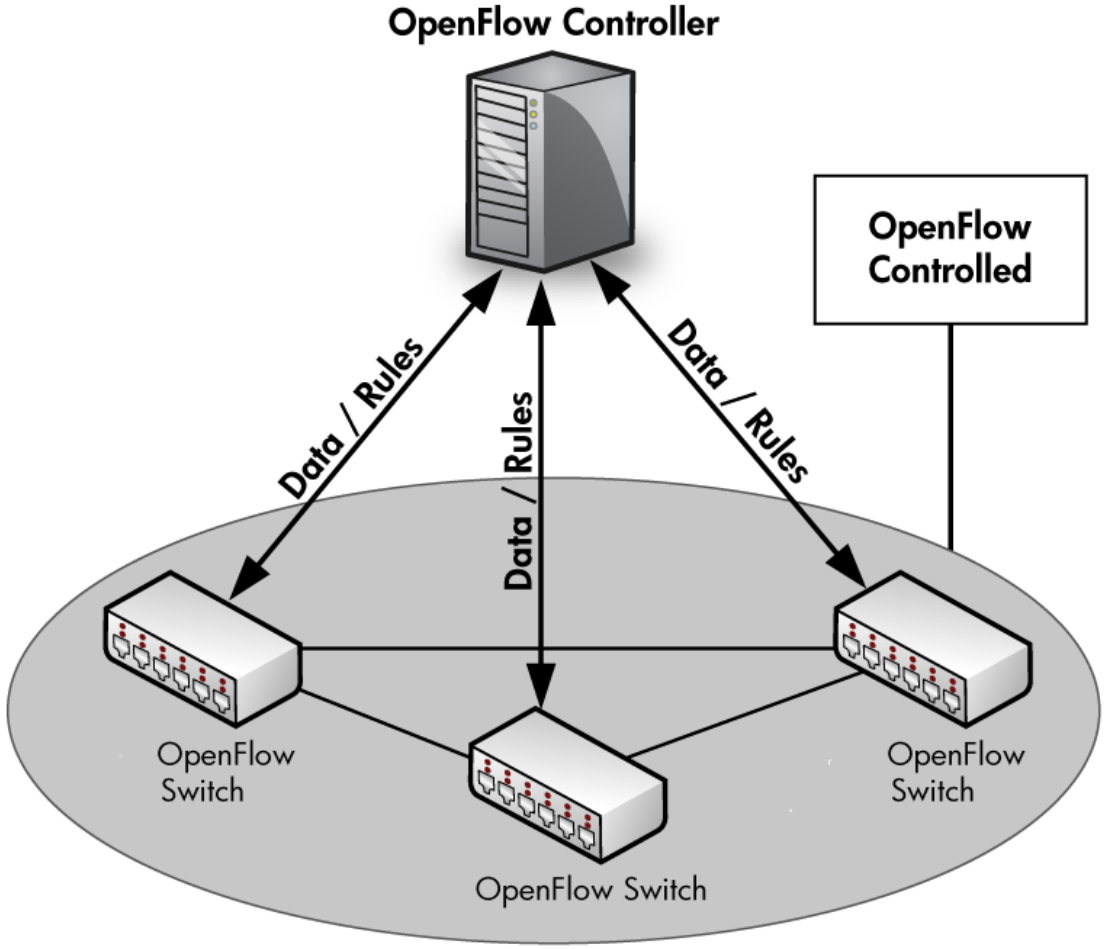
- The control plane and data plane are separated.
- The routing device performs forwarding only, while the remote controller computes and distributes forwarding tables. The control plane tells the data plane how to forward packets accordingly.
- The remote controller might be implemented in a remote data center with high reliability and redundancy, and might be managed by the ISP or some third party.

→ Centralized network control, better network visibility.



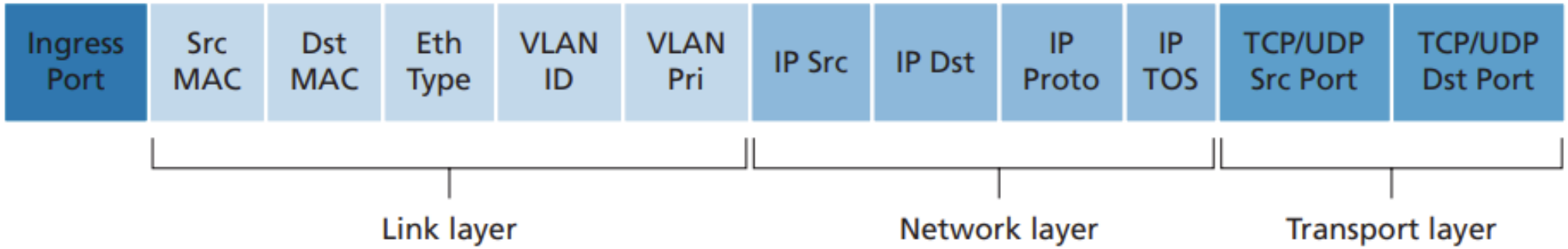
2. OpenFlow

- OpenFlow is a communications protocol used in Software-Defined Networking (SDN) to let the SDN controller directly interact with the forwarding plane (switches or routers).
- Separates the control plane (where decisions about routing are made) from the data plane (where packets are forwarded).
- Allows the controller to install rules (match-action entries) into the switch's flow tables.



2. OpenFlow

OpenFlow Match:



- The ingress port refers to the input port at the packet switch on which a packet is received.

- Flow table entries may also have wildcards. For example, an IP address of 128.119.*.* in a flow table will match the corresponding address field of any datagram that has 128.119 as the first 16 bits of its address. Each flow table entry also has an associated priority. If a packet matches multiple flow table entries, the selected match and corresponding action will be that of the highest priority entry with which the packet matches.

2. OpenFlow

OpenFlow Action:

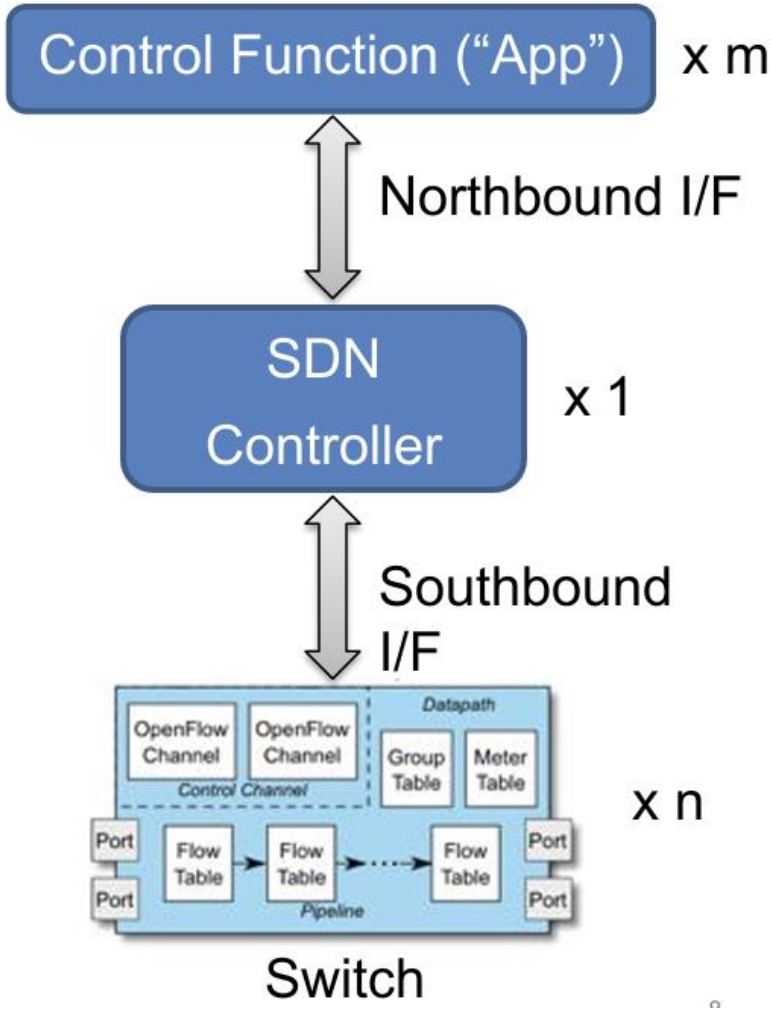
- Forwarding: An incoming packet may be forwarded to a particular physical output port, broadcast over all ports (except the port on which it arrived) or multicast over a selected set of ports. The packet may be encapsulated and sent to the remote controller for this device. That controller then may (or may not) take some action on that packet, including installing new flow table entries, and may return the packet to the device for forwarding under the updated set of flow table rules.

- Dropping: A flow table entry with no action indicates that a matched packet should be dropped.

- Modify-field: The values in 10 packet-header fields (all layer 2, 3, and 4 fields except the IP Protocol field) may be re-written before the packet is forwarded to the chosen output port.

3. The Problem for SDN

- Data-plane protocol evolution requires changes to standards (12 → 40 OpenFlow match fields).
- Limited interoperability between vendors (OpenFlow / netconf / JSON / XML variants).
- Limited programmability.

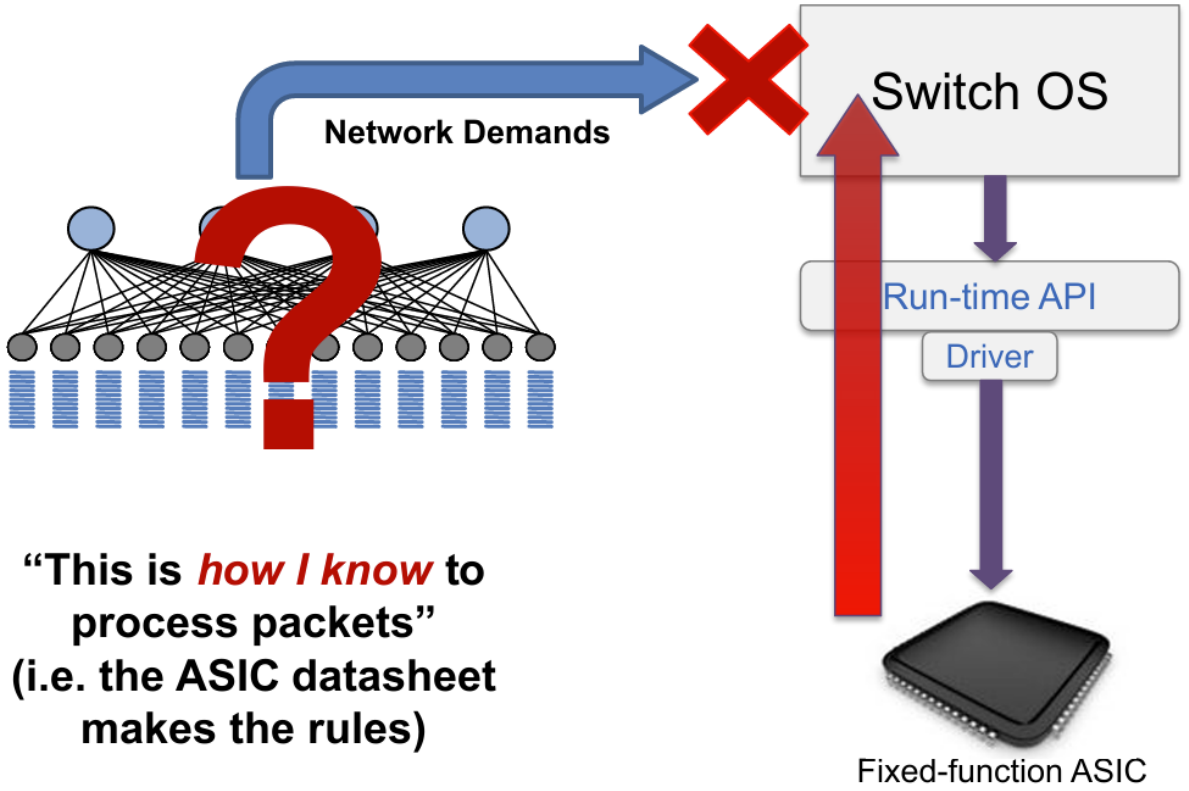


3. The Problem for SDN

Bottom-up design:

- The pipeline of the packet switch is fixed, you can't change or program the pipeline.
- The controller based on the rules of the pipeline to populate the forwarding table.

→ The data plane is hardware-fixed, high reliance on the controller can lead to bottlenecks and controller saturation.



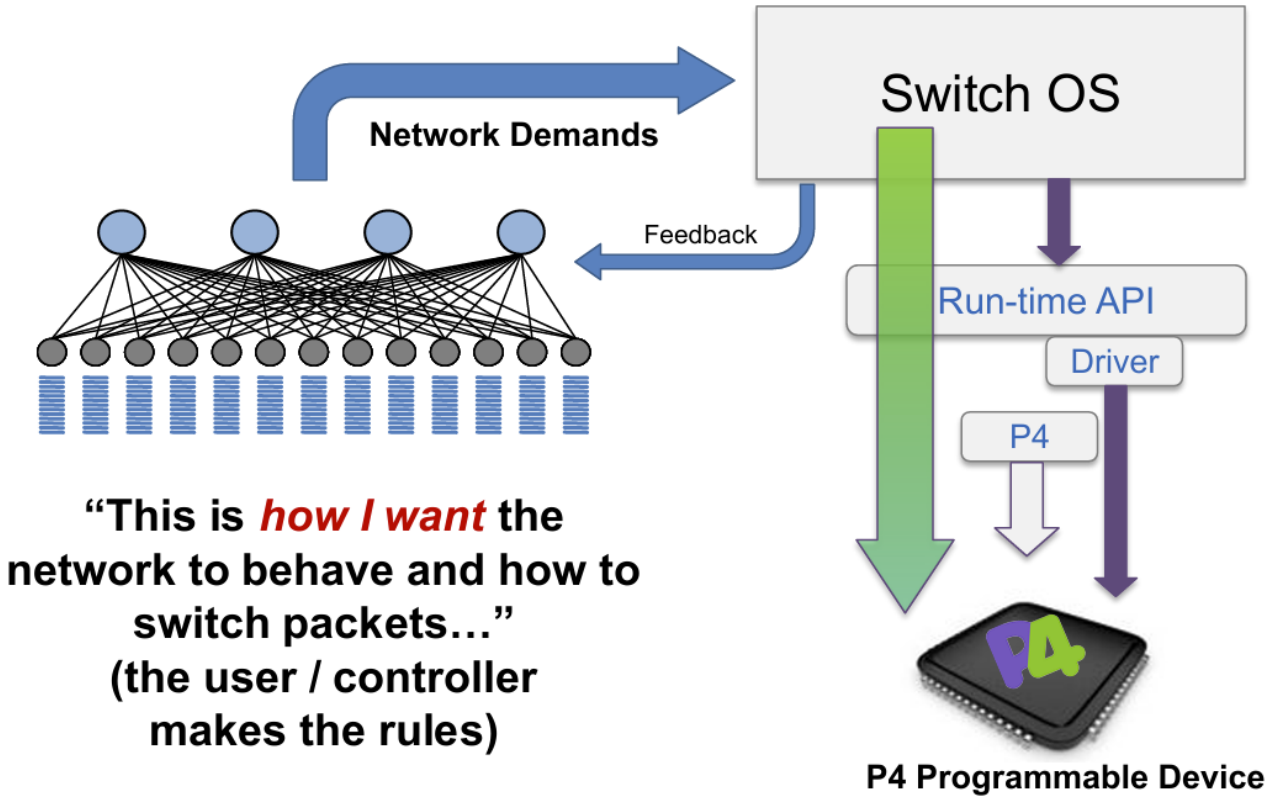
3. The Problem for SDN

Bottom-down design:

- The user is able to program the data plane.
- The user can define his/her own protocols.
- The user can define how switches process data, match-action tables.

→ Data-programmable network, user-defined network rather than based on traditional fixed protocols.

→ P4 programming language.



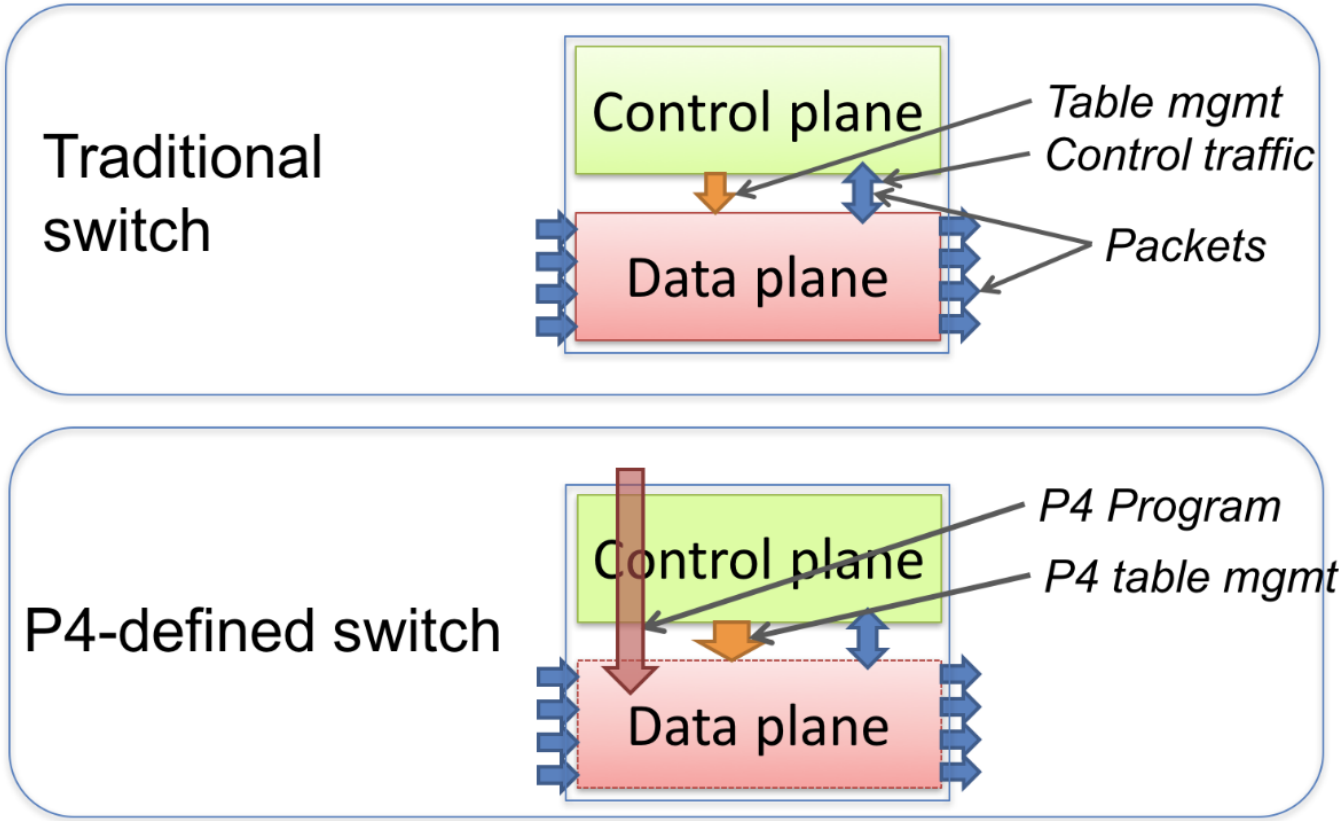
4. P4 programming language

Brief History of P4

- May 2013: Initial idea and the name “P4”
- July 2014: First paper (SIGCOMM CCR)
- Aug 2014: First P414 Draft Specification (v0.9.8)
- Sep 2014: P414 Specification released (v1.0.0)
- Jan 2015: P414 v1.0.1
- Mar 2015: P414 v1.0.2
- Nov 2016: P414 v1.0.3
- May 2017: P414 v1.0.4
- Apr 2016: P416 – first commits
- Dec 2016: First P416 Draft Specification
- May 2017: P416 Specification released
- October 2024: Stable release version 1.2.5

4. P4 programming language

Programming Protocol-Independent Packet Processors: P4 is a language for expressing how packets are processed by the data plane of a programmable forwarding element such as a hardware or software switch, network interface card, router, or network appliance.



Traditional switches vs. programmable switches.

4. P4 programming language

A target is a packet-processing system capable of executing a P4 program.

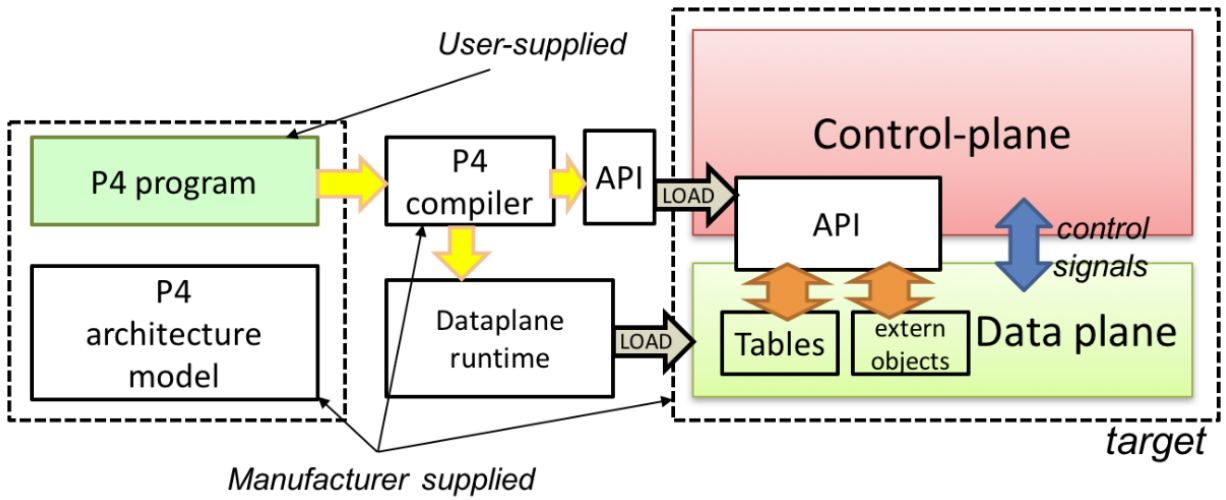
The core abstractions provided by the P4 language are:

- Header types describe the format (the set of fields and their sizes) of each header within a packet.

- Parsers describe the permitted sequences of headers within received packets, how to identify those header sequences, and the headers and fields to extract from packets.

- Tables associate user-defined keys with actions.

- Actions are code fragments that describe how packet header fields and metadata are manipulated. Actions can include data, which is supplied by the control-plane at runtime (Metadata is an intermediate data generated during execution of a P4 program).



Programming a target with P4.

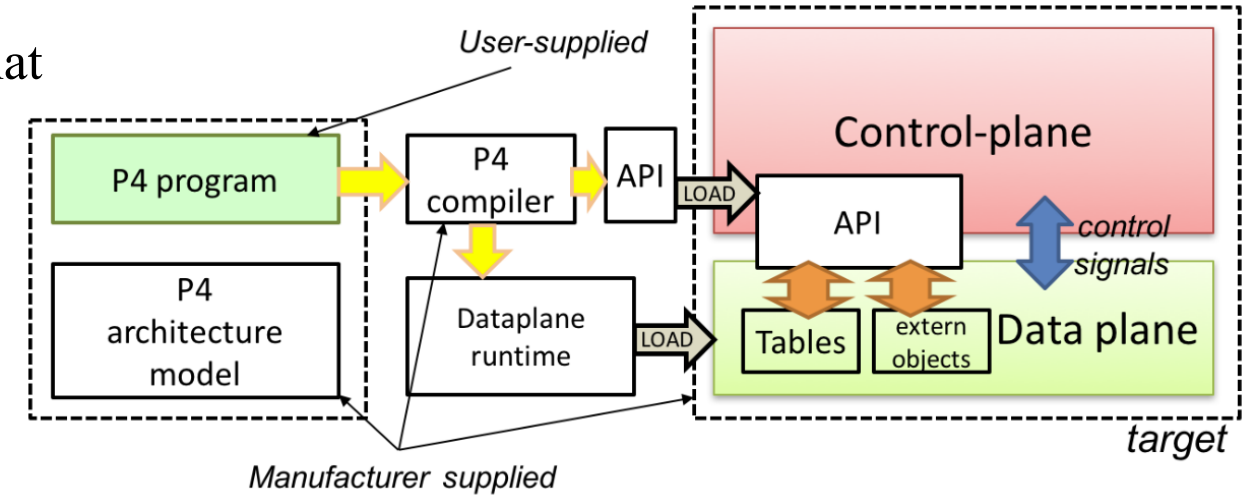
4. P4 programming language

- Match-action units perform the following sequence of operations:
  - Construct lookup keys from packet fields or computed metadata,
  - Perform table lookup using the constructed key, choosing an action (including the associated data) to execute, and
  - Finally, execute the selected action.

- Control flow expresses an imperative program that describes packet-processing on a target, including the data-dependent sequence of match-action unit invocations. Deparsing (packet reassembly) can also be performed using a control flow.

- Extern objects are architecture-specific constructs that can be manipulated by P4 programs through well-defined APIs, but whose internal behavior is hard-wired and hence not programmable using P4.

- Intrinsic metadata: metadata provided by the architecture associated with each packet.

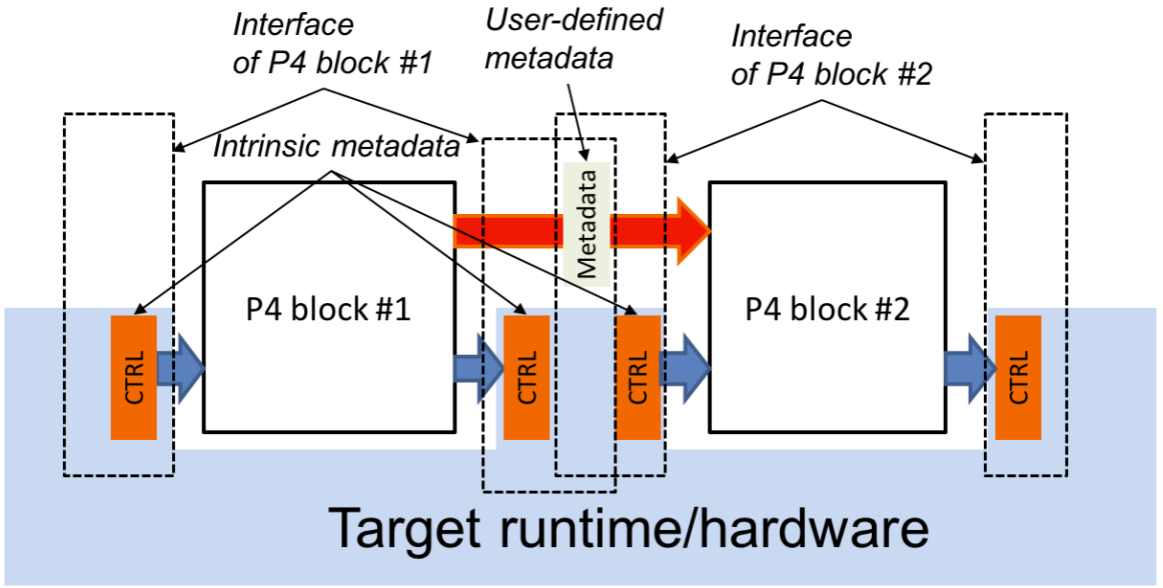


Programming a target with P4.

4. P4 programming language

Architecture Model

- The *P4 architecture* identifies the P4-programmable blocks (e.g., parser, ingress control flow, egress control flow, deparser, etc.) and their data plane interfaces.
- The P4 architecture can be thought of as a contract between the program and the target. Each manufacturer must therefore provide both a P4 compiler as well as an accompanying architecture definition for their target.

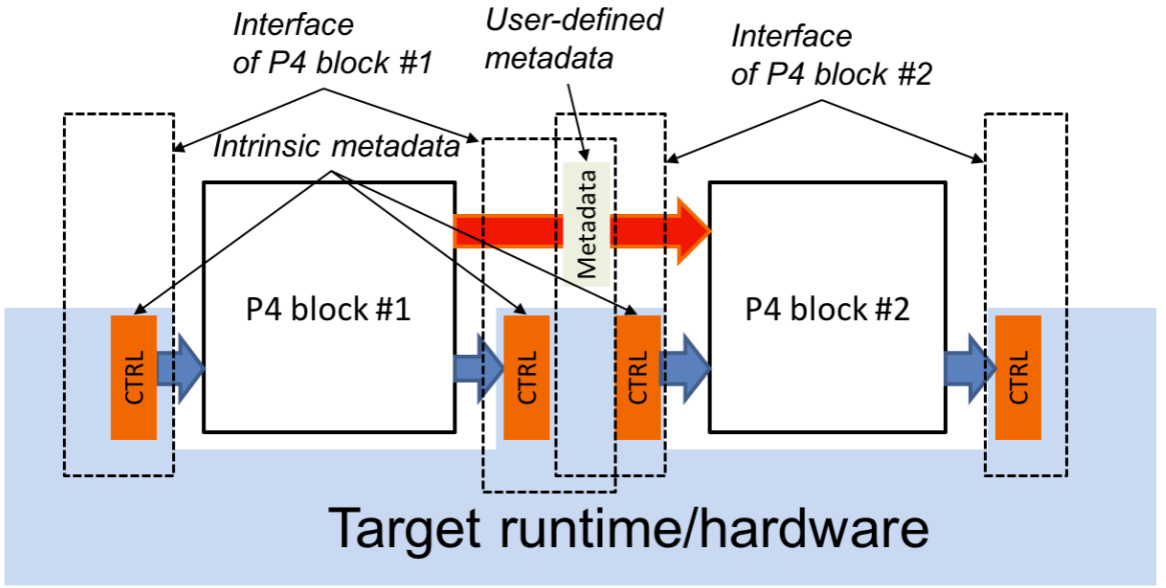


P4 program interfaces.

4. P4 programming language

Architecture Model

- The Figure illustrates the data plane interfaces between P4-programmable blocks. It shows a target that has two programmable blocks (#1 and #2). Each block is programmed through a separate fragment of P4 code. The target interfaces with the P4 program through a set of control registers or signals.
- Input controls provide information to P4 programs (e.g., the input port that a packet was received from), while output controls can be written to by P4 programs to influence the target behavior (e.g., the output port where a packet has to be directed).
- Control registers/signals are represented in P4 as *intrinsic metadata*. P4 programs can also store and manipulate data pertaining to each packet as *user-defined metadata*.

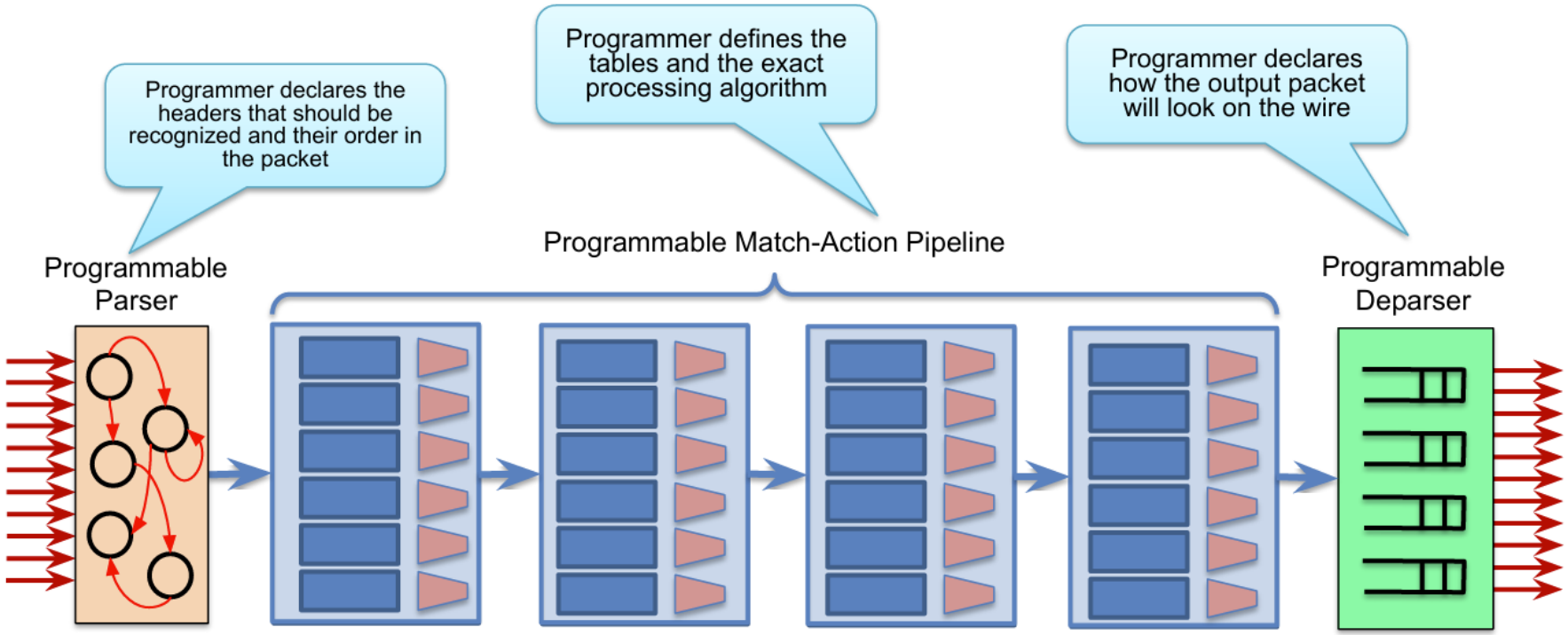


P4 program interfaces.

4. P4 programming language

PISA (Protocol-Independent Switch Architecture)

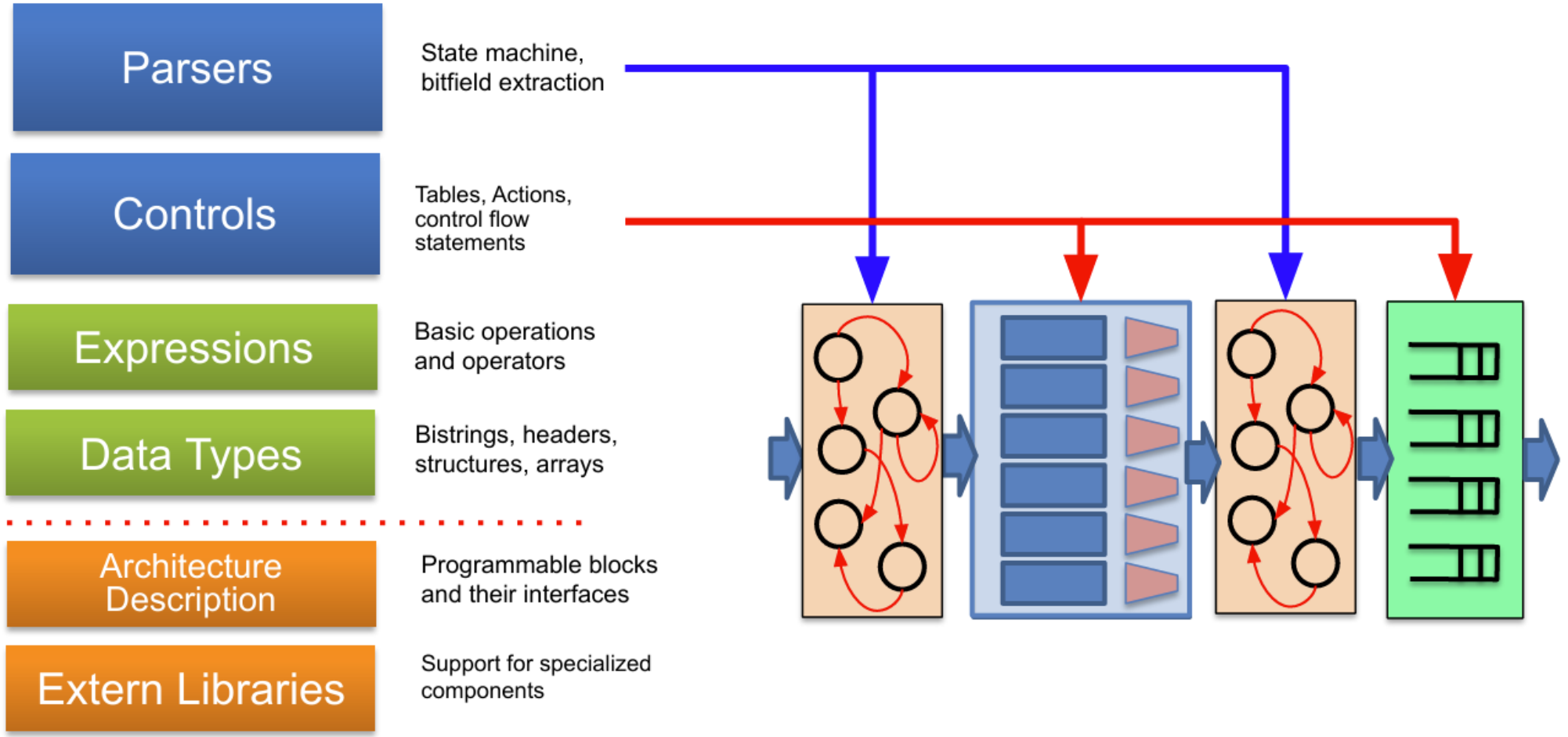
- A hardware and software architecture model that P4 was originally designed to program.
- It describes how a P4-programmable switch is structured, so that the P4 code can be compiled and run on it.



PISA Architecture

4. P4 programming language

$P4_{16}$ Language Elements



$P4_{16}$ Elements

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !