P4: A Hands-on Introduction

Presenters: Bapi Vinnakota
Lab Instructors: Mary Pham, Jici Gao

Abstract

P4 is a new declarative language to describe packet processing in software-defined networks (SDNs). The language extends the match-action dataflow developed in OpenFlow by allowing protocol processing in the dataplane to be specified programmatically. P4 offers developers and network operators several advantages. For applications such as NFV, the dataplane can grow to process new protocols, or change as the protocol mix changes according to the functions deployed. P4 can be used as a mechanism to fully specify the functionality of the dataplane. A single P4 codebase specifying the dataplane can be ported to multiple hardware platforms. Researchers have explored using P4 to enhance network functionality in a wide range of use cases including network telemetry, load balancing, consensus as a service and others. P4 is gradually being integrated into a many open source networking projects such as Open vSwitch, and OpenSwitch.

This tutorial will introduce attendees to developing and executing P4 code for a match-action dataflow-based dataplane. The focus of the tutorial will be hands-on labs in which attendees will develop and execute P4 programs on production hardware, Intelligent Server Adapters (ISAs) from Netronome. Attendees will learn how to develop, test and grow a networking datapath in P4. We will start with a simple NIC expressed entirely in P4. In Netronome ISAs. To implement architecture-specific functions or functions not supported in P4, developers can implement custom actions in a P4 match-action dataflow in C. The tutorial will conclude with labs for C-based actions for flow timestamping, flow tagging and virtual network function (VNF) support for NFV applications.

Introduction

P4 is new declarative language to describe a match-action flow based dataplane in network processing. A P4 program describes in entirety the processing expected in the dataplane. All the packet headers and the fields of interest to the match operations are specified within the program. In addition to standard headers, network designers are also able to specify headers for custom protocols. A P4 program also lists the set of all possible actions to be supported across all tables in the flow. Finally, a P4 program specifies the set of tables, the control flow across the tables. Figure 1 shows the reference architecture from the P4 specification. For each packet, the headers of interest in the packet are extracted using a parser, specified in the program. In addition, the system may add metadata to each packet (not shown in the figure).

Based on the values in the headers and the metadata, the control flow in the program processes packets through a sequence of match-action tables. The websites p4.org and www.open-nfp.org contain several examples of P4 programs.

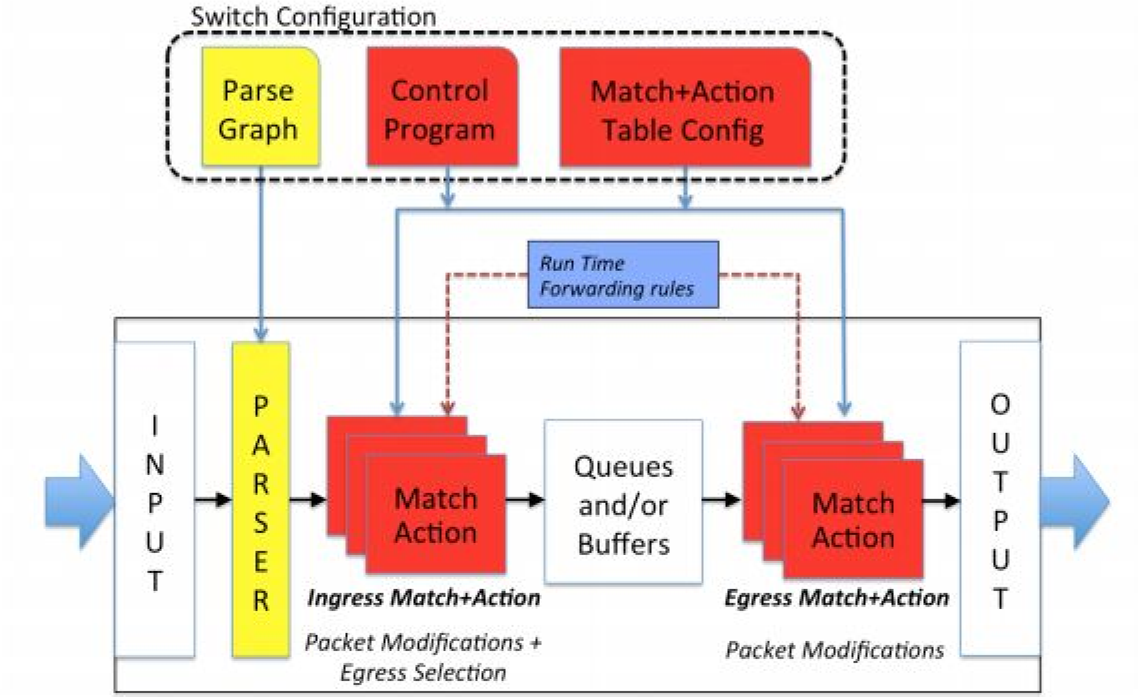
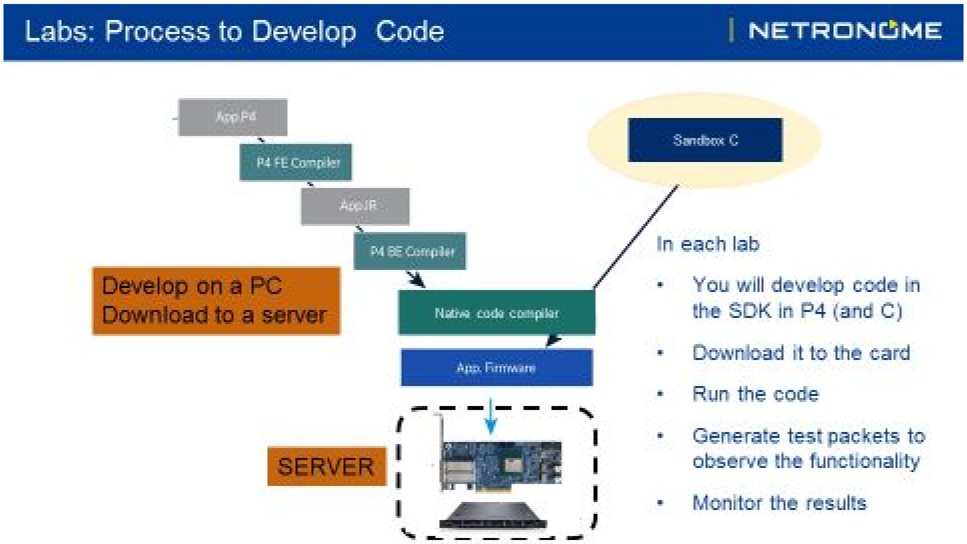


Figure 1: P4 Reference architecture

Netronome has a P4 tool chain to develop and compile P4 code to Network Flow Processors on its Agilio line of Intelligent Server Adapters. To the best of our knowledge, this is the only tool chain with an IDE, compiler, debugger in which engineers can develop and test P4 code on production networking hardware. Figure 2 shows the compile and build process for P4 code with NFPs. A developer creates P4 code describing the parser, match-action tables (with optional actions in C) and the control flow using the Netronome SDK. This code is compiled with tools in the SDK onto the processor.



Labs: Process to Develop Code

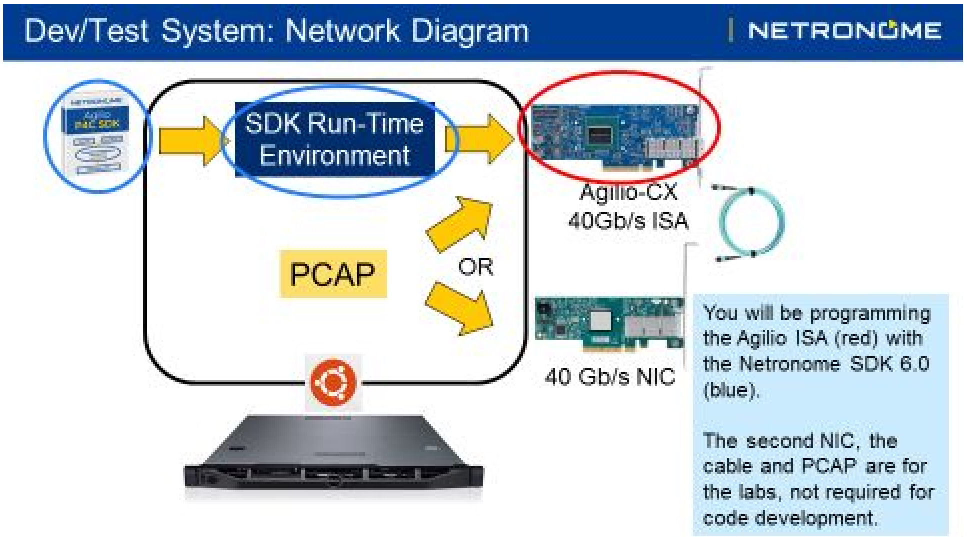
NETRONOME

In each lab

- You will develop code in the SDK in P4 (and C)
- Download it to the card
- Run the code
- Generate test packets to observe the functionality
- Monitor the results

Figure 2: P4 Development process for Netronome NFPs

Netronome has made the P4 IDE and Agilio ISAs available at a discount to researchers through www.open-nfp.org. The site contains three hands-on labs that introduce developers to the basics of the P4 language and other resources for developers. The tutorials are to be executed on the devtest system shown in Figure 3 below.



Dev/Test System: Network Diagram

NETRONOME

You will be programming the Agilio ISA (red) with the Netronome SDK 6.0 (blue).

The second NIC, the cable and PCAP are for the labs, not required for code development.

Figure 3: Dev/Test System for Labs

Tutorial

This session will be a hands-on introduction to the P4 language that will leverage the technology, content and infrastructure we have developed. The objective of the tutorial will be to enable attendees to understand the potential of the P4 language by actually developing and implementing basic networking functions in P4. We will also introduce the concept of “Custom Actions”, an architecture specific construct supported in the language. This enables networking code to provide features not currently available in the P4 language.

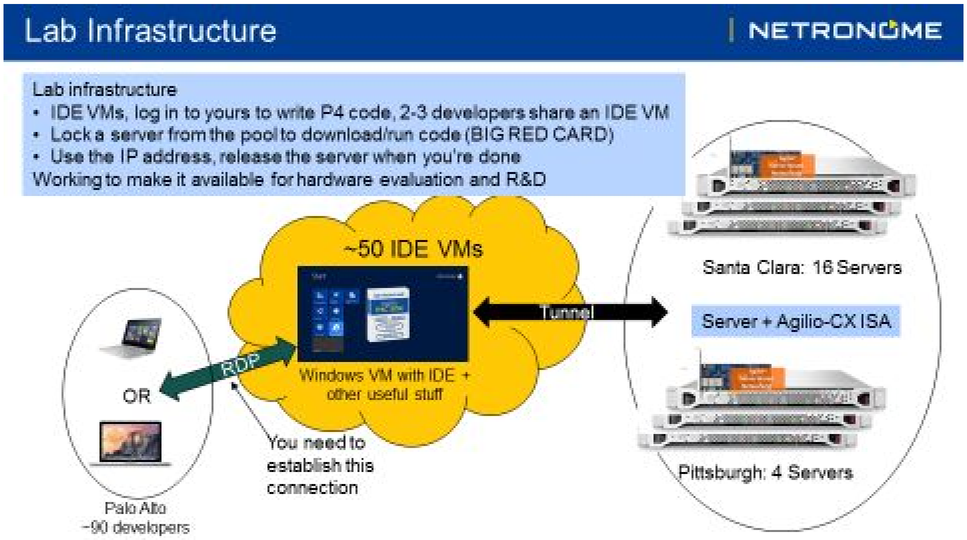
The lecture component of the tutorial will cover: \

1. The motivation for P4
2. A brief overview of the development history
3. Some potential use cases reported in the literature
4. Enough language detail to execute the labs

The lab component of the tutorial will cover:

1. Developing a NIC in P4, already exists on www.open-nfp.org
2. Timestamping packet activities in P4 with custom actions in C, new for this tutorial
3. Multicast packets in P4, new for this tutorial

Our technical requirements for this tutorial are that at least half the attendees possess laptops and wireless internet access at the conference site. We will create a cloud-based infrastructure similar to the one we recently created for P4 tutorial, shown below in Figure 4. Attendees at the tutorial will execute these labs using IDEs accessed from their laptops. (In our experience, the labs are usually done best by the attendees in groups of 2.) The lecture and lab portions of the tutorial will be delivered by experienced engineers from Netronome.



Lab Infrastructure

Lab infrastructure

- IDE VMs, log in to yours to write P4 code, 2-3 developers share an IDE VM
- Lock a server from the pool to download/run code (BIG RED CARD)
- Use the IP address, release the server when you're done

Working to make it available for hardware evaluation and R&D

Figure 4: Remote delivery lab infrastructure