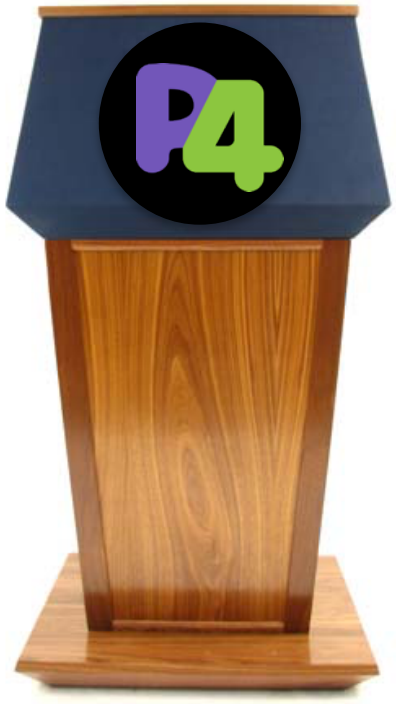
P4 Workshop Welcome

Nate Foster

Cornell University

State of P4



Industry Momentum

- Several P4-enabled targets now available
- New P4 products and deployments announced

Academic Interest

- P4 is increasingly being used as a platform for teaching and research
- Multiple P4 papers at SIGCOMM ‘17
- Tutorials at PLDI, SIGCOM, etc.

Open Source Community

- $P4_{16}$ released yesterday
- Growing collection of P4 software tools
- New working groups on Archiectures and APIs

P4.org Members

Original P4 Paper Authors:

BAREFOOT NETWORKS, Google, intel, Microsoft, PRINCETON UNIVERSITY, Stanford University

Operators/ End Users

Alibaba Group 阿里巴巴集团, Baidu 百度, FOX, Goldman Sachs, at&t, COMCAST, Google, kt, Microsoft, Tencent 腾讯, SK telecom

Systems

ARISTA, BROCADE, CISCO, DAHO Networks, CORSA, DELL, Hewlett Packard Enterprise, HUAWEI, Inventec, INTRACOM TELECOM, JUNIPER NETWORKS, mantis, NETBERG, NetBRIC, NoviFlow SDN made smarter, OpenxNet, ZTE, XENTECH

Targets

AEPONYX Moving the Cloud at the Speed of Light, Atomic Rules, BAREFOOT NETWORKS, CAVIUM, EZCHIP, intel, NETCOPE TECHNOLOGIES, Mellanox TECHNOLOGIES, BROADCOM connecting everything, centec networks, freescale, Innovium, MoSys, NETRONOME, PLUMgrid, vmware, MARVELL, STORDIS The High Performance Distributor, XILINX

Solutions/ Services

EstiNet, happiest minds, HGLOBAL HITECH, SDNLAB 专注网络创新技术, FLOW RESEARCH

Academia/ Research

Bii 天地互连, Cornell University Founded A.D. 1865, 國立交通大學 National Chiao Tung University, KOREA UNIVERSITY, PRINCETON UNIVERSITY, salzburgresearch, uni.lu UNIVERSITÉ DU LUXEMBOURG, Università della Svizzera Italiana, Eötvös Loránd University, POLYTECHNIQUE MONTRÉAL, POLITECNICO MILANO 1863, Stanford University, UFRGS UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL, VirginiaTech Invent the Future

- **Open source**, evolving, domain-specific language
- Permissive Apache license, code on GitHub today
- **Membership is free**: contributions are welcome
- Independent, set up as a California nonprofit

Today's Workshop

- **200+ attendees**
- **26 technical talks & demos**
- **29 organizations from academia and industry**

- AT&T Labs
- Barefoot Networks
- Bell Canada
- CESNET
- Cisco Systems
- Cornell University
- ETH Zürich
- Eötvös Lorláng University
- Flowmon Networks
- Google
- IIT Guwahati
- KAUST
- MIT
- Microsoft
- NIKSUN
- Netcope
- Netronome
- NYU
- Princeton University
- Stanford University
- Technion
- UC Berkeley
- Johns Hopkins
- UCDavis
- UIUC
- Universita della Svizzera italiana
- VMware Research
- Xflow Research
- Xilinx Labs

Language Evolution

The P4 Language Specification

Version 1.0.3

November 2, 2016

The P4 Language Consortium

1 Introduction

P4 is a declarative language for expressing how packets are processed by the pipeline of a network forwarding element such as a switch, NIC, router or network function appliance. It is based upon an abstract forwarding model consisting of a parser and a set of match+action table resources, divided between ingress and egress. The parser identifies the headers present in each incoming packet. Each match+action table performs a lookup on a subset of header fields and applies the actions corresponding to the first match within each table. Figure 1 shows this model.

P4 itself is protocol independent but allows for the expression of forwarding plane protocols. A P4 program specifies the following for each forwarding element.

- *Header definitions:* the format (the set of fields and their sizes) of each header within a packet.
- *Parse graph:* the permitted header sequences within packets.
- *Table definitions:* the type of lookup to perform, the input fields to use, the actions that may be applied, and the dimensions of each table.
- *Action definitions:* compound actions composed from a set of primitive actions.
- *Pipeline layout and control flow:* the layout of tables within the pipeline and the packet flow through the pipeline.

P4 addresses the configuration of a forwarding element. Once configured, tables may be populated and packet processing takes place. These post-configuration operations are referred to as "run time" in this document. This does not preclude updating a for-

$P4_{16}$ Language Specification

version 1.0.0

The P4 Language Consortium

2017-05-16

Abstract. P4 is a language for programming the data plane of network devices. This document provides a precise definition of the $P4_{16}$ language, which is the 2016 revision of the P4 language http://p4.org. The target audience for this document includes developers who want to write compilers, simulators, IDEs, and debuggers for P4 programs. This document may also be of interest to P4 programmers who are interested in understanding the syntax and semantics of the language at a deeper level.

Contents

1. Scope
2. Terms, definitions, and symbols
3. Overview
   - 3.1. Benefits of P4
   - 3.2. P4 language evolution: comparison to previous versions (P4 v1.0/v1.1)
4. Architecture Model
   - 4.1. Standard architectures
   - 4.2. Data plane interfaces
   - 4.3. Extern objects and functions
5. Example: A very simple switch
   - 5.1. Very Simple Switch Architecture
   - 5.2. Very Simple Switch Architecture Description
     - 5.2.1. Arbiter block
     - 5.2.2. Parser runtime block
     - 5.2.3. Demux block
     - 5.2.4. Available extern blocks
   - 5.3. A complete Very Simple Switch program
6. P4 language definition

$P4_{16}$ Retains Strengths of $P4_{14}$

- **Declarative programming model**
  Specify the "what" rather than the "how"
- **Familiar domain-specific features**
  Headers, parsers, controls, etc.
- **Higher-level constructs**
  Simplifies programs & facilitates code reuse

New Features in $P4_{16}$

Target-architecture separation

- *Portability across targets*
- *Extensibility through “externs”*

Static type system

- *Rich constructs for structuring data*
- *Primitive operations have well-defined semantics*

Higher-level programming constructs

- *Makes programs more succinct*
- *Encourages code reuse*

Open-Source Software

- **Compiler Framework (`p4c`)**
- **Software switch (`bmv2`)**
- **Tools**
  - Network emulation (Mininet)
  - Test framework (**`ptf`**)
  - Docker container (**`p4app`**)

`https://github.com/p4lang`

Common Misconceptions

Myth	Reality
P4 is controlled by a single vendor	P4 is being developed by an open consortium
P4 is immature	P4 is over 4 years old, and has deployed in production environments
P4 is based on a sequential model	P4 compilers can generate efficient code that takes advantage of parallelism
P4 is limited to simple functionality	$P4_{16}$ is extensible via externs
P4 requires a fully programmable target	$P4_{16}$ architectures allow smooth mixing of fixed-function and programmable components
P4 is match-action centric	$P4_{16}$ is based on standard imperative programming constructs

Commitment to Stability

Our responsibility is to ensure that you *never* dread upgrading to a new version of P4.

If your code compiles with $P4_{16}$ version 1.0.0 it should compile with $P4_{16}$ version 1.x.y with a minimum of hassle.

[Borrowed from Aaron Turon with permission]

Thank You

- **P4 LDWG Past Co-chair**
  - Changhoon Kim
- **P4 LDWG Members**
  - Mihai Budiu (VMware)
  - Calin Cascaval (Barefoot)
  - Chris Dodd (Barefoot)
  - Andy Fingerhut (Cisco)
  - Vladimir Gurevich (Barefoot)
  - Andy Keep (Cisco)
  - Edwin Peer (Netronome)
  - Ben Pfaff (VMware)
  - Cole Schlesinger (Barefoot)
- **Program Committee**
  - Calin Cascaval (Barefoot)
  - Gordon Brebner (Xilinx)
  - Ben Pfaff (Vmware)
  - Lorenzo Vicisano (Google)
  - Andy Keep (Cisco)
- **Local Arrangements**
  - Chloe Darsch
  - Chris Hartung
  - Prem Jonnalagadda
  - Melodie Shafazand