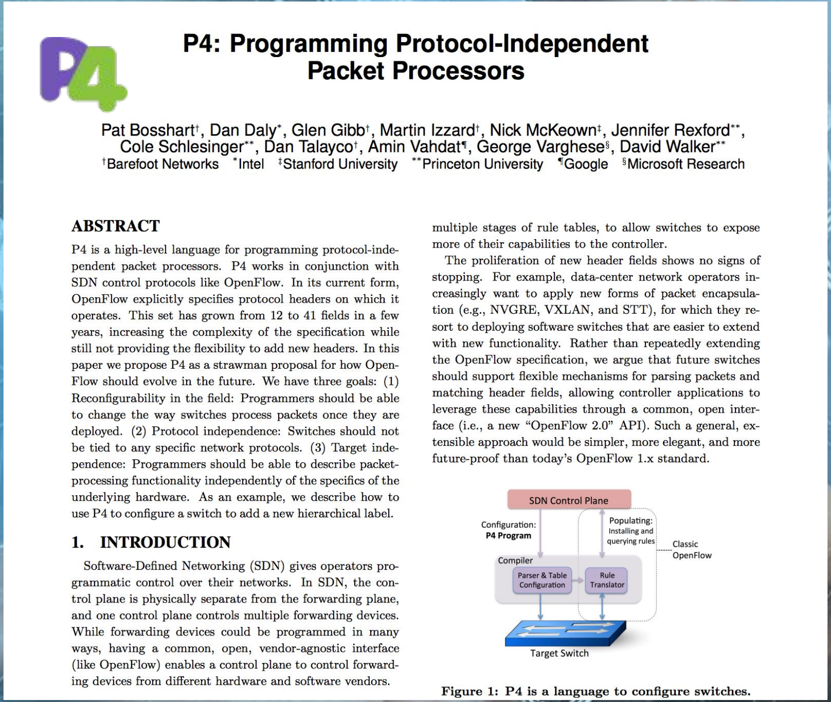
JUNIPER AND P4

Sandesh Kumar Sodhi

sksodhi@juniper.net

P4

P4 Paper



P4: Programming Protocol-Independent Packet Processors

Pat Bosshart[†], Dan Daly[*], Glen Gibb[†], Martin Izzard[†], Nick McKeown[‡], Jennifer Rexford[**], Cole Schlesinger[**], Dan Talayco[†], Amin Vahdat[¶], George Varghese[§], David Walker[**]
[†]Barefoot Networks [*]Intel [‡]Stanford University [**]Princeton University [¶]Google [§]Microsoft Research

ABSTRACT

P4 is a high-level language for programming protocol-independent packet processors. P4 works in conjunction with SDN control protocols like OpenFlow. In its current form, 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 while still not providing the flexibility to add new headers. In this paper we propose P4 as a strawman proposal for how OpenFlow should evolve in the future. We have three goals: (1) Reconfigurability in the field: Programmers should be able to change the way switches process packets once they are deployed. (2) Protocol independence: Switches should not be tied to any specific network protocols. (3) Target independence: Programmers should be able to describe packet-processing functionality independently of the specifics of the underlying hardware. As an example, we describe how to use P4 to configure a switch to add a new hierarchical label.

1. INTRODUCTION

Software-Defined Networking (SDN) gives operators programmatic control over their networks. In SDN, the control plane is physically separate from the forwarding plane, and one control plane controls multiple forwarding devices. While forwarding devices could be programmed in many ways, having a common, open, vendor-agnostic interface (like OpenFlow) enables a control plane to control forwarding devices from different hardware and software vendors.

multiple stages of rule tables, to allow switches to expose more of their capabilities to the controller.

The proliferation of new header fields shows no signs of stopping. For example, data-center network operators increasingly want to apply new forms of packet encapsulation (e.g., NVGRE, VXLAN, and STT), for which they resort to deploying software switches that are easier to extend with new functionality. Rather than repeatedly extending the OpenFlow specification, we argue that future switches should support flexible mechanisms for parsing packets and matching header fields, allowing controller applications to leverage these capabilities through a common, open interface (i.e., a new "OpenFlow 2.0" API). Such a general, extensible approach would be simpler, more elegant, and more future-proof than today's OpenFlow 1.x standard.

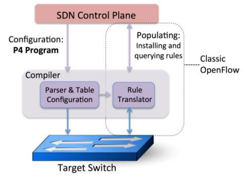

SDN Control Plane
Configuration: P4 Program
Populating: Installing and querying rules
Classic OpenFlow
Compiler
Parser & Table Configuration
Rule Translator
Target Switch

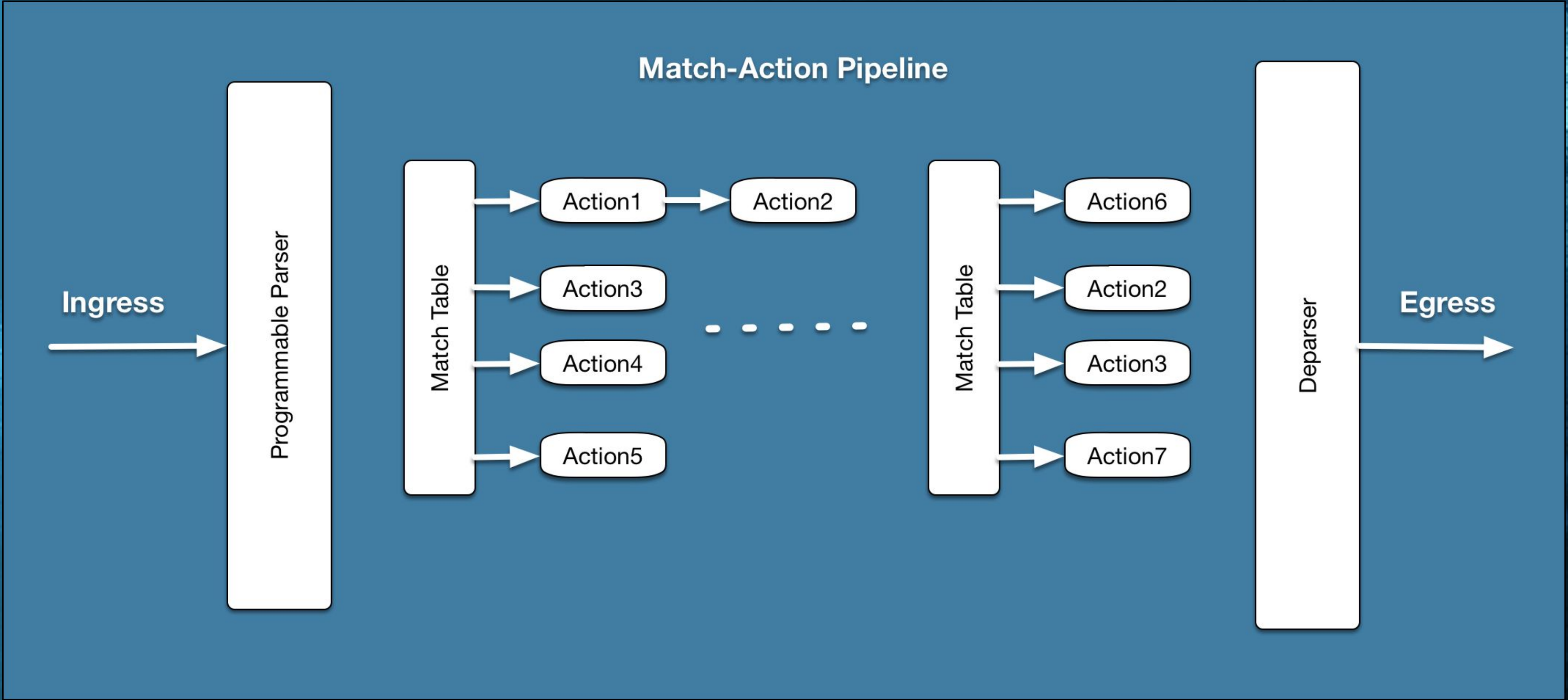
Figure 1: P4 is a language to configure switches.

What is P4?

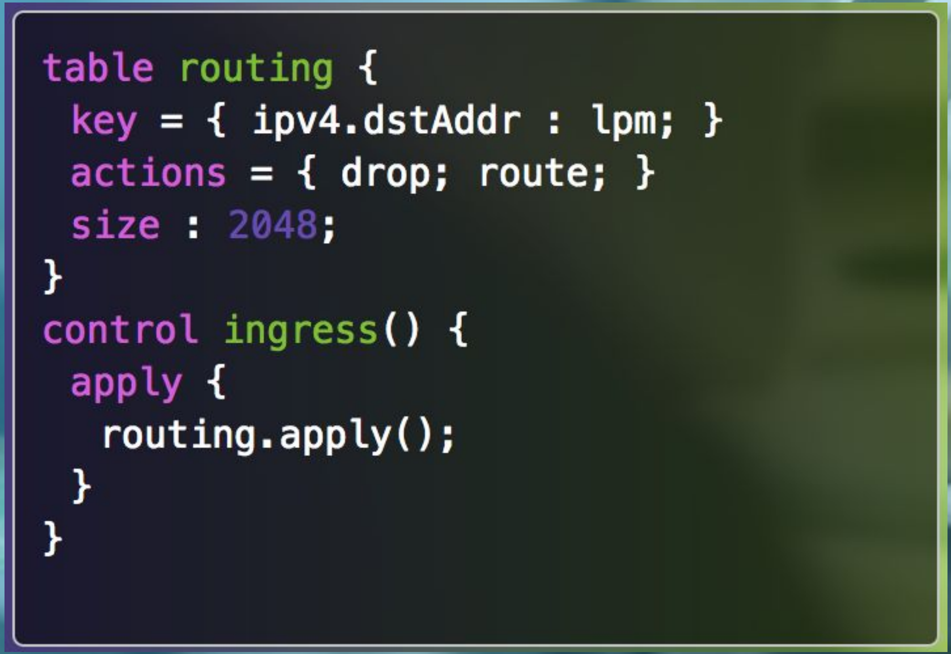
P4 is not a protocol or software.

P4 is a **LANGUAGE** for programming the data plane of network devices.

P4 View of Forwarding Plane

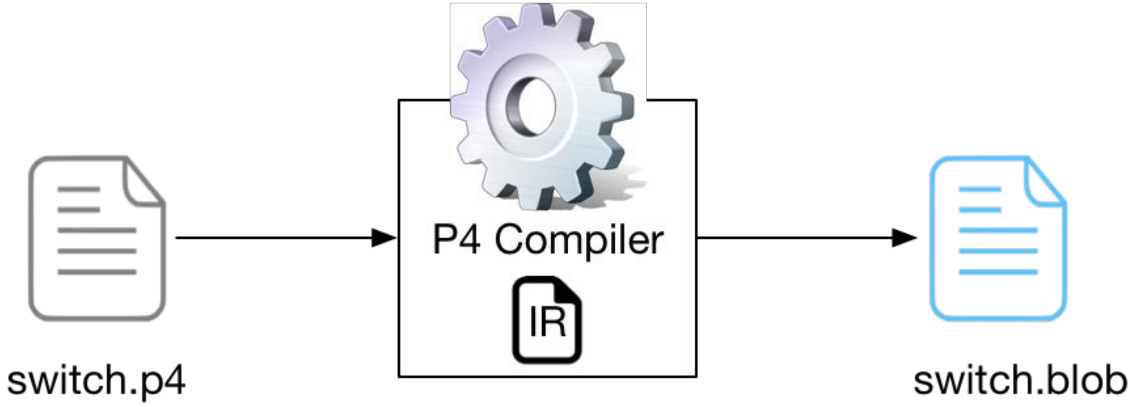


IPv4 Router: P4 Program Snippet

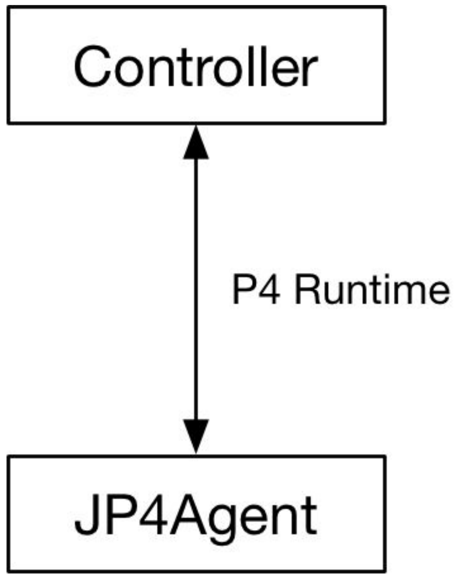


```
table routing {
 key = { ipv4.dstAddr : lpm; }
 actions = { drop; route; }
 size : 2048;
}
control ingress() {
 apply {
   routing.apply();
 }
}
```

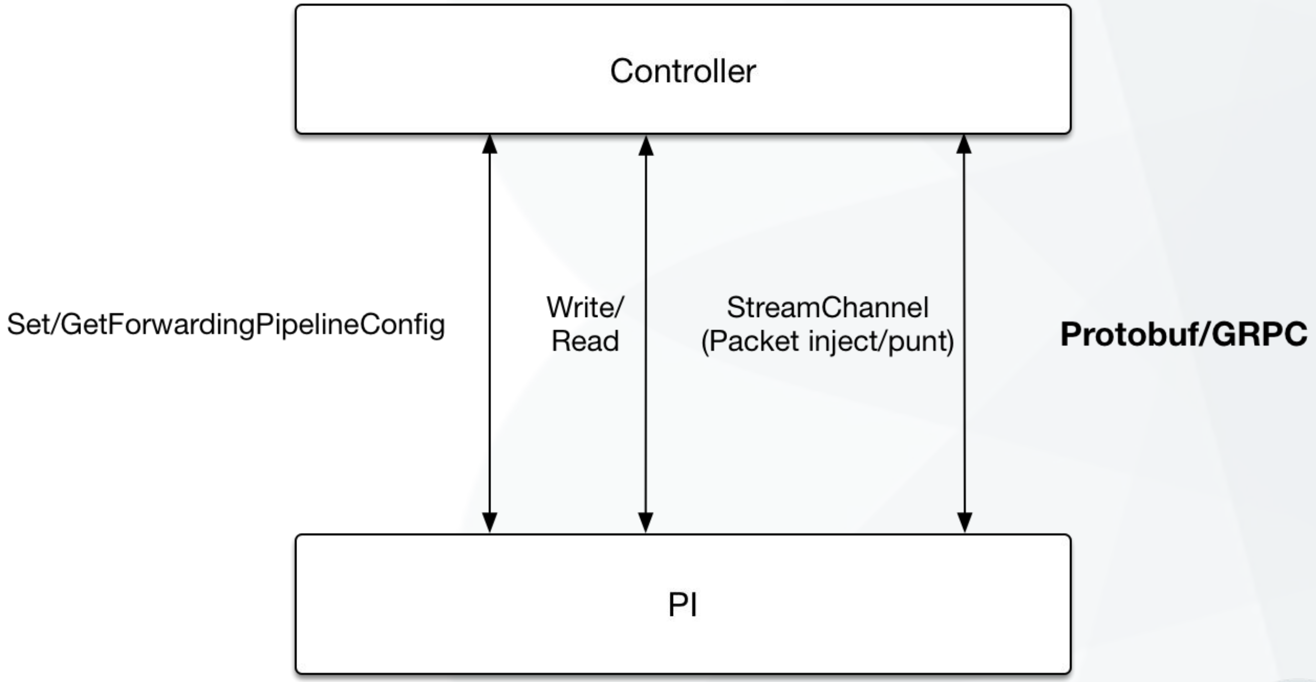
P4 Compiler



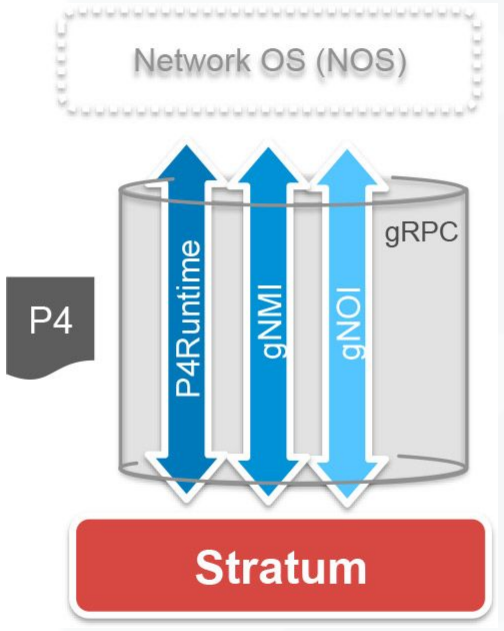
P4 Runtime



P4 Runtime: Deep Dive



The Stratum Project



P4 in Juniper

P4 and Juniper

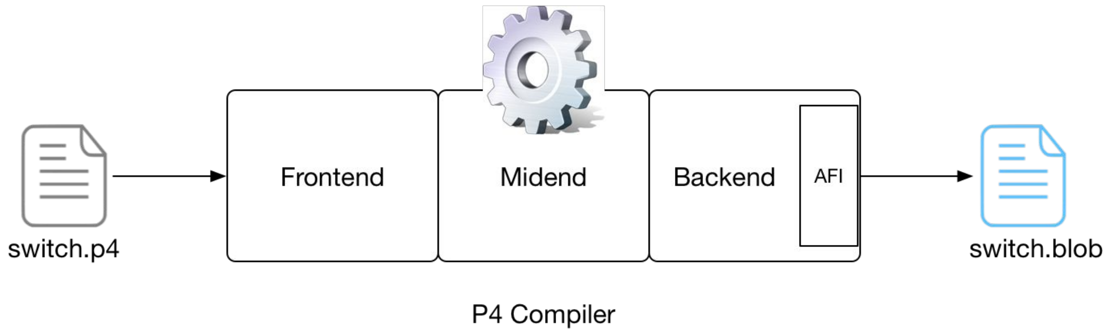
P4 is a great language to have conversations with our customers and partners. In P4, customers can articulate intent of feature which they want Juniper to implement in our forwarding plane. P4 compiler and JP4Agent translates that intent, written in P4, to data plane of Juniper's routing and switching platforms.

TRIO:
With TRIO, full potential of P4 can be realized.

Fixed Function ASICs:
P4 can be used to program our fixed pipeline ASICs based platforms (viz. QFX and PTX).

Merchant Silicon:
Seamless support of P4 on the platforms which Juniper builds using merchant silicon.

Juniper P4 Compiler



JP4C: Juniper P4 Compiler

A P4 compiler translates P4 programs to a representation (say a set of APIs or a program in target hardware specific language) which can be easily be consumed by target hardware.

A P4 compiler has three components: Frontend, Midend and Backend. P4 Consortium (p4.org) provides source code for frontend and midend and a reference implementation of backend. Every vendor needs to write compiler backend which translate the intermediate representation (IR) of p4 program, which is the output of compiler midend, to a representation which their target ASIC can consume/realize.

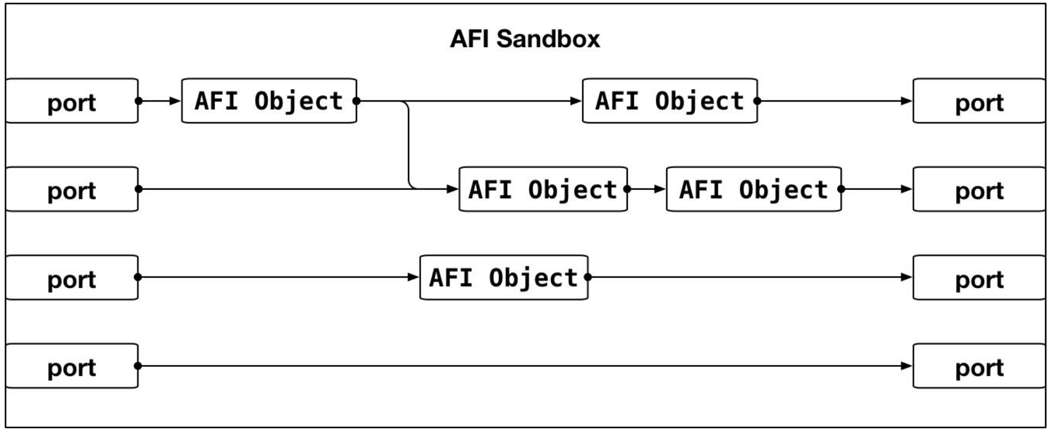
JP4C Contd...

The Juniper’s P4 compiler backend -

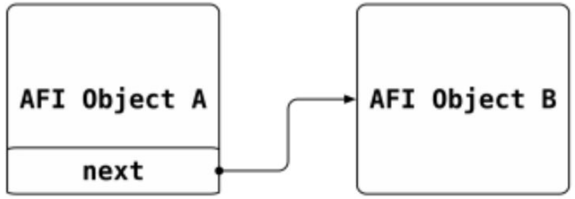
- **Validates/verifies** that the P4 program can be realized in a particular target (such as Broadcom TH3, ZT, ZH and ZX). It the program cannot be realized by the target then it fails the compilation by giving error as to why this program cannot be implemented in the target.
- **Translates** P4 programs to the the AFI data model. Output of P4 compiler is a file (switch.blob in above diagram) which holds the AFI data model representation of P4 program.

AFI Data Model

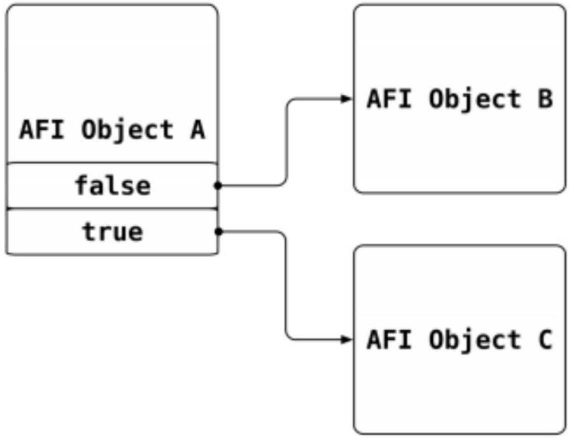
AFI (Advanced Forwarding Interface) is Juniper’s approach for defining data model for forwarding plane of networking devices. AFI data model defines data plane as forwarding topology graph of potential operations (AFIObjects) to be performed by packet forwarding engine (PFE) on packet.



AFI Data Model Contd...



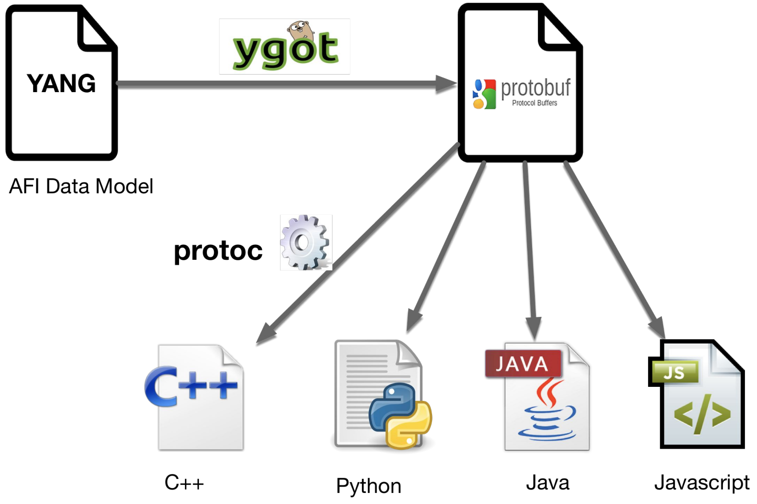
An AFI object can reference other AFI objects



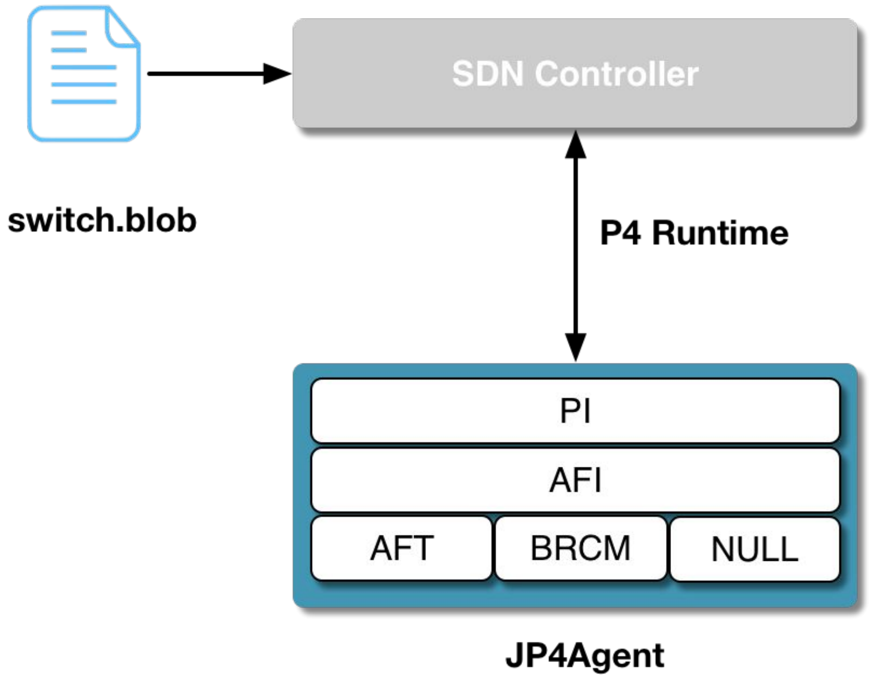
A conditional if-else AFI object

AFI Contd...

The AFI data model is defined in YANG data modeling language. Language bindings for all popular languages can be generated from this YANG model. It allows controller/application to be written any language of choice.



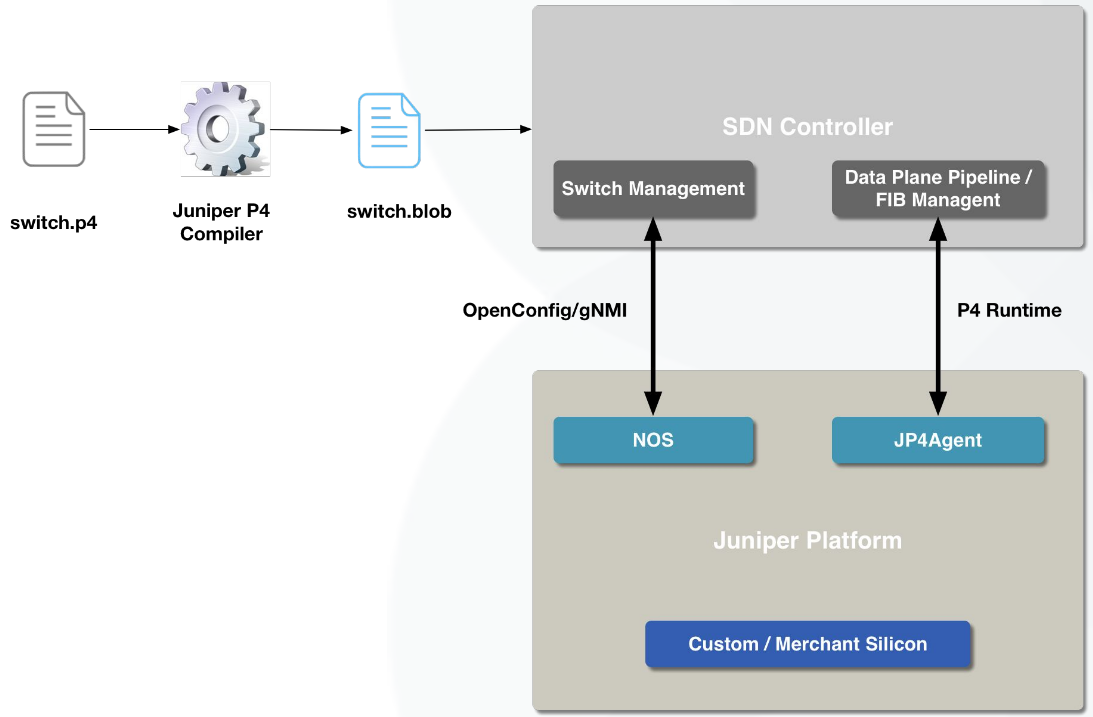
JP4Agent


SDN Controller
switch.blob
P4 Runtime
PI
AFI
AFT
BRCM
NULL
JP4Agent

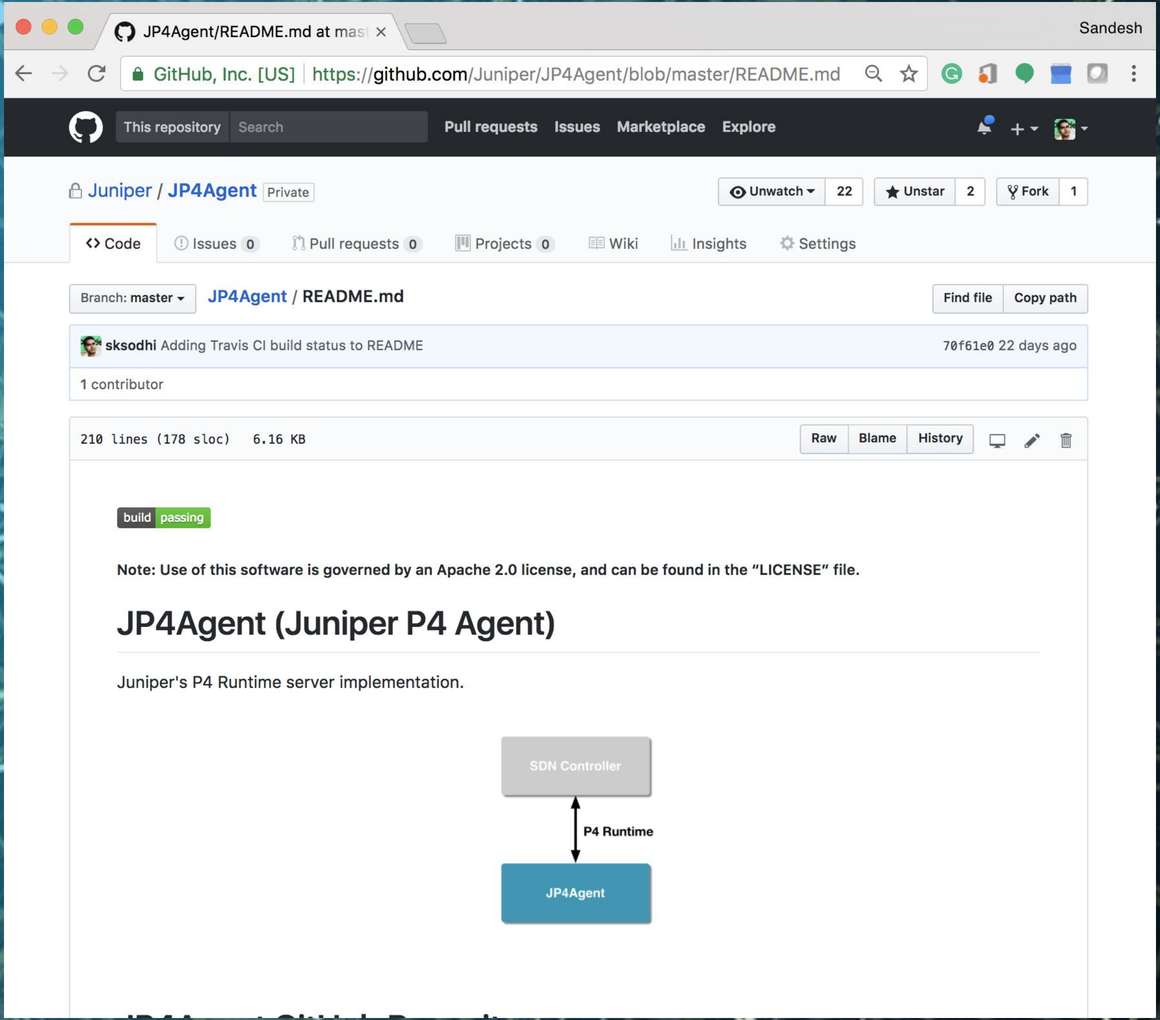
JP4Agent

PI layer hosts the P4 Runtime server which handles P4 Runtime gRPC messages. The AFI data model representation of P4 program is sent to JP4Agent by SDN controller (in SetForwardingPipelineConfig gRPC of P4 Runtime) . **AFI layer** interprets the AFI data model and sends the AFI objects to the layer below it, which is target specific. Multiple targets can be supported under AFI layer of JP4Agent. **AFT target** support all the platforms based on Juniper's custom silicon (ZH, ZT, ZX etc). BRCM (Broadcom) target is for Broadcom ASIC based platforms. **NULL target** is used for running unit test and regression on JP4Agent code.

P4 - Complete View



JP4Agent on GitHub.com



JP4Agent/README.md at mast ×

Sandesh

GitHub, Inc. [US] https://github.com/Juniper/JP4Agent/blob/master/README.md

Juniper / JP4Agent Private

JP4Agent / README.md

sksodhi Adding Travis CI build status to README 70f61e0 22 days ago

1 contributor

210 lines (178 sloc) 6.16 KB

Note: Use of this software is governed by an Apache 2.0 license, and can be found in the "LICENSE" file.

JP4Agent (Juniper P4 Agent)

Juniper's P4 Runtime server implementation.

Juniper and P4 Community

P4.org: Language Consortium

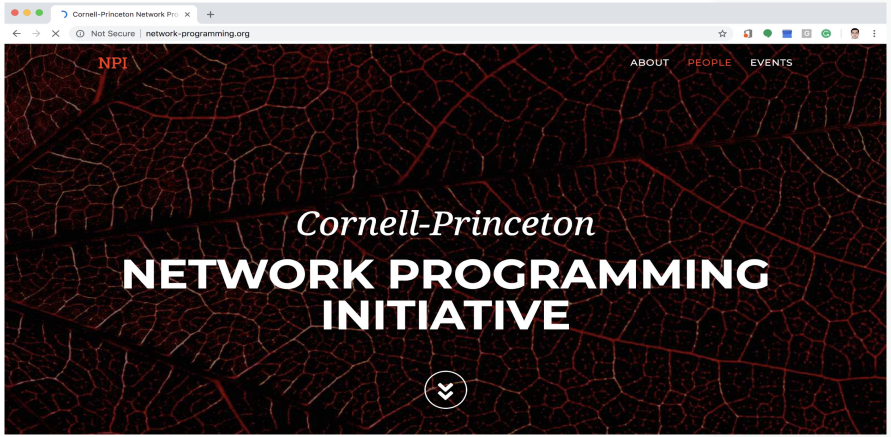
API Working Group: P4 Runtime

Architecture Working Group

Language Design Working Group

Application Working Group

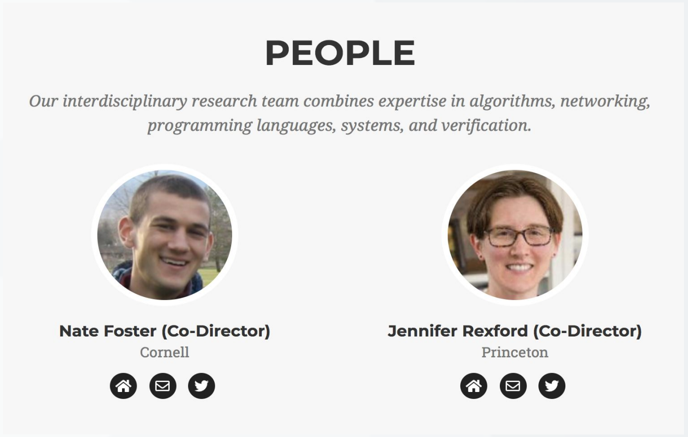
Network Programming Initiative



Cornell-Princeton Network Pro
Not Secure | network-programming.org
NPI
ABOUT
PEOPLE
EVENTS

Cornell-Princeton

NETWORK PROGRAMMING INITIATIVE



PEOPLE

Our interdisciplinary research team combines expertise in algorithms, networking, programming languages, systems, and verification.

Nate Foster (Co-Director)
Cornell

Jennifer Rexford (Co-Director)
Princeton

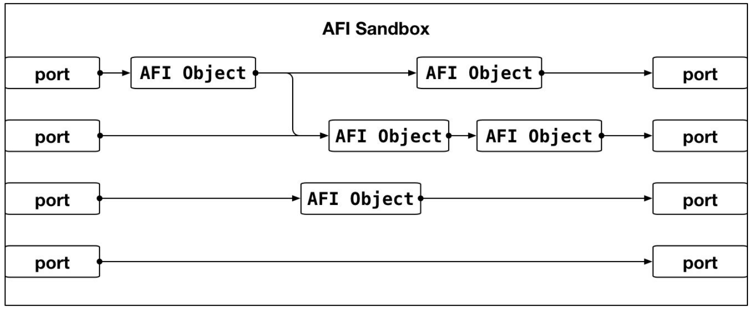
http://network-programming.org

Incremental Insertion of P4

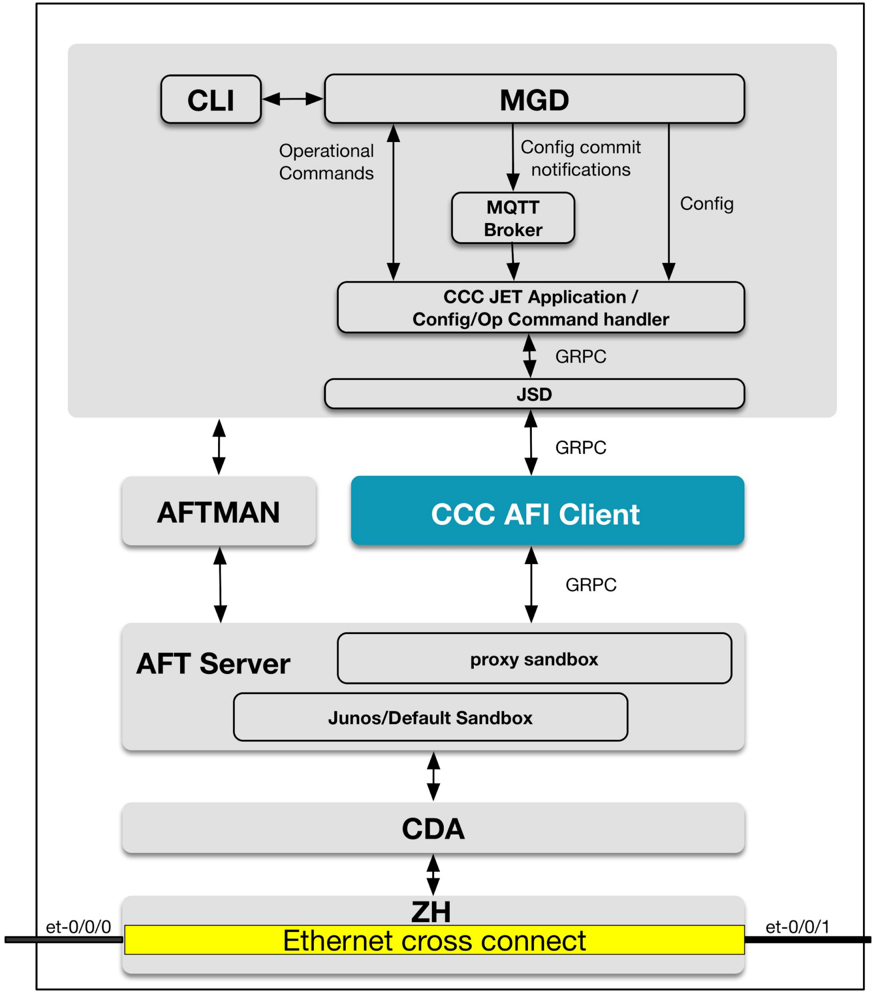
- ❏ Existing data plane code of various networking vendors has value which has been created over a long period of time. Defining (re-writing) the complete data plane in P4, which would involve compiler backend changes and changing the existing local control planes’ south-bound interface to P4Runtime) for all the features, which are currently supported by vendors’ routers and switches, would require significant development and testing efforts.
- ❏ Customers may not want to assume complete ownership of control plane. They may not either be equipped (read resource constraints) or do not want to handle/rewrite P4 enabled control plane (even if vendors are willing to provide P4 programmable data plane) for features which are already available from vendors’ existing field-proven control-plane and data-plane code.

AFI Sandbox

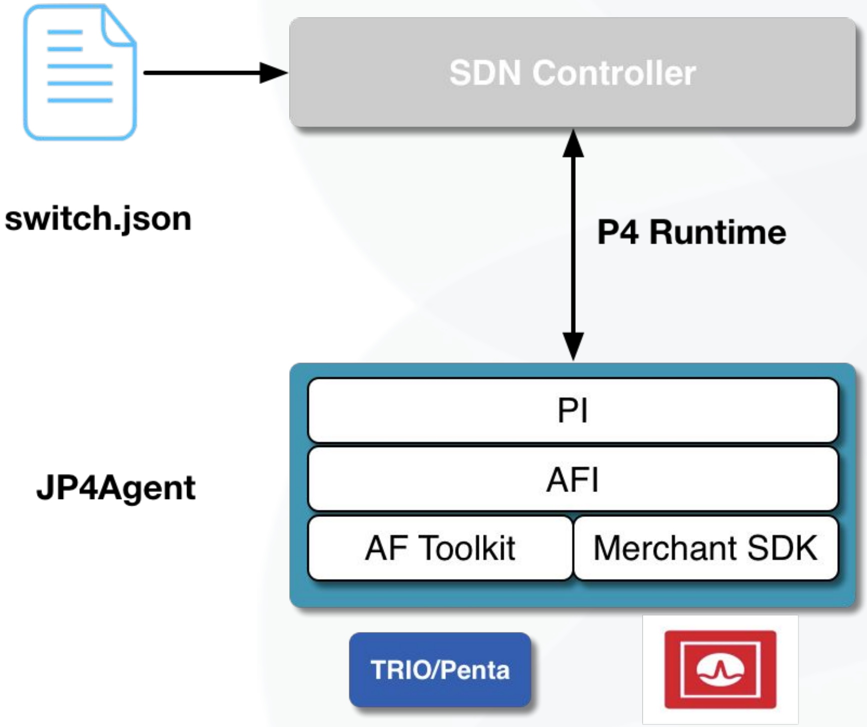
AFI data model defines data plane as forwarding topology graph of potential operations (**AFI Objects**). AFI defines a **sandbox** as a section of forwarding topology graph. A sandbox is small virtual container which can be programmed via AFI. The construct of sandbox allows for different parts of forwarding topology being controlled by different control plane applications. In a typical scenario, a small section of forwarding topology can be managed by external SDN controller, while rest of the forwarding topology being managed by JUNOS.



Circuit Cross Connect (CCC)



P4 in Juniper...



THANKS