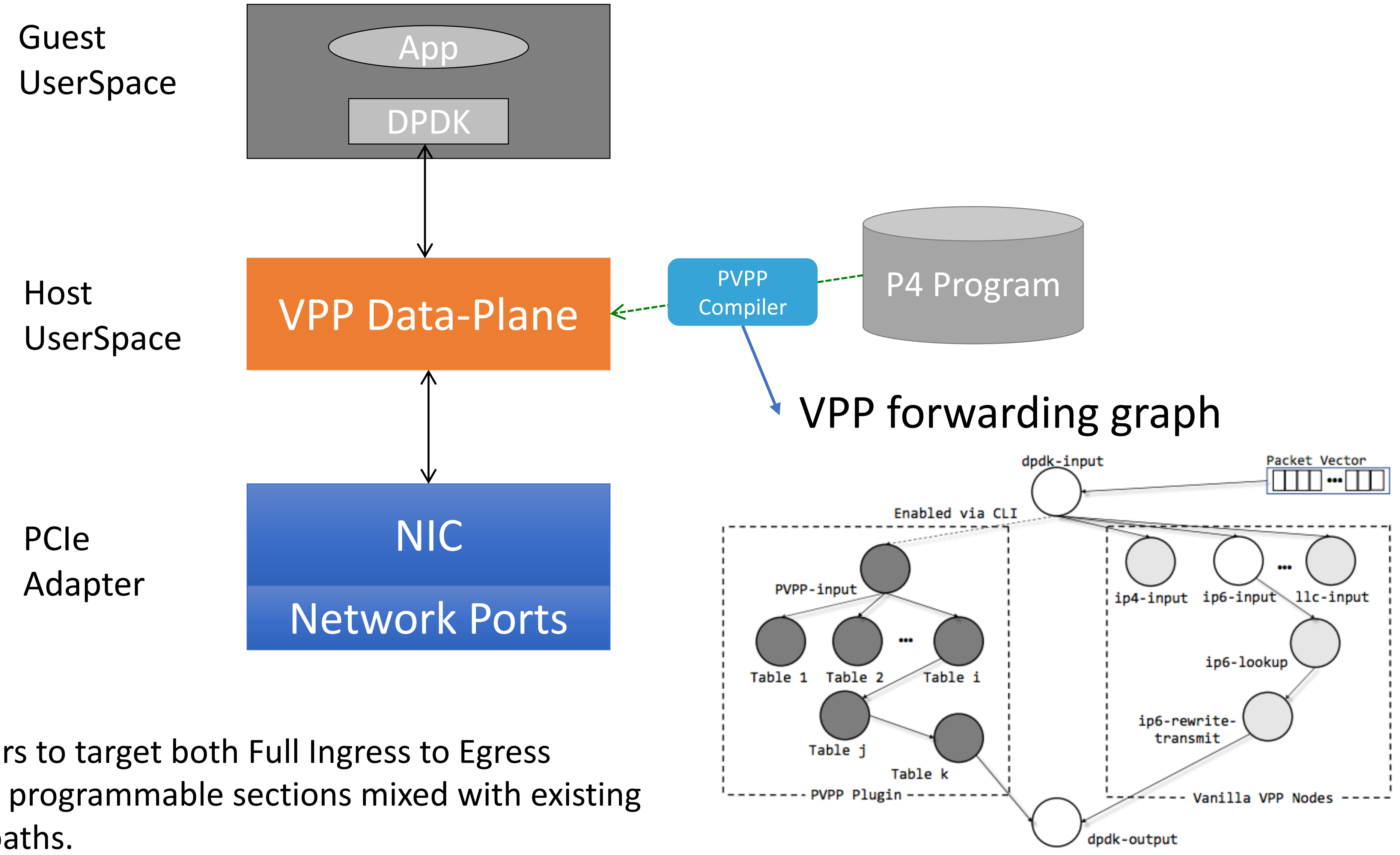
P4VPP FD.io Project Kickoff

August 24, 2017

P4VPP Project Overview

- P4 is a domain specific language that allows developers to write packet processing applications for a variety of architectures, including Switching ASICs, Network Processors and CPUs. The P4VPP project will enable these capabilities to be mapped onto the FD.io VPP dataplane software platform.
- The P4VPP project will build a P4 toolchain that maps the user supplied P4 program onto a FD.io VPP software platform, thereby minimizing the effort to create new features, and moving the complexity of achieving performance to the P4 compiler.
- The goal of this project will be to create a functional framework first and foremost, and enable ongoing development to further optimize the performance results from the P4C VPP compiler backend.

P4VPP – P4 to VPP Compiler (vSwitch Example)



Enable P4/VPP developers to target both Full Ingress to Egress processing, as well as P4 programmable sections mixed with existing VPP nodes and feature paths.

Project Scope

The scope of the P4VPP project will be the creation of software tools that are necessary to enable a P4 program to compile and run on the FD.io VPP software platform.

Minimally will include the target specific extensions to the p4.org P4C compiler to target the VPP platform, both in terms of the data-plane and the control plane interfaces.

The packaging of the compiler artifacts will target a VPP plug-in model, allowing for out-of-tree development, packaging and release.

Contributions will target the following capabilities:

- P4C Compiler Backend that targets the VPP Software Platform, including both the data-plane packet manipulations, as well as the control plane interface into table/action population.
- Generalized VPP libraries that can be used by the P4 compiler. These libraries will minimally include P4 table lookups (exact, ternary, range, and lpm) and P4 stateful memories such as counters and registers.
- Low-level Performance instrumentation that can be used to measure and validate the results of compiler optimizations.
- Compiler generation for API definition and backend implementation for all configurable parts of the P4 program, specifically match tables and action parameters
- Toolchain generation of VPP specific build environment and plugin instrumentation that allows P4 programs to be built out-of-tree, and dynamically loaded into the VPP execution context.
- P4 Program Fragments that can be used to test the P4VPP backend functional correctness.
- P4 Programs and associated test harnesses that represent the VPP CSIT functional and performance test cases. The goal being to establish functional equivalency and performance baselines that can be used to compare the correctness of the compiler as well as the performance results.

P4VPP – Work to Date

- Started as a summer PhD intern researh project in June 2016 co-sponsored by Cisco and Barefoot Networks
  - Nick McKeown (Stanford/Barefoot) expressed interest and volunteered Sean Choi who was interning at Barefoot in Summer/16
  - Xiang Long, PhD student at Cornell also was interested in this area
  - Ultimately pulled in additional help on the paper writeup from Muhammad Shahbaz (Princeton) who co-authored the PISCES work and helped with the results/paper editing.
- Summer/Fall16
  - p4.org was going through two disruptive changes: P4_16 was being defined and the language was going through a fair amount of churn. Original Python based compiler for P4_14 was being completely rewritten in C++ and is was transitioning to what is now P4C. The compiler was not deemed stable enough at the time to natively integrate into mid/back end passes.
  - Resulting decision was to hook into the JSON backend the P4C compiler generated for the BMv2 simulation environment as this was deemed very stable as it was used for both P4_14 and P4_16 simulation.
  - The initial backend leverage Python COG templates as a mechanism to generate VPP C-code from the JSON IR the P4c Compiler used
  - Work was published as a Poster to SOSR17* and as a short paper to APNET17**
- Summer/17 – Restarted work in this area with a goal to move the contributions into the FD.io project
  - Cisco has 2 resources actively working on getting a functional prototype in place and ready to contribute that started point to the community as soon as the project is approved and GIT repos are created
- Initiator set of Committers:
  - Andy Keep LF-ID: akeep
  - James Coole LF-ID: jamescoole
  - Cian Ferriter LF-ID:CianFerriter

* https://sosr17demos.hotcrp.com/doc/sosr17demos-final19.pdf

** http://stanford.edu/~yo2seol/static/pvpp-apnet.pdf