P4 to VHDL

P4

P4 is new language used for describing packet processing. Its main advantages are that the language itself is completely protocol and target independent. Therefore it gives the network administrator the power to describe packet processing in a unified way, using decent levels of abstraction, without needing to care about the underlying infrastructure of different switches. For more info, specification and examples of P4 visit: http://p4.org/

What does *p4vhdl* do?

The framework can automatically generate VHDL source code from P4 sources. So user can easily describe how he wants to process packets in P4 language and then use this framework to get VHDL implementation of such processing.

Packet processing works in 3 main stages (as specified by P4). First of all packets (packet headers) are parsed based on described protocols. Obtained packet header fields are then used to match packets against sets of rules in match+action tables. Every rule has some kind of action (drop, permit, add header, modify header...) associated with it. This action is applied on each packet that matches said rule. In the end the packet with possibly modified headers is put back together.

Parser

- any protocol that user defines in P4
- header parsing (parsed out header fields can be used in later stages)
- keyword **current**
- everything after last parsed header is considered to be payload

Match+Action

- modification of header fields
- increase/decrease of values in header fields (for example TTL)
- adding/removing of entire headers
- **if-else** conditions
- keywords **hit**/**miss** (can be used within conditions to apply some table only if the last table successfully matched)
- keyword **valid**
- runtime population of M+A tables via memory interface

Deparser

- properly puts together packets

NETCOPE TECHNOLOGIES a.s., U Vodarny 2965/2, Brno, 616 00, CZ
www.netcope.com / info@netcope.com / +420 511 205 264

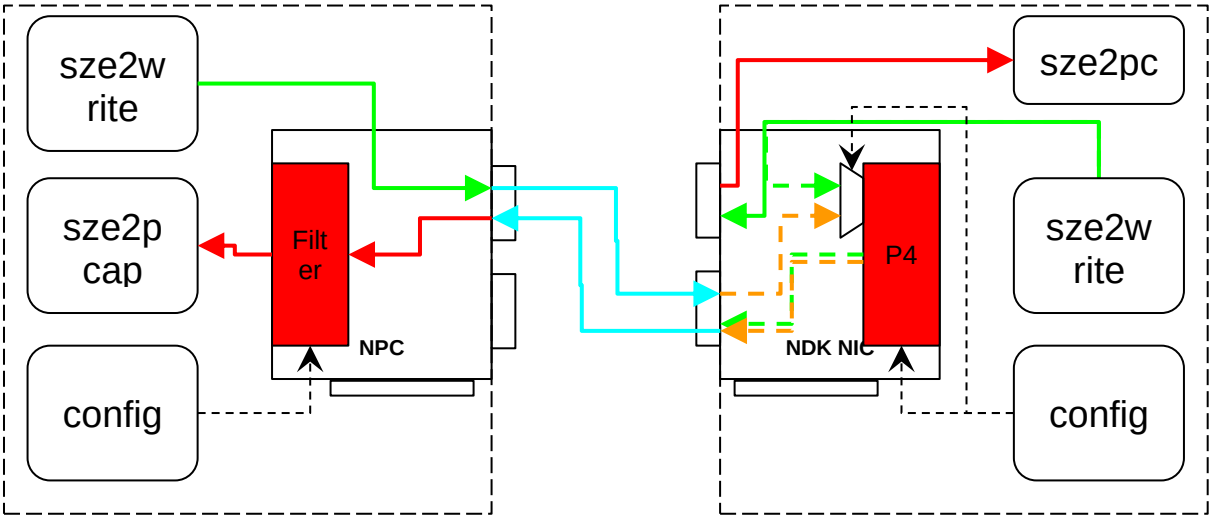
Demo application

Goal

Application shows P4's capabilities to describe packet processing and also demonstrates *p4vhdl* framework. Application is a use case that shows the ability to generate packet filter, which can also tag specific packets with VLANs, from P4 description.

This can be further extended to let anyone describe simple packet processing in P4, which we can then, within hours, turn into functional firmware.

Setup



Generated VHDL component is used within NDK (Netcope Development Kit) on NFB-100G2 card (which sits on one side of communication). There are two modes of operation available. In first mode packets are send into card from SW (either prepared PCAPs or randomly generated packets). These packets are unaltered send out through card's port 0. Additionaly the packets are send to the P4 generated component, which adds VLAN header with specific VLAN number onto some of them. These packets are then send out through port 1. Second mode is very similar, but the input packets are taken from port 1 instead of from SW. Both P4 component and switching between modes is configurable from SW.

On the other end of the communication there is another NFB card, this time with Netcope Packet Capture firmware (NPC). This card however serves only as a simple filter that only filters out all the packets that have not been tagged. Remaining packets are then send into SW and showed via Wireshark to illustrate that they indeed have been tagged with VLAN.

This scenario demonstrates that both traffic from software and traffic from network can be processed using P4 generated pipeline to e.g. create overlay networks for a specific subset of traffic or even create distributed virtual switch. As the P4 is a programming language, the limits are almost endless.