Modeling P4 programmable devices using YANG

Daniel Lazkani Feferman, Christian Esteve Rothenberg
Departamento de Engenharia de Computação e Automação Industrial (DCA)
Faculdade de Engenharia Elétrica e de Computação (FEEC)
Universidade Estadual de Campinas (Unicamp)
Caixa Postal 6101, 13083-970 – Campinas, SP, Brasil
{fefer, chesteve}@dca.fee.unicamp.br

Abstract – Software Defined Networking express the logical centralization of programmability over a set of network functions. In order to define the proper abstractions to realize packet forwarding abstractions in data plane elements, proper methods to define network functions pipeline programs are being defined in the literature. In this paper we take our first step on the goal to integrate a layer 2 switch using P4 language with an open source network controller. We present an initial YANG data model to describe the specifications of a P4 switch, an initial approach towards methods of programming data plane forwarding abstractions.

Keywords – SDN, P4 language, YANG, OpenDaylight controller

1. Introduction
By introducing Software Defined Networks (SDN) [1], the control plane has been separated from the data/forwarding plane, making a huge change on networks architectures and centralizing the controller on the network. The Programming Protocol-independent Packet Processor (P4) [2] is a high-level language that gives the capability to program the data plane. The P4 language has three main goals: Reconfigurability, Protocol independence and Target independence. However, the support to P4 is still considerably small, the number of available switches that support the language is scarce and the number of controllers that support it is even smaller.

This project aims to initiate the process of integration between the controller with a switch that supports P4 language. We will model a layer 2 switch using the YANG data-modeling language [3] to achieve our goal to integrate both of them. In this paper, we use as reference the l2_switch on the P4 language GitHub [4] and a controller that supports YANG, on our case we propose OpenDaylight (ODL) [5].

The remainder of the paper is structured as follows. In section II, we give a background of what is going to be used, in section III we give an overview of the architecture we propose, in section IV we describe the YANG data model of the P4 program and section V discuss future works and the conclusion of this paper.

2. Background
In this section we will present a brief overview of what will be used on our architecture, we give a small introduction of the SDN controller, P4 language, Honeycomb agent and the YANG data-modeling language.

2.1. SDN controller
In 2008, UC Berkeley and Stanford University proposed to decouple the network control from the packet forwarding, enabling the control plane to be easily programmed and allowing the network intelligence to be centralized in SDN controllers [1] [5]. Therefore, this revolution on the network architecture led to the development of multiple controllers: Beacon [6], Floodlight [7], NOX [8], POX [9], Ryu [10], ONOS [11] and ultimately OpenDaylight [5].

The SDN controller can be compared as the brain of the network. It acts as the strategic control point in the network, manage flow control to switches and routers to deploy intelligent networks. Thus, the controller is similar to the core of the network. It lies between network devices at one end and applications at the other end. Any communications between applications and devices need to pass through the controller. Another feature of the controller is to take the control plane off the network hardware and run it as a software instead, automating network management [1], [5].

2.2. P4
Considering the development of OpenFlow (OF) protocol [12] over the years, few limitations were found (e.g., most switches have multiple policies and stages of match-action tables, limited TCAM space, etc). Although some of them were possible to be implemented using OpenFlow, over the years it was necessary to update OF versions adding new
headers and since there was a necessity of retro-compatibility, the only solution was to increase the header of each version. In 2013, the last version was released (version 1.4) containing 41 fields of header. Table 1 shows the number of header of each version of OF [2] over time.

These limitations led to the necessity of an “OpenFlow 2.0” that should have some features like [2]:

- The packet parser should be configurable and not tied to a specific header format;
- The match+action table should be able to match on all defined field and support multiple tables;
- It should support header fields and metadata packet-processing primitives like copy, add, remove, modify;

In 2014, the first paper about the P4 language was published. The article was academically led by Princeton and Stanford Univ. with the collaboration of Google, Intel, Microsoft and a startup named Barefoot Networks [2].

The P4 language establish three main goals [2]:

- Protocol independence: The language should allow to configure the way packets will be parsed and define a set of match+action tables;
- Target independence: It should be programmable without the knowledge of switch details, relying just on the compiler to configure the target switch;
- Reconfigurability: The change of parsing and processing in the field should be easily allowed;

2.2.1. P4 concepts

A P4 program is composed of five basic components [2]:

- Tables: Mechanism to make the packet processing. Inside the table there are the matches and actions to be executed;
- Actions: P4 allows the construction of actions using simple protocol-independent primitives;
- Parser: Analyze the packet headers and sequences of the packet;
- Control programs: Defines the order of match+action applied to a packet;
- Headers: Specifies fields widths and values;

2.3. Honeycomb

We are currently using Honeycomb as an agent that is mainly used to connect Vector Packet Processing (VPP), a system developed by cisco [13] with ODL controller. In our proposal, we use Honeycomb to integrate a P4 switch with ODL, this should be accomplished using YANG data-modeling language, which is already supported by ODL [5] and Honeycomb [14].

Honeycomb is a generic NETCONF/RESTCONF java-based management agent and it already supports OpenDaylight as Northbound using NETCONF/YANG [14].

2.4. YANG data-model

YANG (RFC 6020) is a data modeling language developed by the Internet Engineering Task Force (IETF) to model NETCONF/RESTCONF protocol. The language uses modules and submodules to make a model, which can import data from other modules/submodules. Another useful tool of YANG is the translation into an XML equivalent named as YANG Independent Notation (YIN). Every code in YANG can be converted into YIN and every YIN can be converted into YANG, so the process of conversion is called lossless [3].

The YANG language can be used to describe any data model, but it is mainly designed to describe networking data models. YANG Data sent to or from a network device will be formatted in XML or JSON, depending if it is going to be used NETCONF or RESTCONF. YANG has a high level of structure, the main components are [3] and [14]:

- Module: Contains three types of statements: module-header, definition and revision statements. The first one gives informations about the module and describe it. The second one is the body where data model is defined and the last one shows when was the last review of it. Every data model is a module that is a top-level hierarchy of nodes;
- Type: Defines an integer, a string, etc;
Table 1. The number of headers increasing over time [2].

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Header fields</th>
</tr>
</thead>
<tbody>
<tr>
<td>OF 1.0</td>
<td>Dec 2009</td>
<td>12 fields (Ethernet, TCP/IPv4)</td>
</tr>
<tr>
<td>OF 1.1</td>
<td>Feb 2011</td>
<td>15 fields (MPLS, inter-table metadata)</td>
</tr>
<tr>
<td>OF 1.2</td>
<td>Dec 2011</td>
<td>36 fields (ARP, ICMP, IPv6, etc.)</td>
</tr>
<tr>
<td>OF 1.3</td>
<td>Jun 2012</td>
<td>40 fields</td>
</tr>
<tr>
<td>OF 1.4</td>
<td>Oct 2013</td>
<td>41 fields</td>
</tr>
</tbody>
</table>

- **Containers**: Group related nodes in a sub-tree, it has no value and only child nodes, like leaves, lists, leaf-lists and containers;
- **Leaf**: Has a simple type as an integer or string and no child;
- **List**: Uniquely identified by the values of key leaves, it can define multiple key leaves and it can contain any number of child nodes;

2.5. Pyang

Pyang [15] is a validator, translator and code generator of YANG models, this program is written in Python with features like:

1. YANG modules validator;
2. Compact tree generation to represent YANG models for quick visualization;
3. Conversion of YANG to YIN and YIN to YANG;
4. Translation of YANG data models to XSD;
5. Schema-aware translation of instance documents encoded in XML to JSON and vice-versa;

The most valuable features used in this article is the ability to validate YANG modules and the tree representation of it. In our case we will validate our model using IETF standard. However, it is possible to validate according to canonical too. Another great feature of Pyang is the validation based on an RFC, they can give a feedback of what line of the YANG code is wrong/missing and it links to the part of the RFC that the code need to comply.

The IETF standard for YANG data modeling has another interesting feature, to comply with the standard the developer must declare a description of each module, list and node of the code. This is mainly useful to another developer that need to implement or understand the model.

3. Architecture overview

In this section, we take a top-to-bottom approach explaining the design of our architecture. Theoretically any P4 program, compiler and controller can be used. However, to achieve this we would need to adapt them. Thus, we are working with l2_switch [4] as our P4 program, compiled using the MACSAD [16] as the compiler, and OpenDaylight (ODL) [5] as the controller, considering that it already supports YANG [3]. Therefore, to intermediate and integrate the P4 switch with ODL we will use YANG/NETCONF and Honeycomb agent to transport this model. In Figure 1, we demonstrate the proposed architecture.

4. Layer 2 YANG model

The objective of this YANG model is to send to ODL informations about the switch that are useful and necessary to decide the actions to be made. Some informations described on our YANG model are: which actions that can be done, name and id of each table and the number of ports the switch has.

YANG hierarchically model the data as a tree, defining modules and submodules and it child nodes. Figure 2 shows the tree of our YANG code.

5. Conclusion and Future Work

In this article we presented a brief description and history of SDN controllers, we made a description...
about the P4 language, a powerful and promising language to program and change the way switches work and we introduced the honeycomb agent, which will be the agent used to transport our YANG model. Then, we made our first approach to integrate a P4 program and an open source controller. We used YANG to describe the specifications of the P4 program.

As a future work we need to implement the way this model will be read and used by the controller and the switch, using NETCONF/RESTCONF protocol.

ACKNOWLEDGMENT

This work was funded by Ericsson Telecomunicações.

References