Improving current P4 prototyping tools

Semester thesis proposal

In the last years, there has been a big paradigm shift in the networking industry with the rise of Software-defined networking. Software-defined networking enables network operators to program devices from the top (control plane) to the bottom (data plane). Among others, this programmability enables extremely fast innovation and allows network operators and devs to have a tighter control of what their networks can do. With the apparition of P4 [8], a domain-specific programming language that allows to program network data planes, data-plane programmability has become accessible to all, making it a hot topic in the networking community (research and industry). The P4 language is designed to be target-independent, which means that can be compiled to a wide range of architectures (NPUs, CPUs, FPGAs) and the world's fastest Ethernet switching ASIC (12.8 Tbps) [1]. P4 is also protocol-independent, meaning that target devices are not tied to any specific network protocol.

To be able to easily prototype and test data plane applications without having to buy a physical device, P4 software switches [4] have been developed. These software switches run as normal process in a CPU and are able to listen and send packets from virtual interfaces. Together with already existing virtual networks orchestrators, such as mininet [3] we can test and debug the behavior of our data plane applications.

Existing tools such as p4-utils and p4app [6,7] allow for easy P4 prototyping using the mininet [3] network virtualization framework. p4-utils has been developed at our lab and is currently used in our lecture Advanced Topics In Communication Networks as well as in one of the most used P4 tutorials [5] out there.

In this thesis, we aim on improving p4-utils such that it can be more easily used, better integrated with mininet, and with newer and up to date features.

Some of the main objectives for this thesis (list can vary during the project) are listed below:

Objectives

- Understanding main building blocks of mininet (node, link, mininet, interfaces, etc). Also understand the underlying technology (network namespaces, veth, tc, etc). Mininet core code.
- Understanding how to add new features and node types.
- Understanding the main building blocks of P4-utils (p4app, topologies, Thrift controller).
- Investigate all the additions to the main repositories during the last year. Specifically: bmv2, p4c, and P4runtime.
- Migrate p4-utils to python3.
- Update p4-learning such that all examples, and exercises work with python3.
- Study P4Runtime.
- Rewrite the p4-utils controller code such that it can be used with Thrift and P4Runtime at the same time.
- Add P4Runtime to p4-learning exercises and examples.
- Integrate p4-utils with Mininet better such that not only json files are used to define the topology. The goal should we that we can create topologies in a more pythonic way, while keeping the old way for simplicity.
- Add an easy way to schedule processes in nodes. We would have a conf file with processes, start time, end time and it should be scheduled to all hosts.
- Add a native traffic generator. One option could be to integrate minigenerator [2].
- Improve and remove some topology limitations we have right now. For example it would be nice to add the following supports: host-to-host connectivity, host-to-multiple-switches, multiple links between switches.
- Add new nodes: OVS, LinuxBridge, FRR router.

Requirements
- Having attended the Advanced topics in communication networks is required.
- Knowledge of the P4 language.
- Knowledge of linux network virtualization tools is recommended.
- Intermediate/Advanced Knowledge of Python is required.

Contact
- Edgar Costa Molero, cedgar@ethz.ch
- Prof. Laurent Vanbever, lvanbever@ethz.ch

References