NOSIX: A Portable Switch Interface for the Network Operating System

Andreas Wundsam (ICSI)

Software-Defined Networking (SDN), currently successful in both research [1, 6] and industry [3, 2], gives operators the opportunity to program the control plane of their networks, with the ubiquitous OpenFlow network API [9] serving as the linking element of the ecosystem. Using an event-driven message passing paradigm, it allows a controller to control switch behavior through flow-based rules. These flow-based rules match packets based on various packet header fields (e.g., source and destination IP addresses, ports) and perform simple actions (e.g., dropping, counting, forwarding).

While OpenFlow enables control plane programmability, it is widely acknowledged that it is challenging to write portable, correct, and efficient applications\(^1\) on top of it [7]. This is due to the mismatch between application expectations and (i) switch heterogeneity (ii) the primitives offered by OpenFlow. Ideally, an SDN programmer expects to program a fast, full-featured switch without being encumbered by resource constraints, and to update rules quickly and correctly. However, in reality, she has to deal with a range of switches that differ significantly in architecture (e.g., hardware vs. software), feature set (e.g., flow table sizes, supported rewrites), and performance (e.g., fast vs. slow path) and their interpretation of the specification (e.g., for the BARRIER command) [4]. Consequently, each application has to be adopted to each specific switch design to work correctly and perform optimally, resulting in duplication of effort and a combinatorial explosion of customizations.

We propose to focus on operations on one switch, separate application expectations from the switch implementation specifics, and create an interface that enables portability while preserving efficiency and switch-level expressiveness. In contrast to previous work on abstracted programming models for the entire network [10, 5, 8], we introduce a low-level API called NOSIX\(^2\), as a minimalistic API optimized for low-level applications “close to the metal” and for building other controller frameworks (Figure 1). Thus, we focus on primitives that are (i) local, i.e., that can be implemented on a single switch, (ii) ubiquitous, i.e., that are required by most applications, and (iii) expressive, i.e., that don’t restrict the expressiveness of OpenFlow.

At the core of our API, we define the virtual flow table, which is an idealized flow table that has no re-

source constraints and full feature-set. It represents the application’s view of the currently active flow-level rules. NOSIX supports consistent updates of the virtual flow table (i.e., no invalid intermediate states are exposed) and collects counters efficiently from the switch. Underneath NOSIX, vendor-defined switch drivers in the controller map the virtual flow table to the physical flow tables of the switches, and thus hide the switch diversity and perform switch-specific optimization.

NOSIX facilitates writing portable SDN applications through its simple model of an idealized OpenFlow switch, which is expressive enough to be amenable to low-level tasks. NOSIX is useful as a building block for other high-level controller frameworks, reducing duplication of efforts in programming diverse switches. Also, moving the common switch API from the switches up to the controller opens up space for optimization below, as vendors can utilize the specific strengths of their forwarding hardware and optimize the controller-switch communication protocol.

The poster will discuss the design space of creating a universally useful shared low-level API for Software Defined Networks. We also present a concrete architecture draft of NOSIX. We hope that our work can give some impulses to the further development of an Open and portable SDN stack, and incite a discussion within the SDN community about the right common interface for programming SDN switches.

\(^1\)We use applications to denote both controller frameworks and the low-level applications built directly on OpenFlow-like flow model.

\(^2\)In allusion to the well known POSIX standard that defines a low-level OS API that enables portability for OS programs, and serves as a building block for high level platforms.

---

**Figure 1:** NOSIX

![Diagram showing the NOSIX interface between controller and switch drivers](image-url)