daPIPE
DAta Plane Incremental Programming Environment

Mario Baldi
Cisco Systems, Technology Director

While having a programmable data plane opens up several new opportunities, taking advantage of P4 to implement custom packet processing and forwarding behaviors implies a paradigm shift in the way operators and enterprise network administrators deploy network devices. Their network engineers must write their own data plane programs – and consequently also control plane software – possibly starting from open source code. Instead, traditionally they have relied on proven network operating systems, feature-rich data planes, and broad sets of network applications found in turn-key vendor systems or in the open-source community for execution on open platforms.

This demo walks through *daPIPE*, a *DAta Plane Incremental Programming Environment*, that supports a developer, here called *incremental programmer*, in adding data plane features to a network device running a pre-existing data plane program and a corresponding network operating system or controller.

Adding features to the data plane of both a turn-key network device (i.e., a hardware platform distributed by a vendor with a specific data plane program and associated network operating system) and a disaggregated network software stack (i.e., a data plane program and associated network operating system developed for an open platform) leads to a number of challenges mostly stemming from the fact that P4 was not designed for modular, or *incremental*, *programming*. The currently available version of the language and related software development tools assume that a program is written by a single programmer or a group of programmers working together. This is problematic as the network engineer might not want to have to familiarize with the P4 program shipped with a turn-key box or downloaded from an open-source repository, and deal with its extent and complexity, for the sake of just adding a few lines of code. Moreover, modifying the native P4 program can compromise the associated network operating system ability to properly control the native data plane functions, which would make them ineffective, and possibly the whole switching system unusable.

This demo shows how daPIPE addresses the above challenges and can be used to add custom features to the pre-existing data plane of a Cisco Nexus 3400, based on Barefoot Networks' Tofino P4-programmable forwarding engine. As an example, we use daPIPE to add the capability of switching media flows based on the RTP timestamp, a well-known use case published by Fox Advanced Technology on Github (https://github.com/FOXNEOAdvancedTechnology/ts_switching_P4). In doing so we demonstrate how daPIPE, not only ensures that the custom functions do not compromise the native functions of Cisco NX-OS, but also guides the incremental programmers and streamlines their work, to the point that they do not even need to have access to the Nexus 3400 native P4 program.