NS4: A P4-driven Network Simulator

Chengze Fan, Jun Bi, Yu Zhou, Cheng Zhang, Haisu Yu
Tsinghua University
fcz14@mails.tsinghua.edu.cn, junbi@tsinghua.edu.cn,
{y-zhou16,zhang-cheng13,yuhs15}@mails.tsinghua.edu.cn

ABSTRACT

We present NS4, a P4-driven network simulator, which is, to the best of our knowledge, the first research effort in applying P4 [1] to network simulation. Key features of NS4 are: (1) elimination of laborious and redundant work for developing internal models of the simulator; (2) direct migration from simulation code to real-world P4 devices; (3) simulation of P4-enabled devices and network system; (4) seamless compatibility with ns-3; (5) tremendous scalability over other P4 behavioral model validation tools.

We proposed and implemented NS4 by integrating a P4 behavioral model in ns-3, and evaluated its effectiveness by a user case study. Source codes and examples of NS4 are publicly available at https://ns-4.github.io/

1. INTRODUCTION

Network simulation in a computer-based environment is widely used in nearly every aspect of network research, education and industry. Network simulators are typically used for two purposes: validating experimental protocols and testing device design correctness before further production. For the latter, engineers are required to first develop a behavioral model of the device as an internal module inside the simulator and then plug it in a large-scale network to test if the behavioral model is well-behaved and robust. During this process, however, traditional network simulators have some significant ***drawbacks*** :

D1: Time-consuming and laborious work is unavoidable to develop the behavioral model in the simulator. Development of behavioral model requires intimate knowledge of the simulator implementation. However, it is frustrating that the learning curve of the simulator implementation is rather steep.

D2: Internal modules of network simulators can hardly be ported to real-world networks. Most of them are developed in general purpose languages (like C++) based on libraries provided by this particular simulator. Therefore simulation codes are distinct from prototype or production version running on real devices (for example ASICs or FPGAs). As a result, simulation codes cannot be directly migrated to products, leading to both wastes of efforts and hazards of incomplete or incorrect simulation.

D3: With an optimistic outlook of P4, we believe there is an urgent need for tools to validate P4 design correctness and simulate P4 performance. As far as we know, there is no network simulator supporting simulation of P4-enabled networks.

In this paper, we step forward to incorporate P4 in the state-of-the-art network simulation platform ns-3 [2]. In order to solve these problems and satisfy practical requirements of simulating programmable networks, we present NS4, a P4-driven network simulator, which is, to the best of our knowledge, the first research effort on applying P4 to network simulation.

For one thing, we notice the root of ***D1,D2*** is that the behavior of a device is tightly coupled with the implementation in ns-3. Therefore laborious work of redundant development and code migration is unavoidable. P4 as a target independent language decouples behavior of devices and ns-3 implementation. Developers can always write P4 programs to rapidly define a particular behavior regardless of simulator implementation, simulate on NS4 and deploy them on P4 targets (CPU, GPU, FPGA, etc.) with almost no overhead and migration effort. (solution of ***D1,D2***). For another, P4 integration in ns-3 makes ns-3 able to simulate a P4 target where P4 programs run. This integration facilitates P4 program behavioral validation and performance evaluation. (solution of ***D3***)

NS4 also stands out from current P4 behavioral validation tools, such as bmv2 [3] and Mininet with advantages as follows: (1) seamless integration with ns-3, which provides a complete simulation toolset and community support. (2) incredible scalability: a single commercial PC can simulate a network system of any size, any underlying channel, any connection speed and any virtual simulation time.

2. DESIGN OF NS4

Design Overview. As shown in Figure 1 NS4 is divided into control plane half and data plane half, which collectively simulate a complete network system with programmable data plane (PDP) . Developers simulate a controller by instantiating a *'Controller Model'* and simulate a P4 device by instantiating a *'Programmable Data Plane Model'*. Traditional devices like router or switch are also supported in NS4 but omitted in the figure.

Workflow. The procedures to simulate a P4-enabled network are shown as numbers in Figure 1 : (1) config-

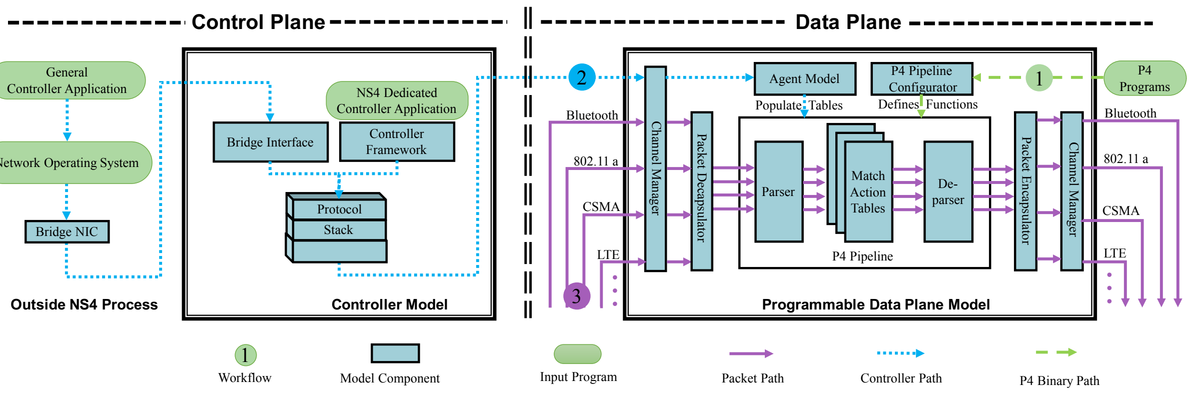

Figure 1: NS4 Design

ure the behavior of data plane by inputting a compiled P4 program to the P4 pipeline configurator; (2) create control plane and populate table entries to P4 devices, by either implementing a controller application inside the controller model or simply running an arbitrary controller application (such as ONOS) outside NS4; (3) add ports (maybe connected to various channels) to the PDP model. At this point, we have this PDP model configured and functional as a P4 device. Next steps are building network topology, installing applications at terminal nodes and triggering the simulator.

Control Plane. A controller does not directly manipulate table entries in P4 devices via function calls. Instead, it is created as an independent network component and populates tables to P4 devices via packets transmission, just in the real-world way. Also, we notice the fact that when focusing on simulating P4 devices, developing a controller application just to populate table entries can be an arduous but unnecessary work, so we make the controller model able to communicate with a controller application running outside NS4 process. This bridge connecting from P4 devices in NS4 process and controllers outside NS4 process eliminates unnecessary work of developing controller applications for simulation environment and also provides a method of testing general controller applications.

Programmable Data Plane. Unlike fixed-function devices, P4 devices need setting up simulation environment to define their behaviors (see workflow paragraph). The core of the PDP model is a complete P4 pipeline which contains a parser, match-action tables, a deparser and buffers amid them. We then add a channel manager and a packet de/encapsulator beyond link layer to mask details of underlying channels and provide a uniform interface for the P4 pipeline. The channel manager expands the scope of simulatable P4 devices, from devices connected via NICs to ones connected via virtually any channel.

Note that the path from P4 devices to a controller is not shown in the figure for simplicity, but this can be easily established through the reverse controller path.

3. PROOF OF CONCEPT AND DEMO

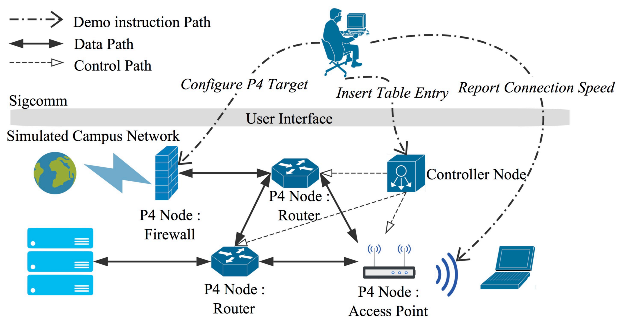

Figure 2: Demo

In order to further demonstrate the effectiveness and convenience of NS4, We orchestrate a user case of a typical campus network miniature with heterogeneous network involved, as depicted in Figure 2.

During demonstration, we plan to demonstrate the detailed process of simulating the campus network as described above using a local laptop. On site, we will provide an interactive user interface for attendees to (1) explore interested information such as performance statistics (2) dynamically operate NS4 by giving instructions to the simulated network through a user interface. Attendees' possible instructions can perform different types of operations like, changing network topology, configuration of P4 devices, population of tables and many more. The showcase will also demonstrate at length the installation of NS4 environment on Linux operating system, configuration of P4 devices, setting up the controllers, population of table entries, triggering simulator and evaluation of critical and interested statistics in real NS4 usage scenarios.

4. REFERENCES

[1] Bosshart et al. P4: Programming protocol-independent packet processors. *SIGCOMM Comput. Commun. Rev.*, 44(3):87–95, July 2014.

[2] ns-3. https://www.nsnam.org/.

[3] p4lang/behavioral-model. https://github.com/p4lang/behavioral-model.

Demo Description
Our demo is described at length in the paper.
Our demo requirements are as follows:

- **Equipment to be used for the demo** : A laptop and a keyboard
- **Space needed** : A table for a laptop and a keyboard
- **Setup time required** : Less than 1 hour
- **Additional facilities needed, including power and any Internet access requirements** : None