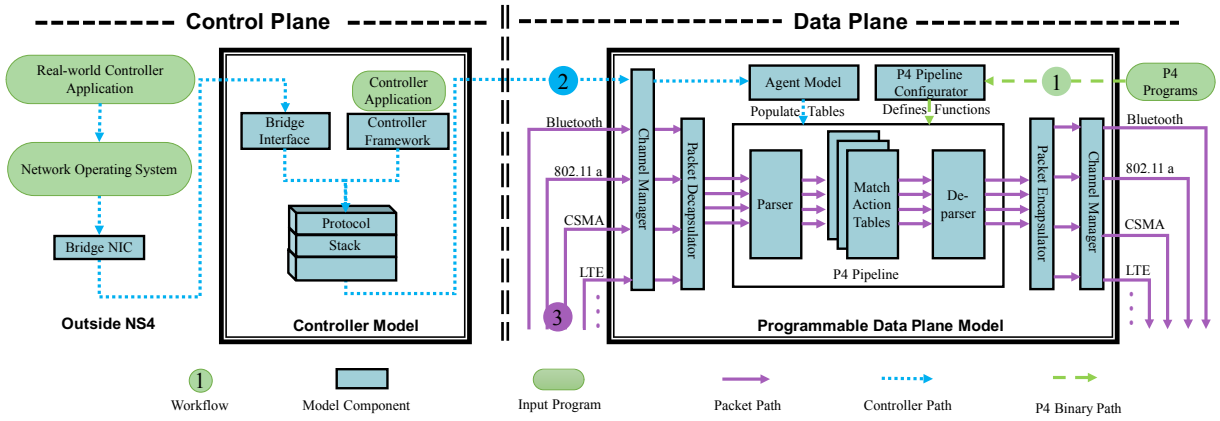

Figure 1: NS4 design overview.

. Developers simulate a controller by instantiating a *'Controller Model'* and simulate a P4 device by instantiating a *'Programmable Data Plane Model'*. Traditional devices like routers or switches 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) configure 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 enable the controller model 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 to set up simulation environment to define their behaviors (see workflow paragraph above). 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.

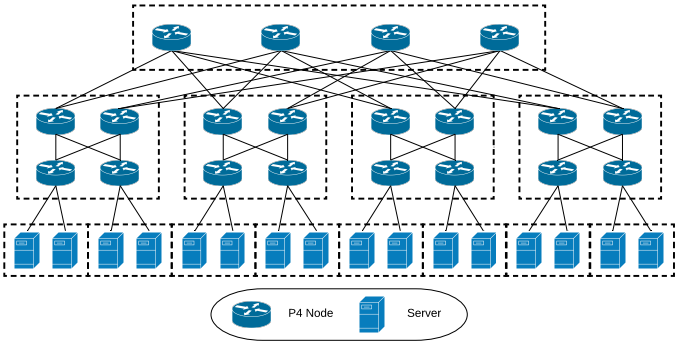

Figure 2: Topology of the demo.

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 DEMO

In order to further demonstrate the effectiveness and convenience of NS4, We simulate a user case of a data center network miniature, as depicted in Figure 2. The data center network is scalable and considered hard to simulate by real traffic because of its complex structure and high performance infrastructure, not to mention introducing P4 into simulation can bring a considerable overhead. We hope to demonstrate the scalability of NS4 by including a large number of P4 nodes in the demo.

During demonstration, we plan to demonstrate the detailed process of simulating a data center network as described above using a laptop. On site, we will showcase at length how to apply P4 codes to NS4 PDP models and how to conduct simulation based on them. We will also exhibit performance comparison between NS4 and mininet, especially in their scalability of network size and transfer speed.

REFERENCES

[1] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker. P4: Programming protocol-independent packet processors. *SIGCOMM Comput. Commun. Rev.*, 44(3):87–95, July 2014.

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

[3] P4 Language Consortium. P4-bmv2. Website. https://github.com/p4lang/behavioral-model.

[4] Bob Lantz, Brandon Heller, and Nick McKeown. A network in a laptop: Rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, Hotnets-IX, pages 19:1–19:6, New York, NY, USA, 2010. ACM.

[5] Barefoot Networks. https://barefootnetworks.com/media/white_papers/Barefoot-Worlds-Fastest-Most-Programmable-Networks.pdf. (Accessed on 07/09/2017).

[6] Bob Lantz et al. What are mininet's limitations? https://github.com/mininet/mininet/wiki/Introduction-to-Mininet#limits. (Accessed on 07/09/2017).