Using FABRIC for Cybertraining on P4 Programmable Data Planes

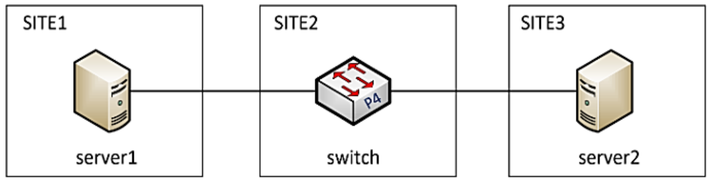
Elie Kfoury, Jorge Crichigno
Integrated Information Technology Department, College of Engineering and Computing
University of South Carolina, Columbia, South Carolina

Abstract

- Recently, data plane programmability has attracted significant attention from both the research community and the industry, permitting programmers to run customized packet processing functions in the data plane.
- The Cyberinfrastructure Lab (CILab) at the University of South Carolina (USC) has developed step-by-step hands-on lab experiments on data plane programming using the P4 language.
- The labs use the Behavioral Model version 2 (BMv2) as the software switch.
- Topics include fundamentals of P4, P4 building blocks, parser implementation in the data plane, populating match-action tables, and others.
- The learner will acquire expertise to create, test, and deploy P4 applications on custom topologies in FABRIC.
- More advanced laboratories are being developed, covering topics such as advanced P4 constructs, advanced parsing, stateful processing, data plane/control plane communication protocols, fine-grained measurements, cybersecurity applications, etc.
- The labs are available to the community through FABRIC's examples repository.

P4 Lab Library

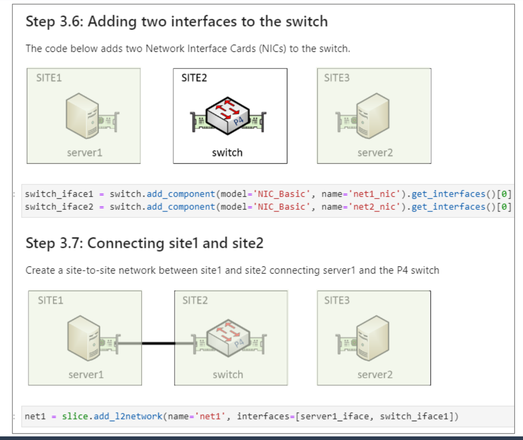
The lab series is developed by the Cyberinfrastructure Lab (CILab) at the University of South Carolina (USC).



Labs:

- Lab 1 - Creating a Slice with a P4 Switch: This lab describes how to create a slice with a P4 switch. It also shows how to deploy the high-performance BMv2 switch to achieve up to ~1Gbps throughput.
- Lab 2 - P4 Program Building Blocks: This lab describes the building blocks and the general structure of a P4 program. It maps the program's components to the Protocol-Independent Switching Architecture (PISA).
- Lab 3 - Parser Implementation: This lab describes how to define custom headers in a P4 program. It then explains how to implement a simple parser that parses the defined headers.
- Lab 4 - Introduction to Match-action Tables: This lab describes match-action tables and how to define them in a P4 program. It then explains the different types of matching that can be performed on keys.
- Lab 5 - Populating and Managing Match-action Tables at Runtime: This lab describes how to populate and manage match-action tables at runtime. It then explains a tool (simple_switch_CLI) that is used with the software switch (BMv2) to manage the tables.
- Lab 6 - Checksum Recalculation and Packet Deparsing: This lab describes how to recompute the checksum of a header. Recomputing the checksum is necessary if the packet header was modified by the P4 program. The lab also describes how a P4 program performs deparsing to emit headers.

Sample Lab: Parser Implementation



Step 3.6: Adding two interfaces to the switch

The code below adds two Network Interface Cards (NICs) to the switch.

SITE1 server1 | SITE2 switch | SITE3 server2

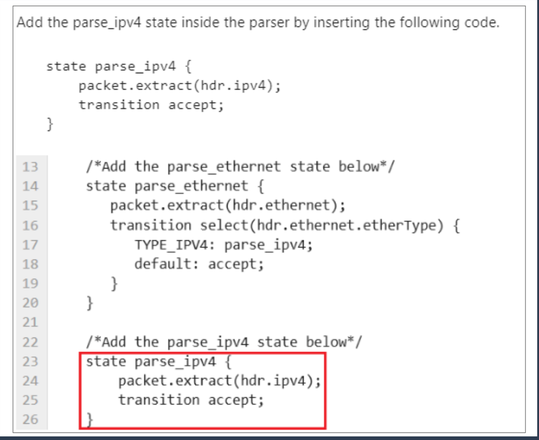
```
switch_iface1 = switch.add_component(model='NIC_Basic', name='net1_nic').get_interfaces()[0]
switch_iface2 = switch.add_component(model='NIC_Basic', name='net2_nic').get_interfaces()[0]
```

Step 3.7: Connecting site1 and site2

Create a site-to-site network between site1 and site2 connecting server1 and the P4 switch

SITE1 server1 | SITE2 switch | SITE3 server2

```
net1 = slice.add_l2network(name='net1', interfaces=[server1_iface, switch_iface1])
```



Add the parse_ipv4 state inside the parser by inserting the following code.

```
state parse_ipv4 {
    packet.extract(hdr.ipv4);
    transition accept;
}
```

```
13    /*Add the parse_ethernet state below*/
14    state parse_ethernet {
15       packet.extract(hdr.ethernet);
16       transition select(hdr.ethernet.etherType) {
17          TYPE_IPV4: parse_ipv4;
18          default: accept;
19       }
20    }
21
22    /*Add the parse_ipv4 state below*/
23    state parse_ipv4 {
24       packet.extract(hdr.ipv4);
25       transition accept;
26    }
```

Getting Started

- The labs are integrated with FABRIC's examples
- start_here.ipynb -> Complex Recipes -> P4 Labs (BMv2)
- Feedback and contributions to additional labs are welcome.

Acknowledgements

This work is in part funded by NSF grant #2118311 and ONR grant #N00014-23-1-2245.

Fine-grained Queue Measurement with P4 on FABRIC