

A Hands-on Workshop on P4 Programmable Switches

Jorge Crichigno, Elie Kfoury
University of South Carolina
<http://ce.sc.edu/cyberinfra>
jcrichigno@cec.sc.edu, ekfoury@email.sc.edu

February 16th, 23rd, 2022

Hands on Session 1: Intro to P4 and BMv2

Examples of P4 Programmable Switches

- Behavioral Model Version 2 (BMv2)
 - Open source
 - Software switch used for teaching, researching ideas
 - Good to validate ideas
- Commercial physical devices
 - E.g., Edgecore Wedge 100BF-65X (based on Intel's Tofino chip)
 - 65x100G switch ports
 - Used in production networks and research
 - Software license and confidentiality agreement (SLACA) with Intel



Introduction to P4 - BMv2 Lab Series

Lab experiments

- Lab 1: Introduction to Mininet
- Lab 2: Introduction to P4 and BMv2
- Lab 3: P4 Program Building Blocks
- Lab 4: Parser Implementation
- Lab 5: Introduction to Match-action Tables (Part 1)
- Lab 6: Introduction to Match-action Tables (Part 2)
- Lab 7: Populating and Managing Match-action Tables
- Lab 8: Checksum Recalculation and Packet Deparsing

Exercises

- Exercise 1: Building a Basic Topology
- Exercise 2: Compiling and Testing a P4 Program
- Exercise 3: Parsing UDP and RTP
- Exercise 4: Building a Simplified NAT
- Exercise 5: Configuring Tables at Runtime
- Exercise 6: Building a Packet Reflector

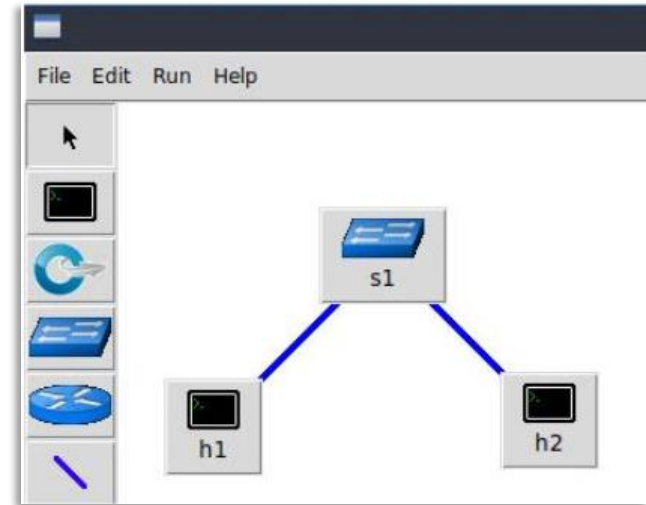
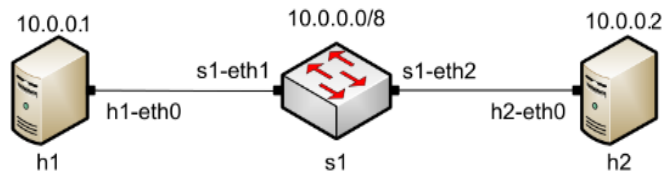
Environment: Mininet

Mininet

- Mininet is a virtual testbed
- Nodes are containers, or more accurately, *network namespaces*
- Features
 - Fast prototyping for new protocols
 - Simplified testing for complex topologies
 - Realistic emulation, real code
 - Open source
 - Complex networks can be created (100s or 1,000s of nodes)

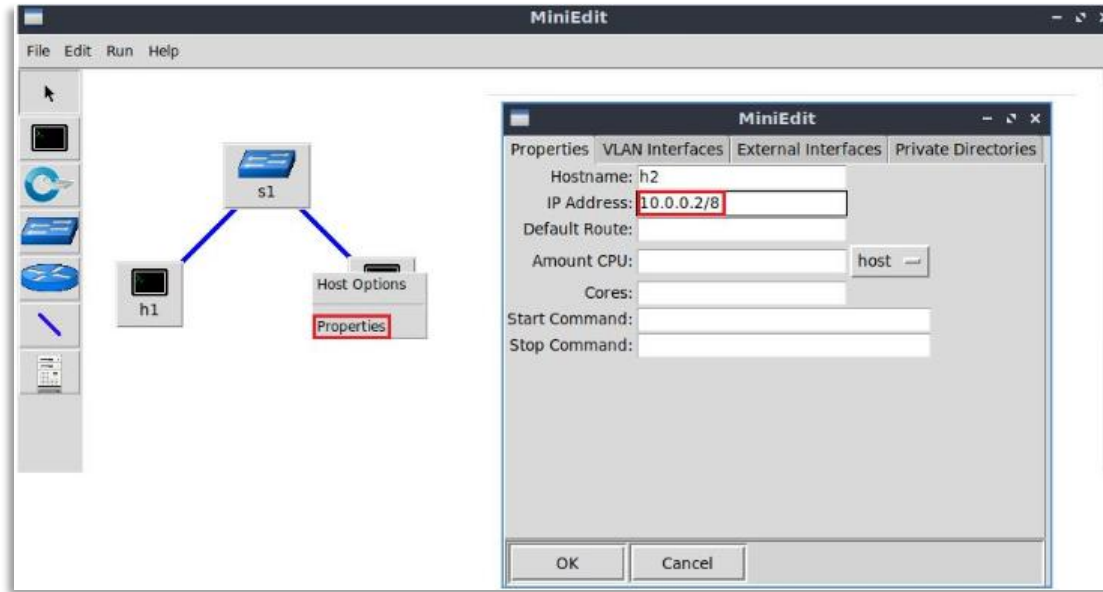
MiniEdit

- To build a topology, we use MiniEdit
- MiniEdit is a simple GUI editor for Mininet
- Example:



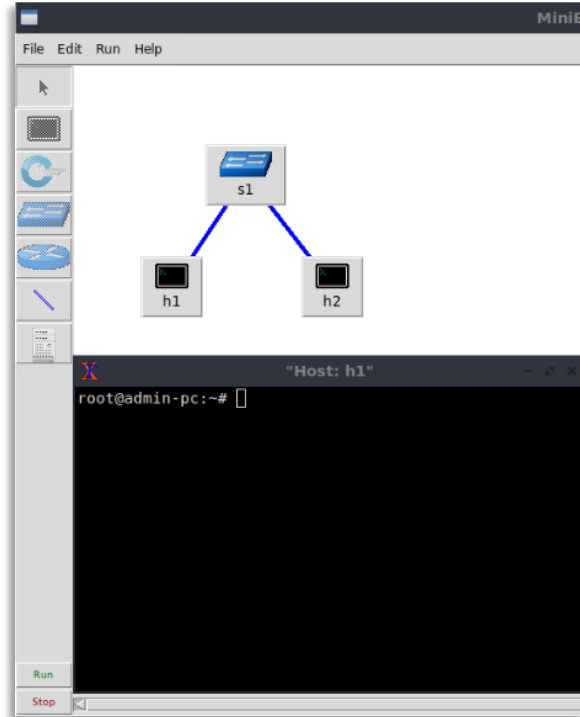
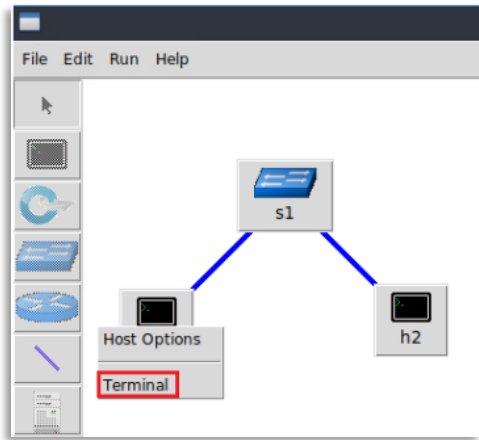
Host Configuration

- A host can be configured by holding the right click and selecting properties on the device



Executing Commands on Hosts

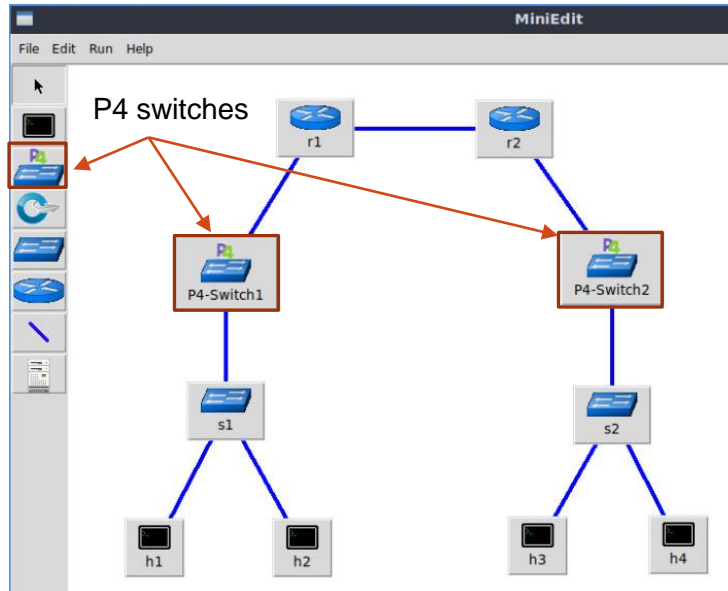
- Open a terminal on host by holding the right click and selecting *Terminal*



```
root@admin-pc:~# ping 10.0.0.2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data:
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.541 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.048 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.033 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.052 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.035 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.042 ms
^C
--- 10.0.0.2 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 110ms
rtt min/avg/max/mdev = 0.033/0.125/0.541/0.186 ms
root@admin-pc:~#
```

Development Environment

- BMv2 switches running inside Docker containers
- Code written in Visual Studio Code with a built-in terminal
- Other devices available: FRR routers, OvS switches, Linux hosts



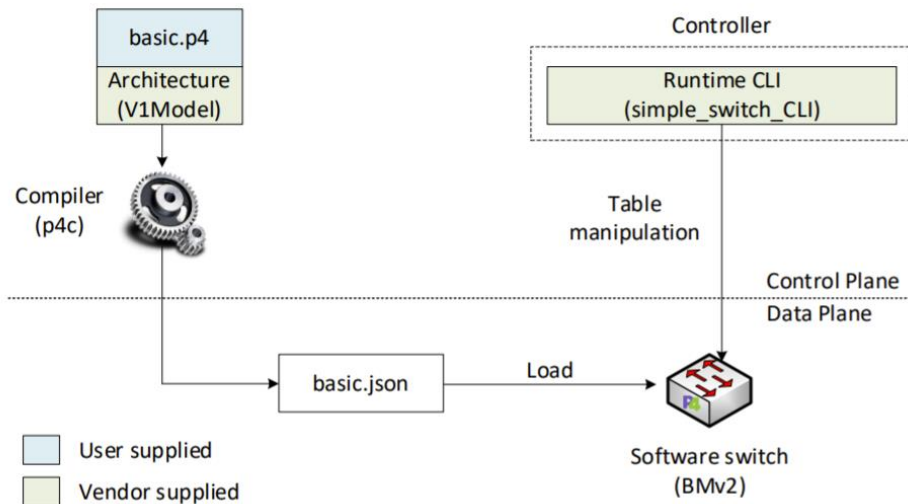
The screenshot shows the Visual Studio Code interface for a file named "basic.p4". The Explorer pane on the left shows the file structure with "basic.p4" selected. The Editor pane displays the following code:

```
1 /* -*- P4_16 -*- */
2 #include <core.p4>
3 #include <v1model.p4>
4
5 const bit<16> TYPE_IPV4 = 0x800;
6
7 /****** HEADERS *****/
8
9
10
11 typedef bit<9> egressSpec_t;
12 typedef bit<48> macAddr_t;
13 typedef bit<32> ip4Addr_t;
14
15 header ethernet_t {
16     macAddr_t dstAddr;
17     macAddr_t srcAddr;
18     bit<16> etherType;
19 }
```

Below the editor is a terminal window with the prompt "admin@lubuntu-vm:~/P4_Labs/Lab2\$".

Workflow of a P4 Program

- Workflow used to program the BMv2 switch

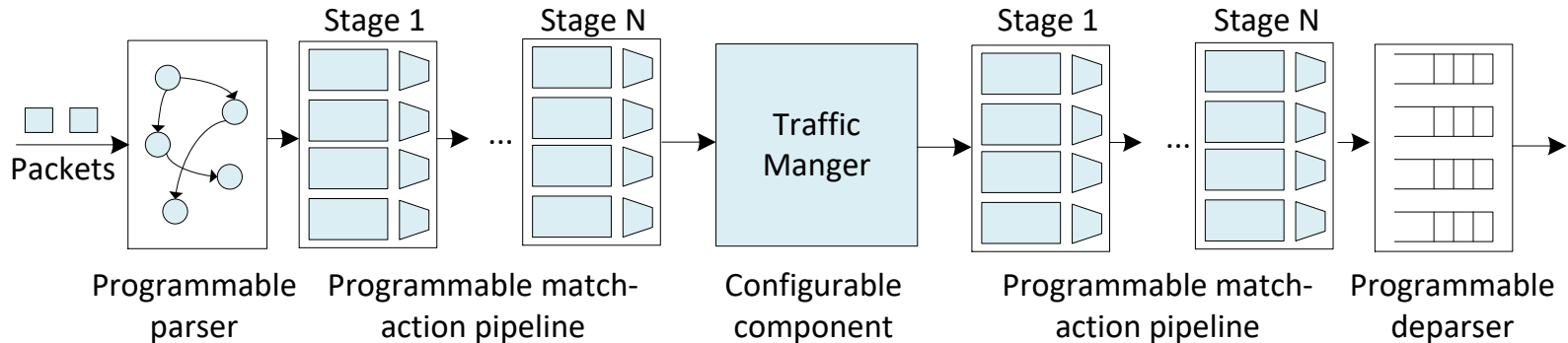


Workflow used in the lab series

Lab 3: P4 Program Building Blocks

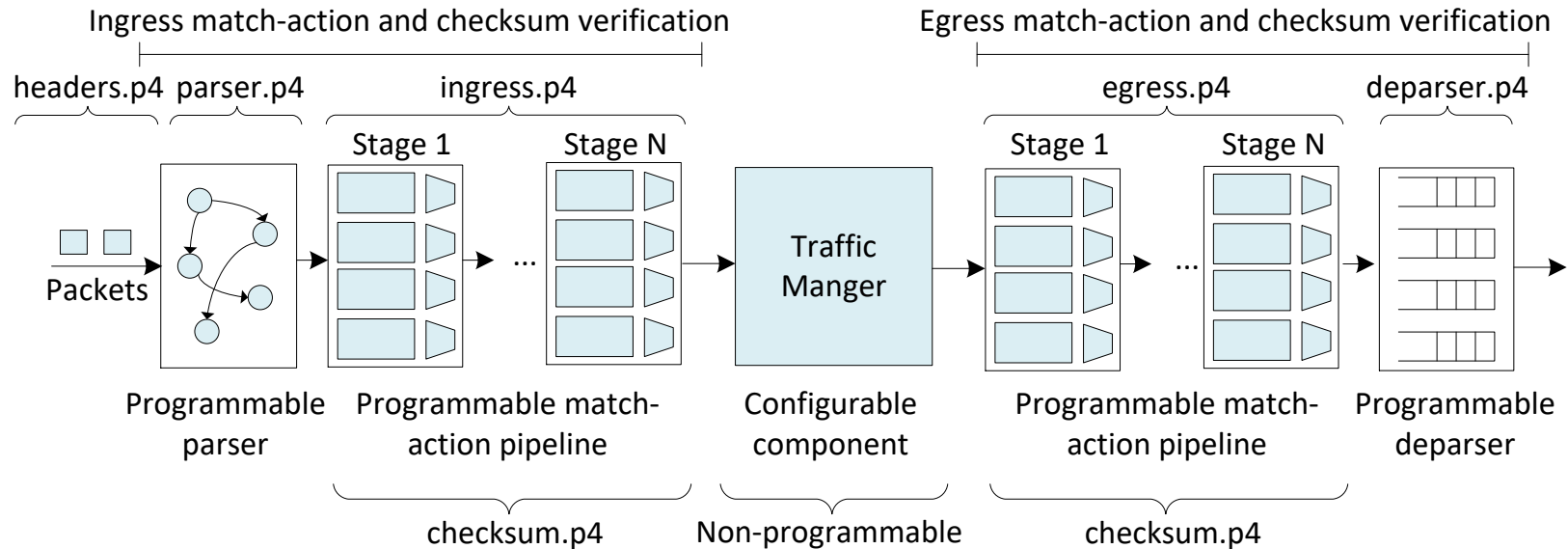
V1 Model

- Common P4₁₆ architecture used with BMv2
- Implemented on top of BMv2's *simple_switch* target
- It consists of a programmable parser, an ingress match action pipeline, a traffic manager, an egress match-action pipeline, and a deparser



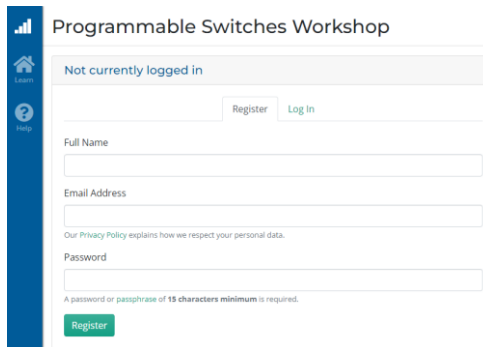
V1 Model

- Common P4₁₆ architecture used with BMv2
- Implemented on top of BMv2's *simple_switch* target



Registering to the Netlab Portal

- If you DID NOT register in the Netlab portal, please use the following link:
- <https://portal.netdevgroup.com/learn/fmgqx8/enroll/>
- Fill out the form with your full name, email address, and password.
- Check your email inbox for the verification key.
- Complete your enrollment by accepting the terms and conditions and claiming your free access
- Finalize the registration by claiming your free access



Programmable Switches Workshop

Not currently logged in

Learn

Help

Register Log In

Full Name

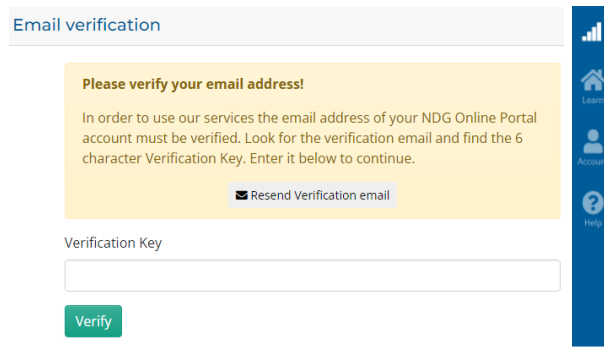
Email Address

Our Privacy Policy explains how we respect your personal data.

Password

A password or passphrase of 15 characters minimum is required.

Register



Email verification

Please verify your email address!

In order to use our services the email address of your NDG Online Portal account must be verified. Look for the verification email and find the 6 character Verification Key. Enter it below to continue.

Resend Verification email

Verification Key

Verify



Programmable Switches Workshop

Learn

Account

Help

Course Access Required

This course may require payment before you can access the labs.

NSF Campus Cyberinfrastructure (CC*) Workshop

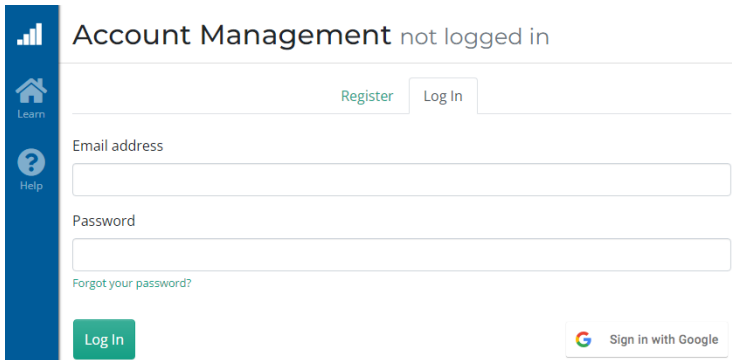
This workshop is sponsored by NSF. Material will be available for 90 days.

The Online Labs from Network Development Group (NDG) provide an online learning environment with easy access right from your web browser.

Claim Free Access

Accessing the P4 labs

- If you already registered, login to the Netlab portal using the following link:
- <https://portal.netdevgroup.com/account/login>
- Click on the “*Programmable Switches Workshop*” course
- Select the lab you want to run (e.g., Lab 3)



Account Management not logged in

Learn

Help

Register Log In

Email address

Password

Forgot your password?

Log In

Sign in with Google



Programmable Switches Workshop

Labs

Lab 1: Introduction to Mininet

Exercise 1: Building a Basic Topology

Lab 2: Introduction to P4 and BMV2

Exercise 2: Compiling and Running a P4 Program

Lab 3: P4 Program Building Blocks

Lab 4: Parser Implementation

Exercise 3: Parsing UDP and RTP

Lab 5: Introduction to Match-action Tables (Part 1)

Lab 6: Introduction to Match-action Tables (Part 2)

Exercise 4: Implementing NAT using Match-action Tables

Lab 7: Populating and Managing Match-action Tables at Runtime

Exercise 5: Configuring Match-action Tables at Runtime

Lab 8: Checksum Recalculation and Packet Deparsing

Exercise 6: Building a Packet Reflector

Learn

Teach

Account

Help

Lab Topology and Objectives

- The topology consists of two hosts: h1 and h2; one P4 switch: s1
- Compiling a P4 program and pushing the output to the data plane
- Starting the switch daemon and allocating interfaces
- Testing and verifying the P4 program

```
root@s1: /behavioral-model
root@s1:/behavioral-model# simple_switch -i 0@s1-eth0 -i 1@s1-eth1 basic.json &
[1] 34
root@s1:/behavioral-model# Calling target program-options parser
Adding interface s1-eth0 as port 0
Adding interface s1-eth1 as port 1
```

