Enabling P4 Hands-on Training over the Cloud: NSF Cybertraining Projects

Elie Kfoury, Jorge Crichigno
https://research.cec.sc.edu/cyberinfra

University of South Carolina (USC)

P4 Developer Days - Online
Wednesday, August 21st, 2024

Agenda

- Motivation
- NSF Cybertraining project 1: Hands-on training on P4 programmable switches
- Cloud system
- Virtual Labs on P4 switches
- NSF Cybertraining project 2: Hands-on training on P4 SmartNICs/DPUs
- FABRIC testbed
- Leveraging FABRIC for education
- Access to the vLabs

Motivation for Virtual Labs and Academic Cloud

- According to the IEEE and ACM[1], the IT curriculum should emphasize "learning IT core concepts with authentic practice" and "use of professional tools and platforms"
  - "It is not enough to simply attend courses and read books. Hands-on learning is essential…"
- Using physical laboratories has been challenging
  - Difficult to scale
  - Expensive (space, maintenance, staff)
  - Since COVID-19 emerged, the capacity of labs has been further reduced (distance requirements)

1. Information Technology Curricula 2017, ACM/IEEE Joint Committee. Online: https://tinyurl.com/4nqqwa5m.

Motivation for Virtual Labs and Academic Cloud

- "The Missing Millions"(NSF report – Oct. 2021. https://tinyurl.com/5awhdazy)
- A report on what can be done to reach out those who are yet to be engaged in STEM workforce
- 15 focus groups, experts on research computing infrastructure
  - "The present research computing and data ecosystems look impenetrable to many of those not yet engaged…"
  - "Lower barriers to entry, but build up the controls at the same time"
  - "Invest in cyberinfrastructure and community laboratories at the edge, enabling broader and more diverse participation in science and engineering"
  - "Explore investments in research computing and data infrastructure approaches that are easily accessible (such as GUIs, science apps, and field tools)"

NSF 2118311: Cybertraining on P4 Programmable Devices using an Online Scalable Platform with Physical and Virtual Switches and Real Protocol Stacks

Start Date: October 1st, 2021

https://www.nsf.gov/awardsearch/showAward?AWD_ID=2118311&HistoricalAwards=false

Cybertraining on P4 Programmable Devices Project

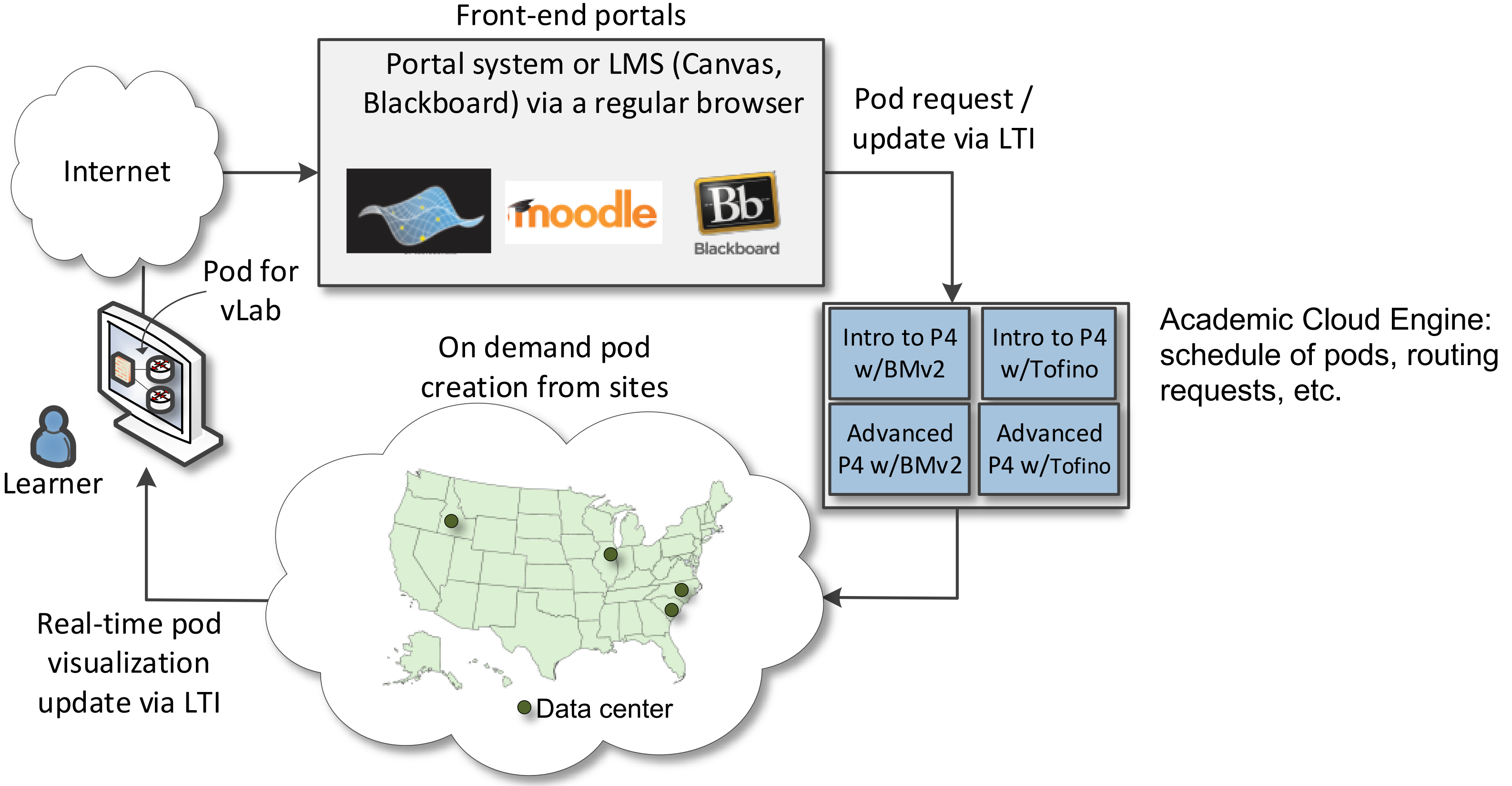
- Goal 1: Increase and facilitate the adoption of programmable P4 devices
  - Obj. 1: Install the equipment for hosting vLabs and remote-access devices in the academic cloud.
  - Obj. 2: Develop vLabs on P4, Next-generation SDN (NG-SDN), SDN, and key legacy technologies.
  - Obj. 3: Provide online, face-to-face, and self-paced training options on programmable technologies for CI engineers and practitioners in general.
- Goal 2: Integrate P4 vLabs material into associate, bachelor, and graduate degrees nationwide
  - Obj. 1: Facilitate adoption of vLabs at academic institutions by embedding access to the cloud into LMS systems and by developing a catalog system.
  - Obj. 2: Provide online and face-to-face workshops for instructors.

Academic Cloud

- The University of South Carolina (USC) (SC), the Network Development Group (NDG) (NC), and Stanly Community College (SCC) (NC) are deploying the Academic Cloud
- A system dedicated to teaching, training, and research
- The Academic Cloud provides remote-access capability to lab equipment via Internet
- It seamlessly pools and shares resources (CPU, memory, storage) from four data centers; resources are allocated to run virtual laboratories

Academic Cloud

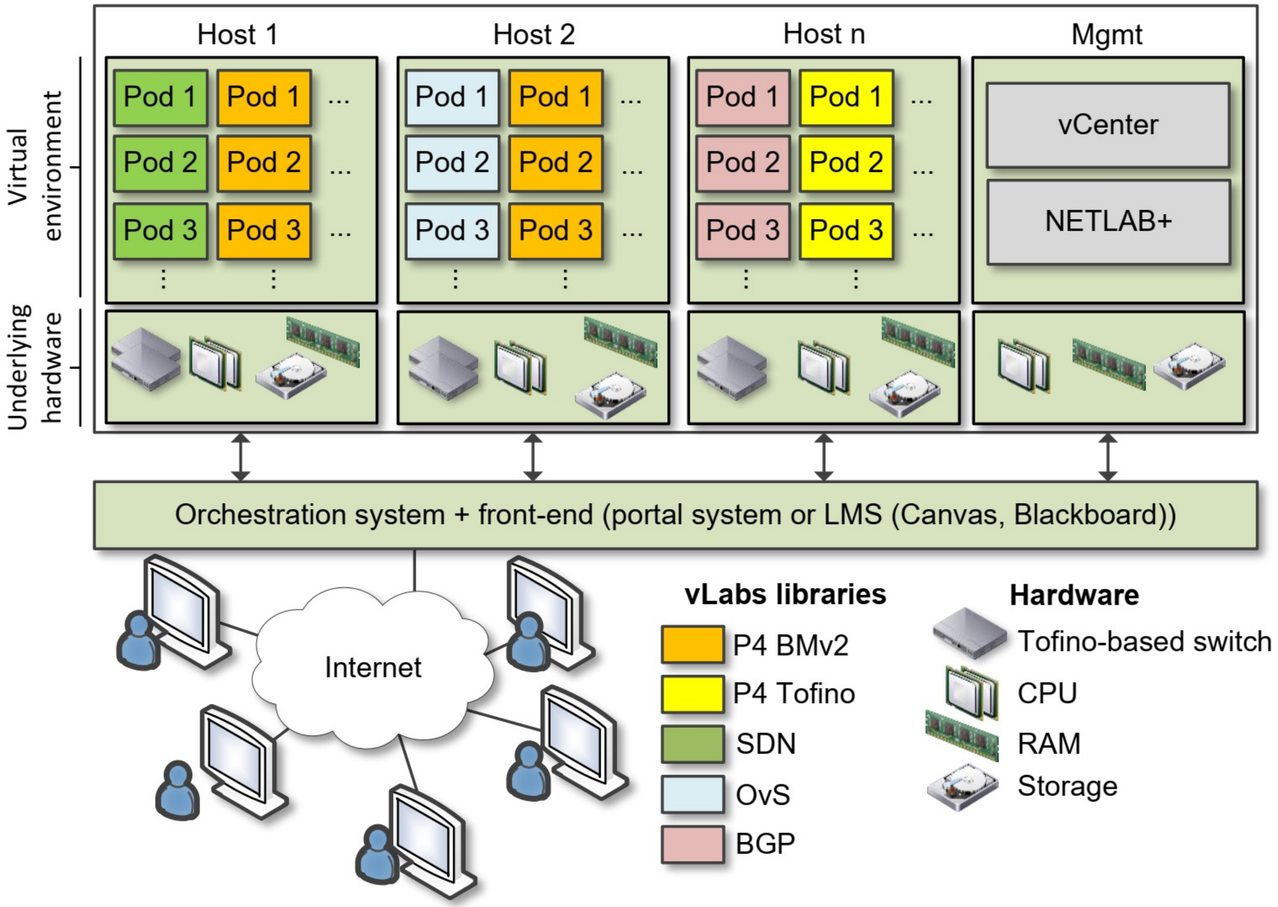
- Data center locations: USC (South Carolina), SCC (North Carolina), NDG (IL), and Idaho National Laboratory (ID)



LMS: Learning Management System. LTI: Learning Tools Interoperability

Inside a Data Center

- Hosts 1-n store virtual machines (VMs) for virtual labs
- Management server runs vCenter, Management Software (NETLAB+)
- Partnership with Network Development Group (NDG)[1]



1. Network Development Group (NDG). Online: https://netdevgroup.com

Inside a Data Center

- Example: Stanly Community College

Device	Cores	Storage (TBs)	RAM Memory (GB)
Server 1 (management SCC)	20	12	264
Server 2 (hosting vLabs pods)	32	4	768
Server 3 (hosting vLabs pods)	32	4	768
Server 4 (hosting vLabs pods)	32	4	768
Server 5 (hosting vLabs pods)	32	4	768
Server 6 (hosting vLabs pods)	32	4	768
Server 7 (hosting vLabs pods)	48	1.92	768
Server 8 (hosting vLabs pods)	48	1.92	768
Server 9 (hosting vLabs pods)	48	1.92	768
TOTAL	**324**	**37.76**	**6408**

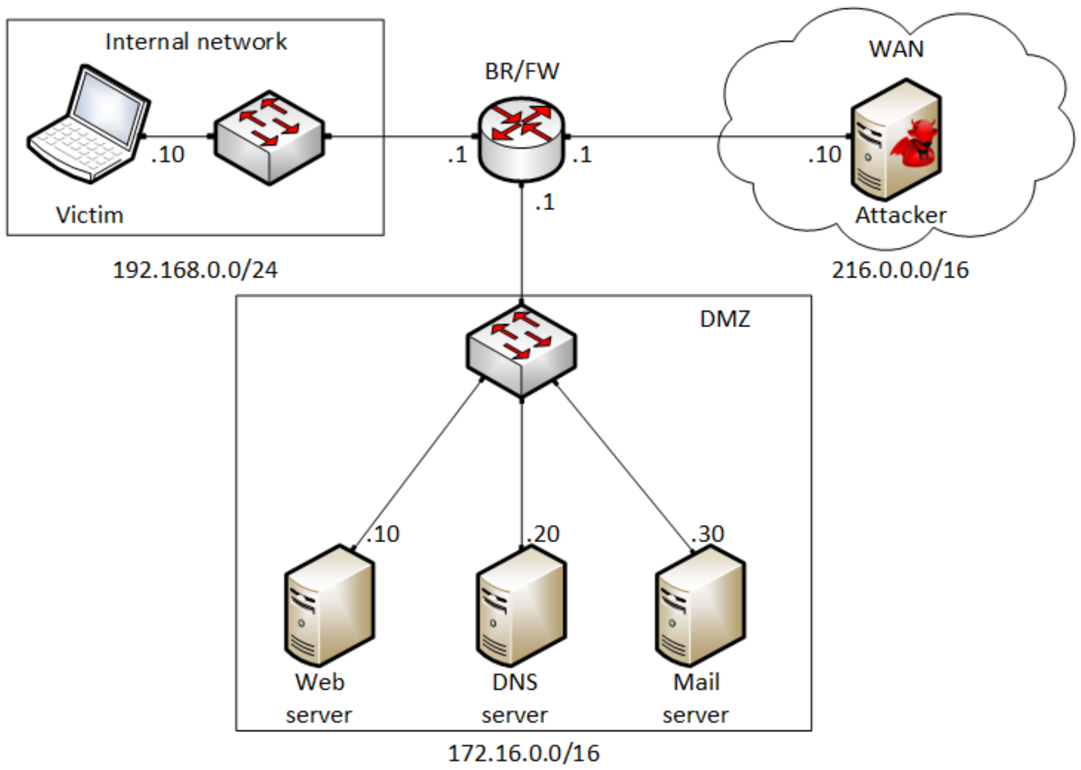
Libraries

- A library consists of between 10-20 lab experiments
- Each lab experiment includes a detailed, step-by-step manual
- Once a learner completes all experiments, the learner acquires significant knowledge and hands-on expertise, and may earn an academic credential or certificate
- Information about libraries are available at
  https://research.cec.sc.edu/cyberinfra/cybertraining

Pod Design – Example 1: Fully Virtual Pod

- A virtual laboratory experiment requires a **pod** of devices, or simply pod
- Example: Cybersecurity Fundamentals

Pod for Cybersecurity Fundamentals Library



Cybersecurity Fundamentals Library

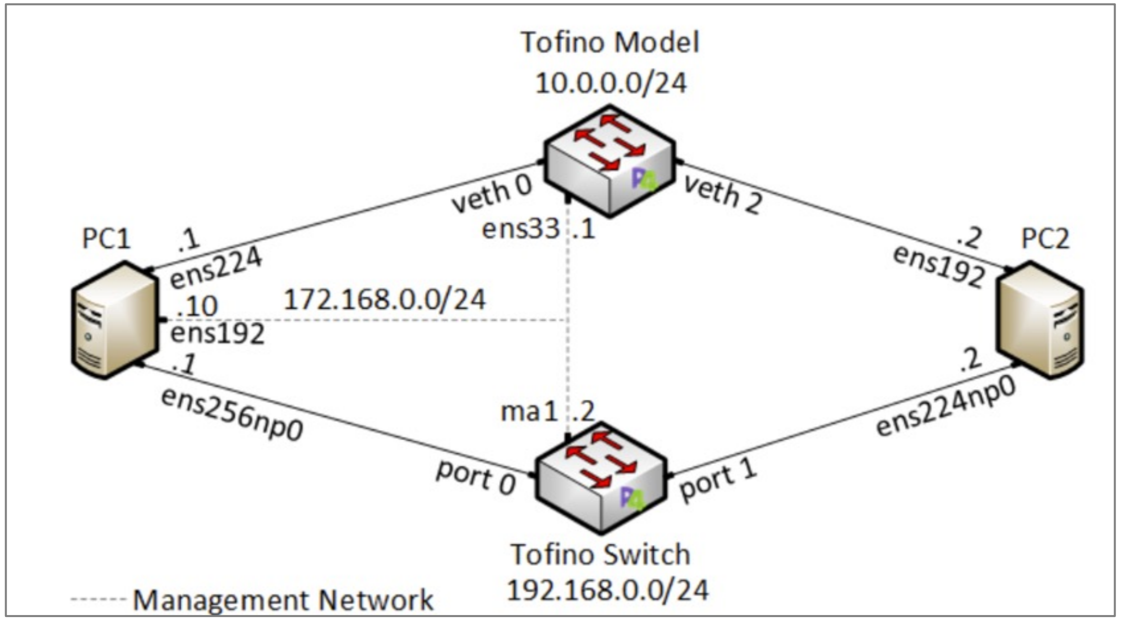
Lab	Topic
1	Reconnaissance: Scanning with NMAP, Vulnerability Assessment with OpenVAS
2	Remote Access Trojan (RAT) using Reverse TCP Meterpreter
3	Escalating Privileges and Installing a Backdoor
4	Collecting Information with Spyware: Screen Captures and Keyloggers
5	Social Engineering Attack: Credentials Harvesting and Remote Access through Phishing Emails
6	SQL Injection Attack on a Web Application
7	Cross-site Scripting (XSS) Attack on a Web Application
8	Denial of Service (DoS) Attacks: SYN/FIN/RST Flood, Smurf attack, and SlowLoris
9	Cryptographic Hashing and Symmetric Encryption
10	Asymmetric Encryption: RSA, Digital Signatures, Diffie-Hellman
11	Public Key Infrastructure: Certificate Authority, Digital Certificate
12	Configuring a Stateful Packet Filter using iptables
13	Online Dictionary Attack against a Login Webpage
14	Intrusion Detection and Prevention using Suricata
15	Packet Sniffing and Relay Attack
16	DNS Cache Poisoning
17	Man in the Middle Attack using ARP Spoofing
18	Understanding Buffer Overflow Attacks in a Vulnerable Application
19	Conducting Offline Password Attacks

DEMO 1 – Cloud System

Pod Design – Example 2: Hybrid Pod

- A pod may also have physical devices and connection to real networks
- Example: Introduction to P4 switches with Tofino

Pod for Introduction to P4 switches with Tofino



Introduction to P4 switches with Tofino library

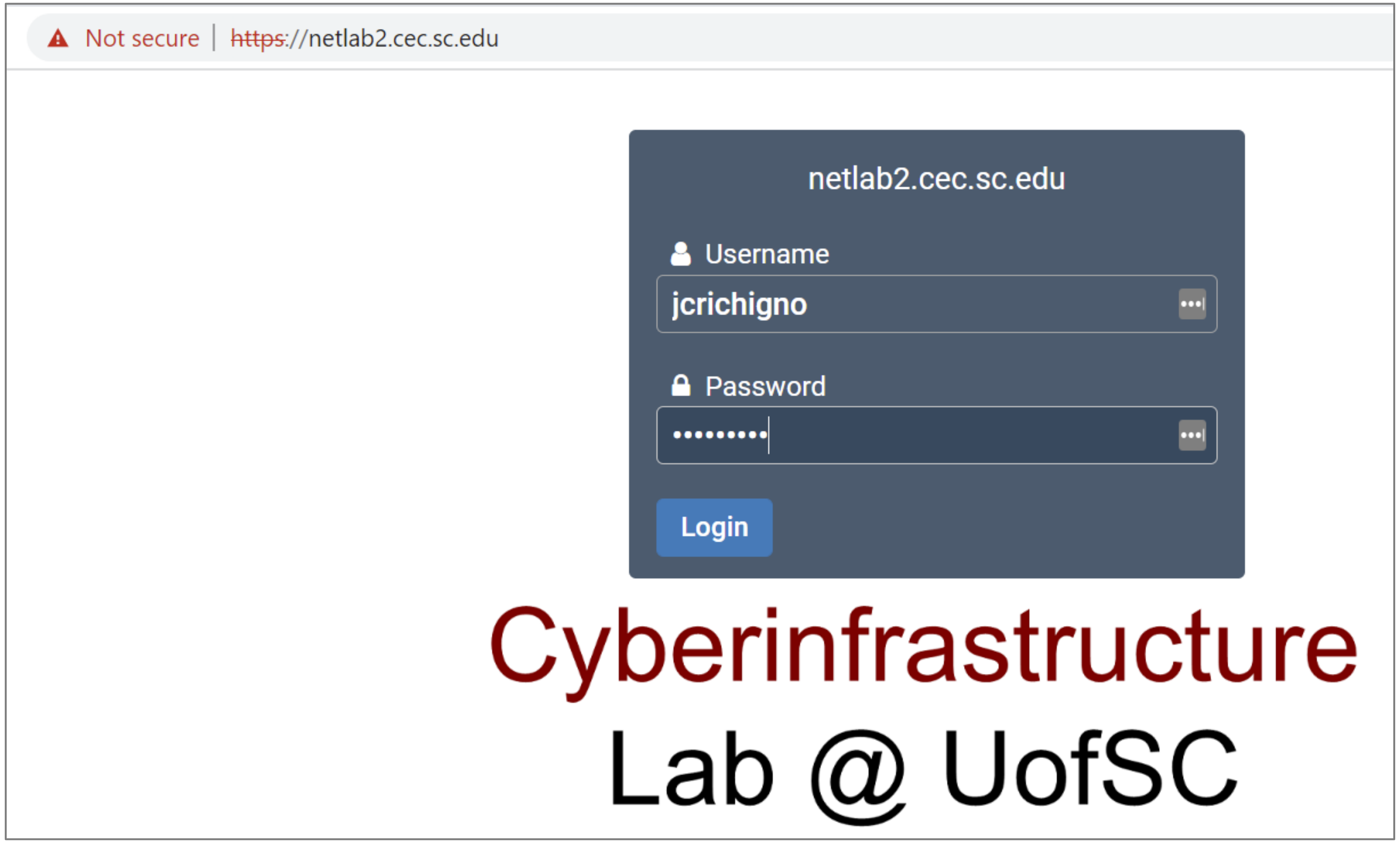
Lab Name
Lab 1: Introduction to P4 and Tofino
Lab 2: Introduction to P4 Tofino Software Development Environment (SDE)
Lab 3: Parser Implementation
Lab 4: Introduction to Match-action Tables
Lab 5: Populating and Managing Match-action Tables at Runtime
Lab 6: Checksum Recalculation and Packet Deparsing
Lab 7: Inspecting the Resource Usage in the Tofino Switch

DEMO 2 – Cloud System

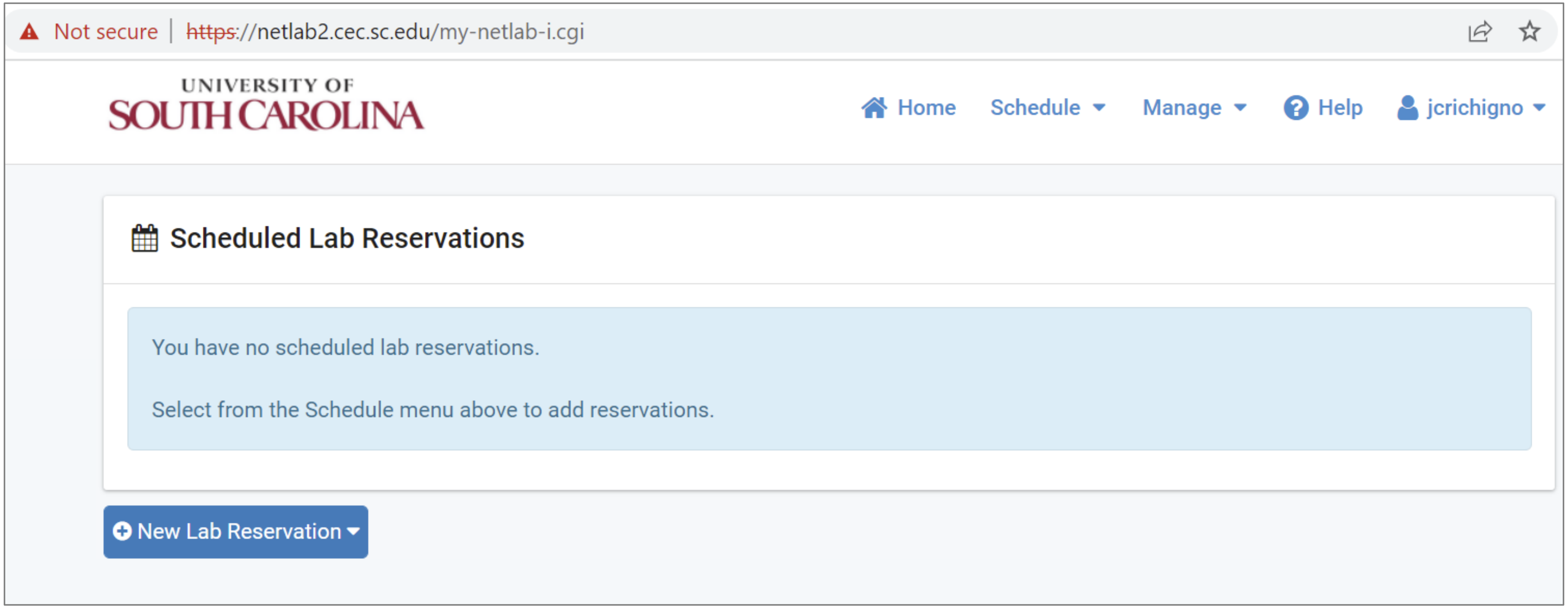
Cloud System

Feature	Description
Allocation of resources	Granular allocation of physical resources
Custom pods	Easy to create custom pods
Cost	Cost-effective when used extensively
Presentation layer for pedagogy	Flexible. Topology is graphically presented to the learner using a regular browser
Time-sharing resource feature	The owner controls who can access resources. Easy to implement time-sharing policies
Integration of physical devices	Physical hardware can be integrated into pods
Flexible use of IP addresses	Each pod runs in an independently controlled environment. Pods (and learners) have the same topology and IP addresses (overlapping addresses without conflict)
Target	Specially built for pedagogy (education, research, and training)
Typical users	From entry-level learners to PhD researchers

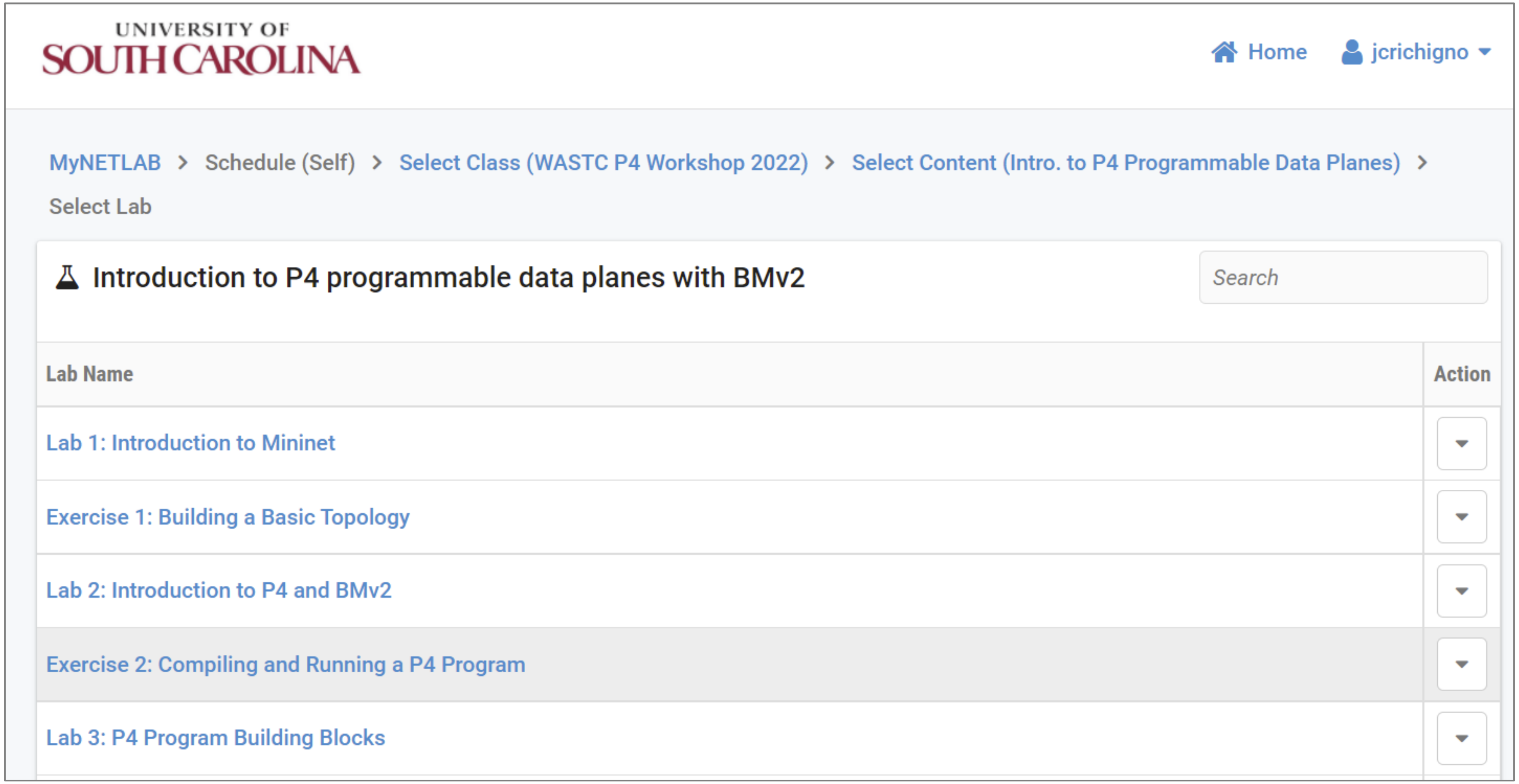
Using the Cloud System



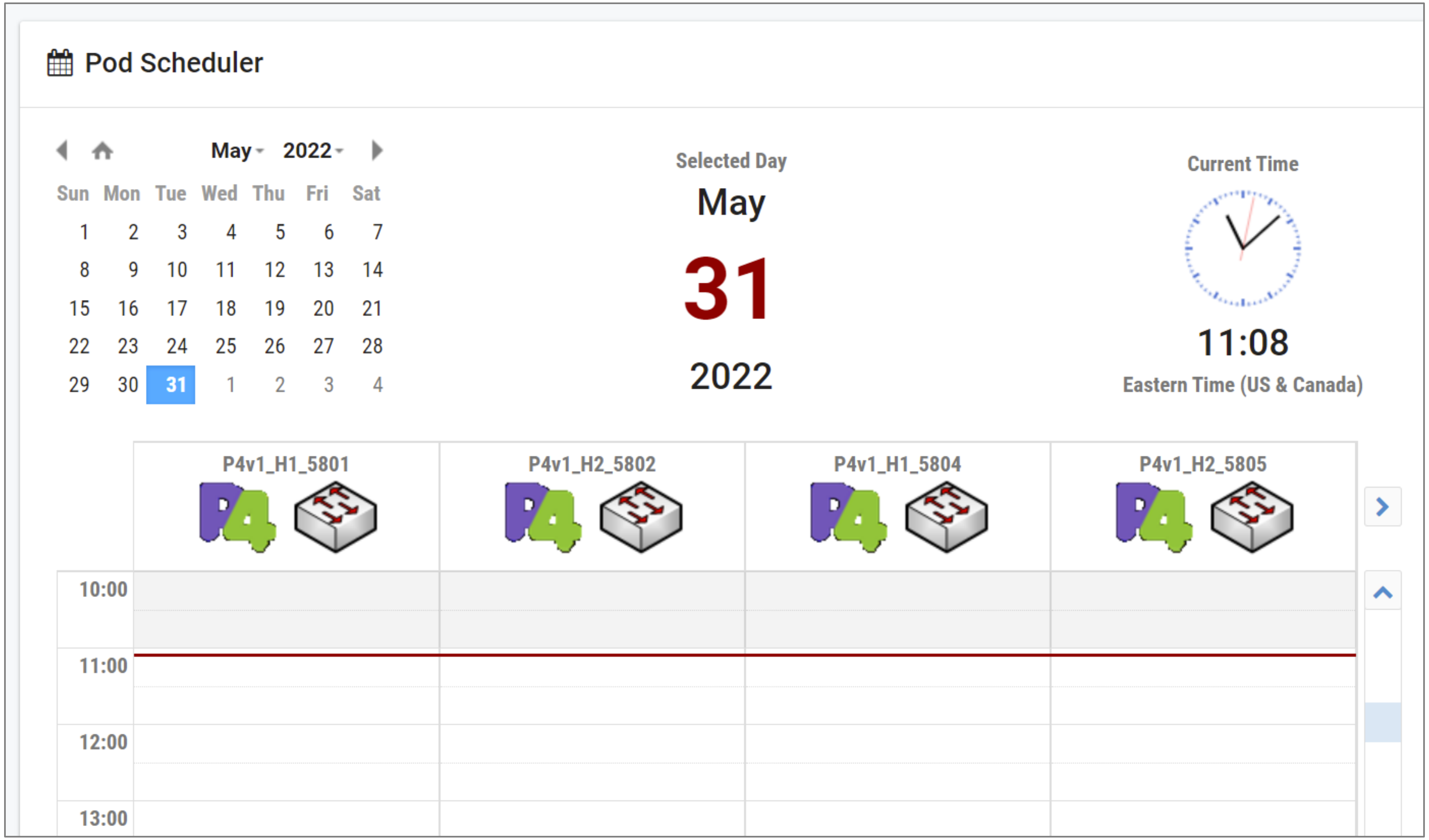
Using the Cloud System



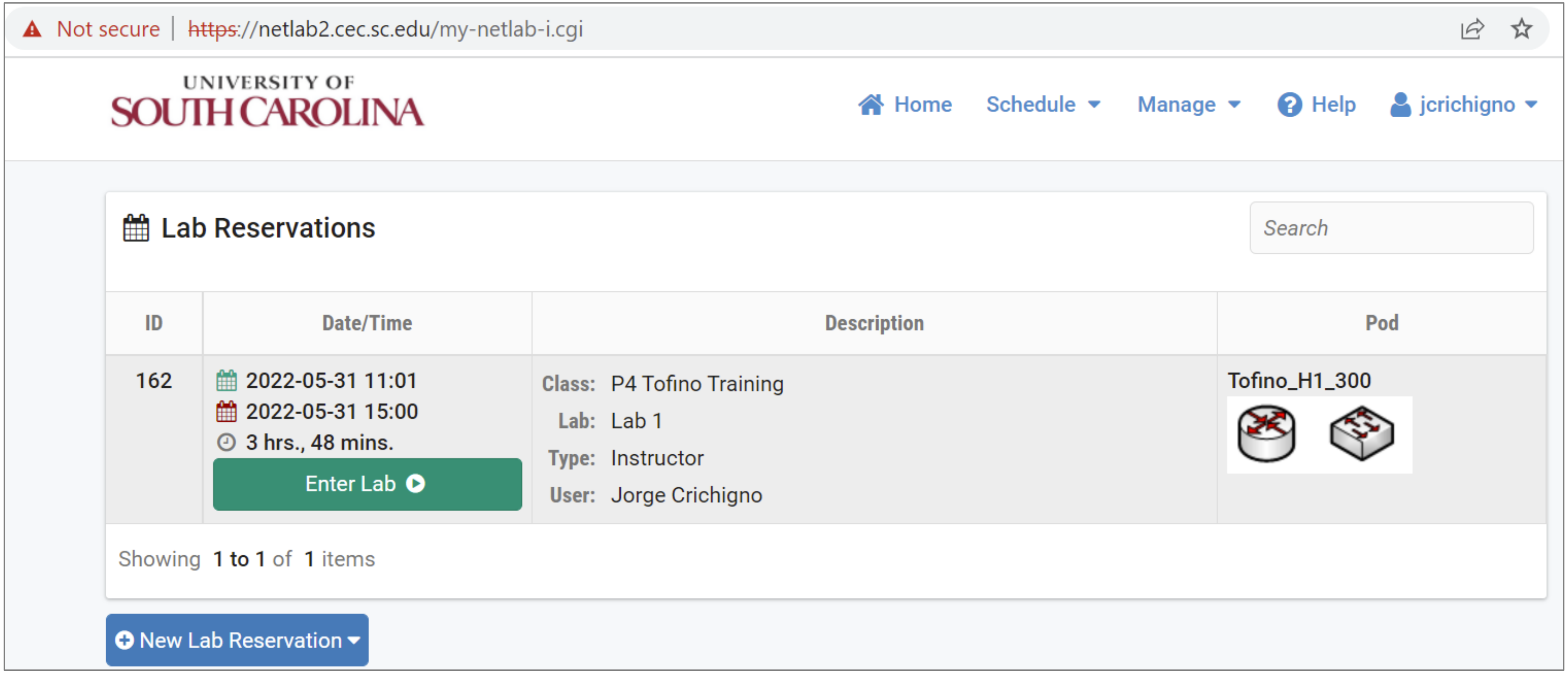
Using the Cloud System



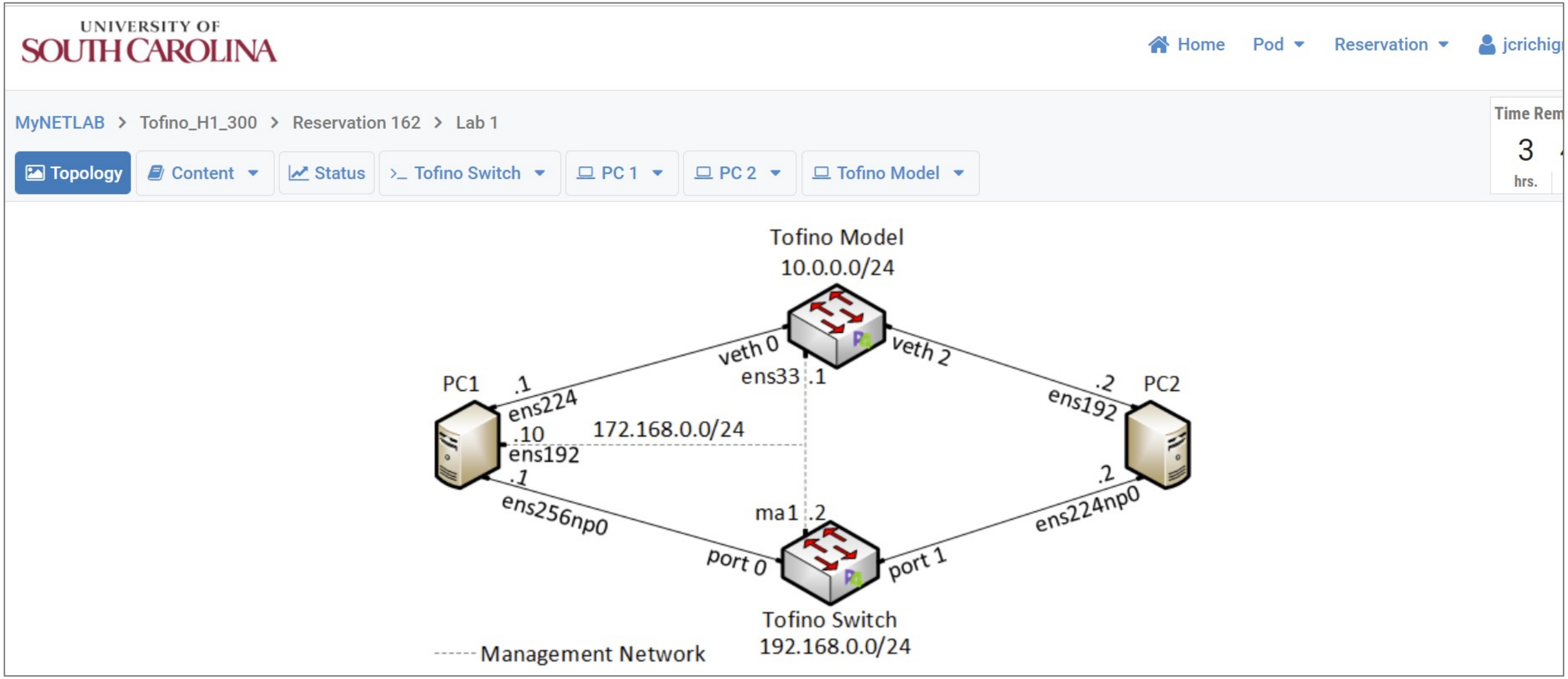
Using the Cloud System



Using the Cloud System



Using the Cloud System



Using the Cloud System

Organization of Lab Manuals

Each lab starts with a section *Overview*

- Objectives
- Lab settings: passwords, device names
- Roadmap: organization of the lab

Section 1

- Background information (theory) of the topic being covered (e.g., match-action tables)
- Section 1 is optional (i.e., the reader can skip this section and move to lab directions)

Section 2… n

- Step-by-step directions

Introduction to P4 and 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

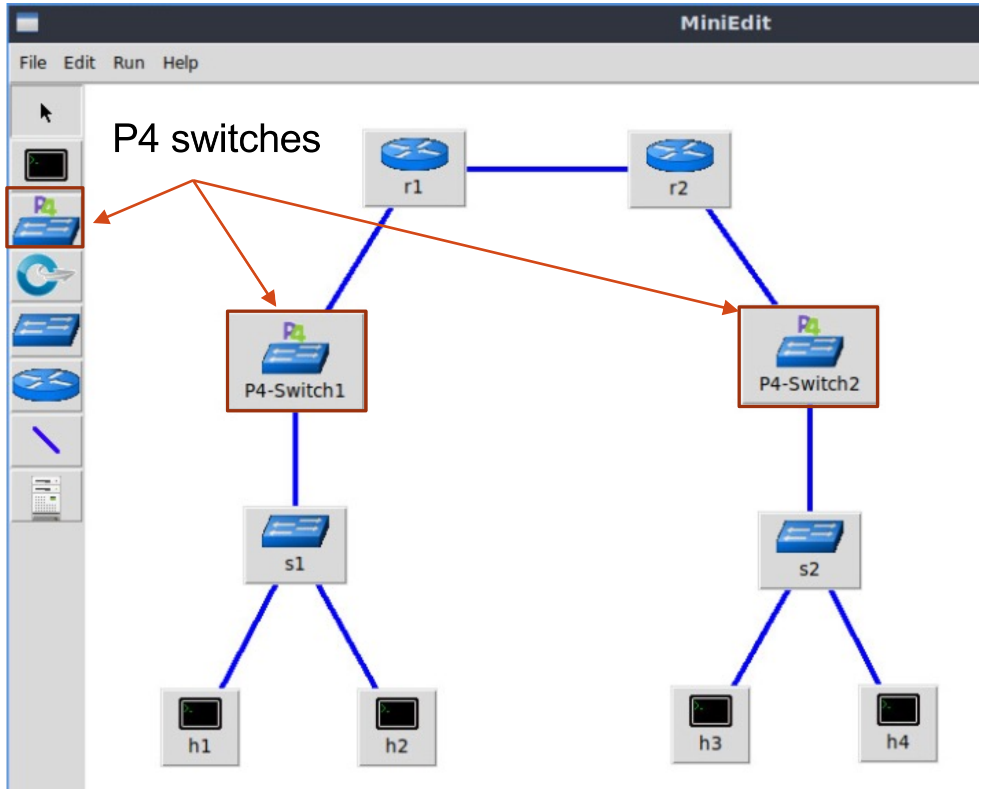
Exercise 4: Building a Simplified NAT

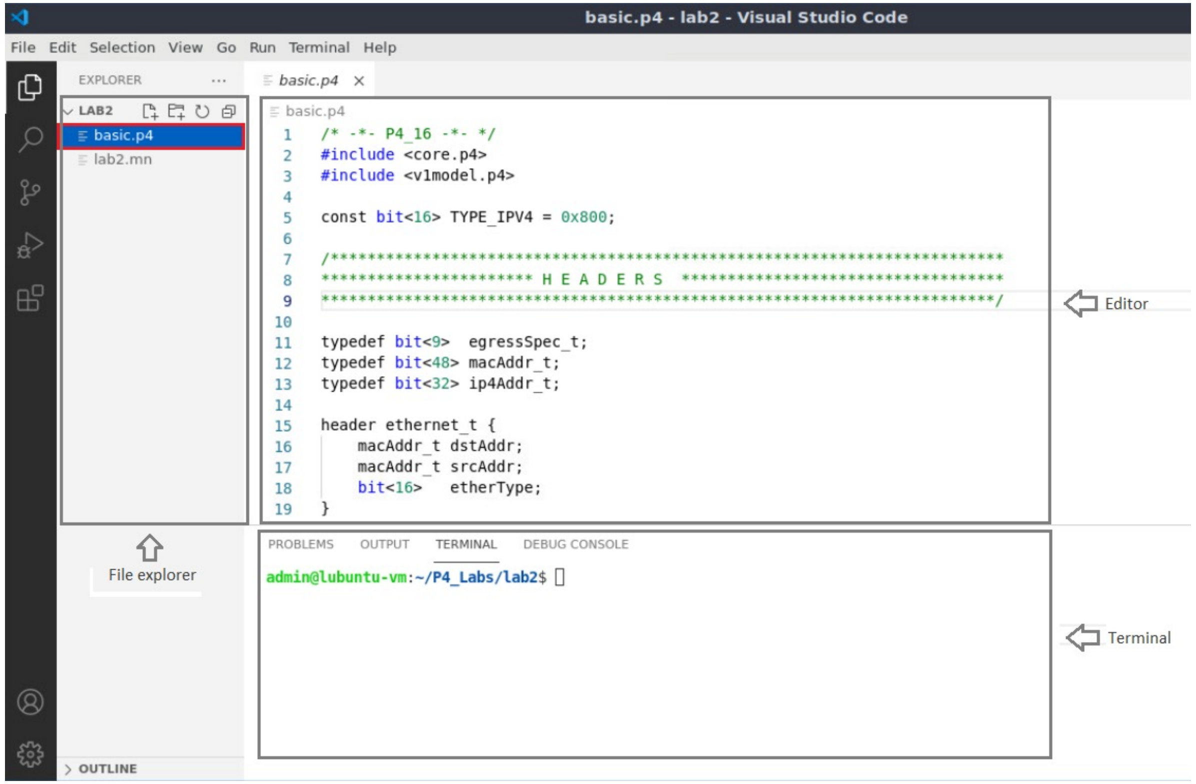
Exercise 5: Configuring Tables at Runtime

Exercise 6: Building a Packet Reflector

Development Environment

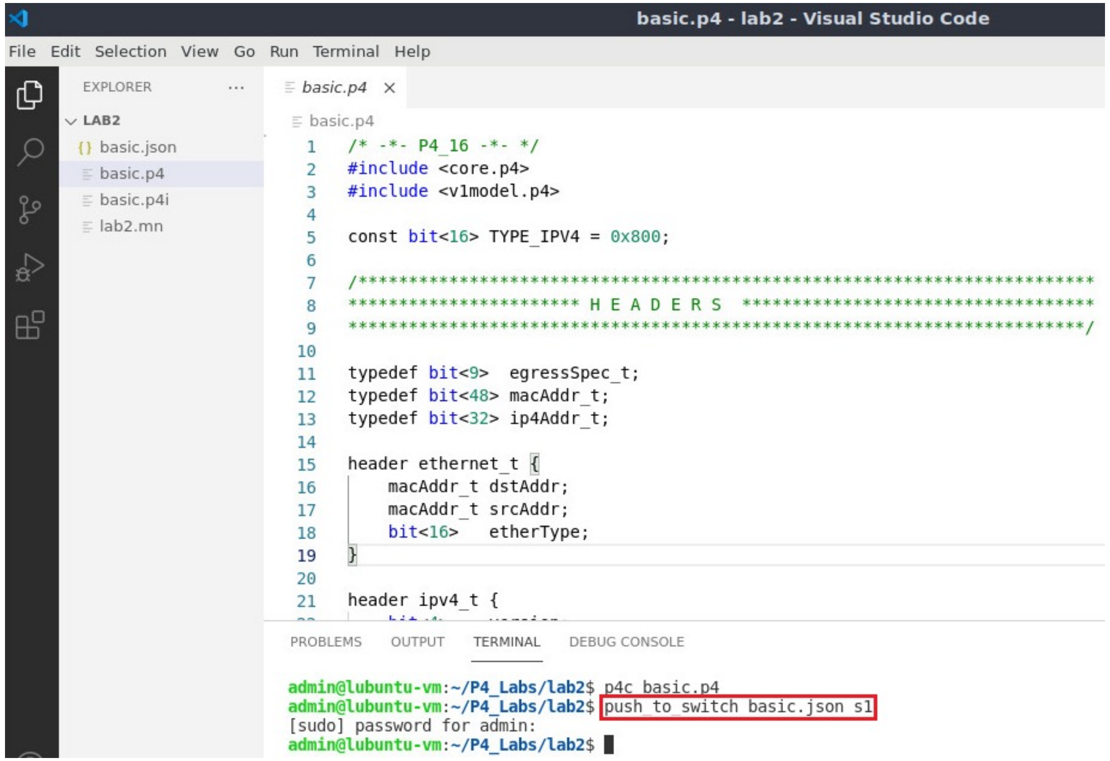
- Topology constructed with a modified version of the MiniEdit editor
- P4 software switches (BMv2) running inside Docker containers (through Containernet)
- Code written in Visual Studio Code with P4 syntax highlighting and a built-in terminal

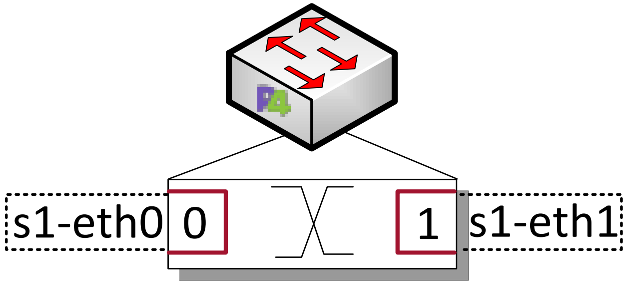


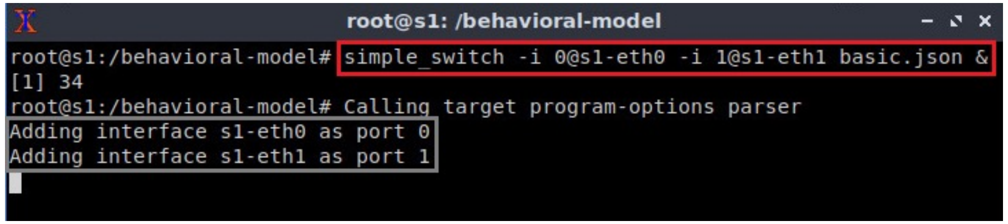


Example Labs

- Compiling a P4 program and pushing the output to the data plane
- Starting the switch daemon and allocating interfaces

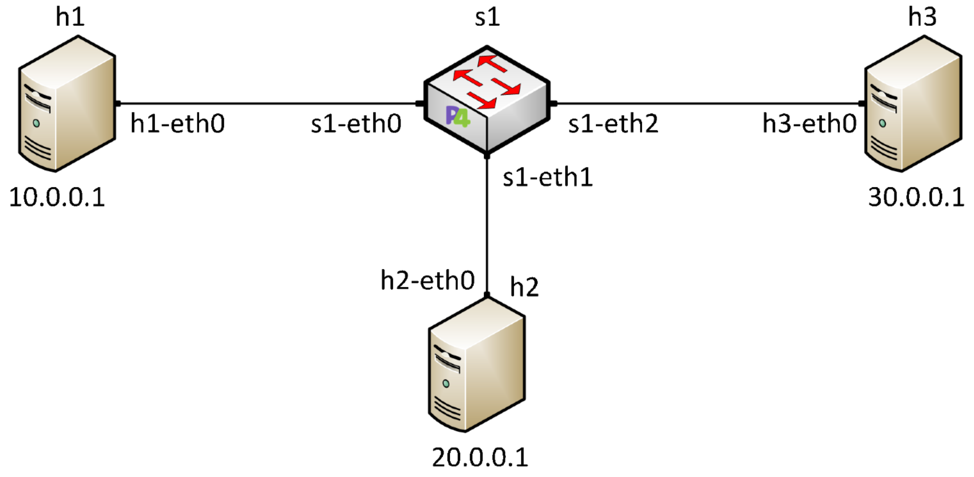






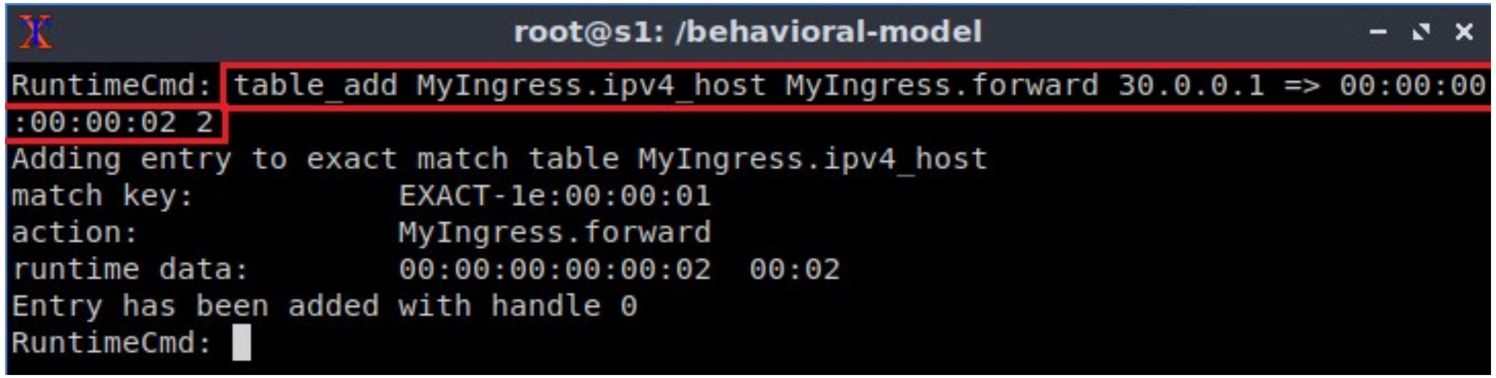
Example Labs

- Programming match-action tables:
  - Exact
  - Longest Prefix Matching (LPM)
- Forwarding using port information:
  - Packets arriving at port 0 are sent through port 1
  - Packets arriving at port 1 are sent through port 0
- Routing using layer-3 information:
  - Matching on the destination IP address
  - Modifying the source and destination MACs
  - Decrementing the Time-to-live (TTL)
  - Assigning the output port



Example Labs

- Populating and managing match-action tables
- Dumping table entries
- Adding/removing/modifying table entries
- Obtaining switch information
- Checking tables



Example Exercises

- Parse UDP and Real-time Transport Protocol (RTP)
- UDP is identified by the "protocol field = 17," in the IPv4 header
- Within UDP, if the destination port = 5004, then the packet is RTP

Packet headers

RTP
UDP
IPv4
Ethernet

UDP header

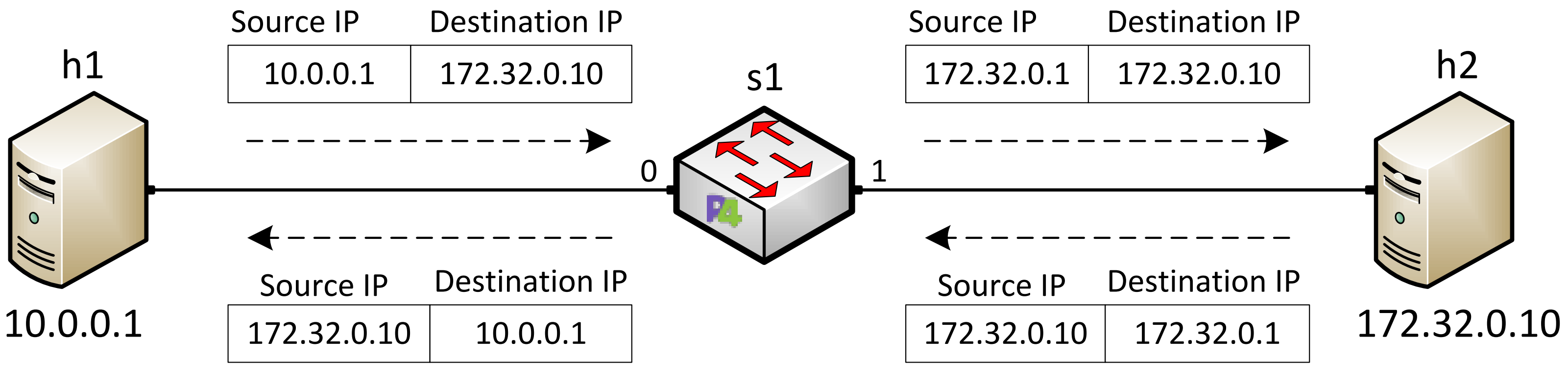
Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Length																Checksum															

RTP header

Offsets	Octet	0								1								2								3							
Octet	Bit [a]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version		P	X	CC				M	PT							Sequence number															
4	32	Timestamp																															
8	64	SSRC identifier																															

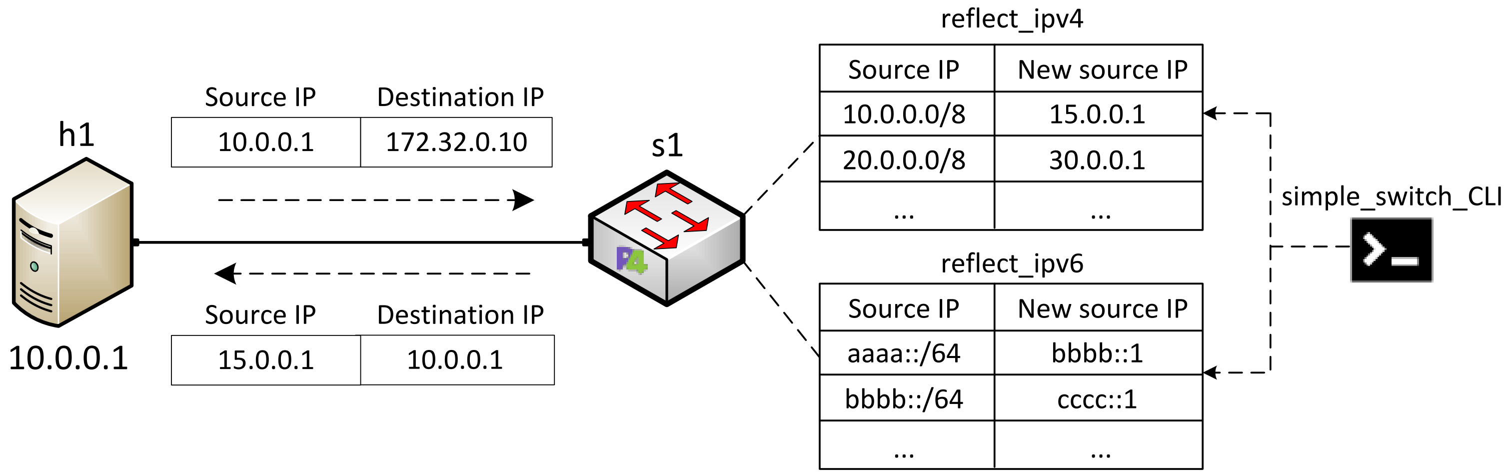
Example Exercises

- Implement a simplified version of the source and destination Network Address Translation (NAT)
- Modify the source IP address of the packet when leaving the network
- Modify the destination IP address of the packet when entering the network



Example Exercises

- Combining all concepts into a single program
- Define headers and parsing IPv4, IPv6
- Implement tables for reflecting IPv4 and IPv6 packets
- Populate the tables from the control plane
- Update the checksum of the IPv4 header



Library on P4 Applications, Stateful Elements, and Custom Packet Processing

Experiments

- Lab 1: Introduction to Mininet
- Lab 2: Introduction to P4 and BMv2
- Lab 3: P4 Program Building Blocks
- Lab 4: Defining and processing custom headers
- Lab 5: Monitoring the Switch's Queue using Standard Metadata
- Lab 6: Collecting Queueing Statistics using a Header Stack
- Lab 7: Measuring Flow Statistics using Direct and Indirect Counters
- Lab 8: Rerouting Traffic using Meters
- Lab 9: Storing Arbitrary Data using Registers
- Lab 10: Calculating Packets Interarrival Time w/ Hashes and Registers
- Lab 11: Generating Notification Messages from the Data Plane

Library on Security Applications with P4

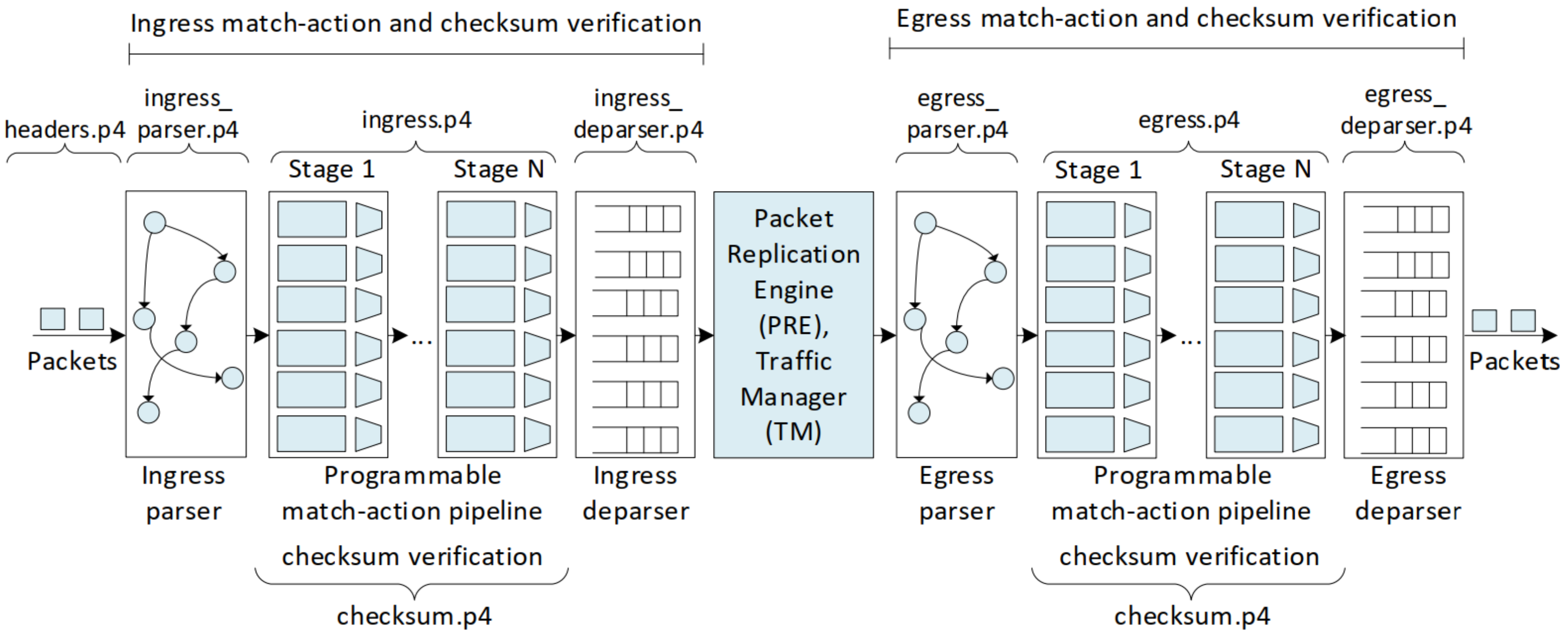
Experiments

- 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
- Lab 6: Implementing a Stateful Packet Filter for the ICMP protocol
- Lab 7: Implementing a Stateful Packet Filter for the TCP protocol
- Lab 8: Detecting and Mitigating the DNS Amplification Attack
- Lab 9: Identifying Heavy Hitters using Count-min Sketches (CMS)
- Lab 10: Limiting the Impact of SYN Flood by Probabilistically Dropping Packets
- Lab 11: Blocking Application Layer Slow DDoS Attack (Slowloris)
- Lab 12: Implementing URL Filtering through Deep Packet Inspection and String Matching

Library on P4 Programmable Data Plane with Tofino

Experiments

- Lab 1: Introduction to P4 and Tofino
- Lab 2: Introduction to P4 Tofino Software Development Environment
- Lab 3: Parser Implementation
- Lab 4: Introduction to Match-Action Tables
- Lab 5: Populating and Managing Match-Action Tables at Runtime
- Lab 6: Checksum Recalculation and Packet Deparsing



Collaborative Research: CyberTraining: Implementation: Medium: CyberTraining on Accelerating Infrastructure Workloads using Next-Generation SmartNICs/DPUs

In collaboration with UNC Chapel Hill (FABRIC Team)

Start Date: August 1st, 2024

https://www.nsf.gov/awardsearch/showAward?AWD_ID=2417823&HistoricalAwards=false

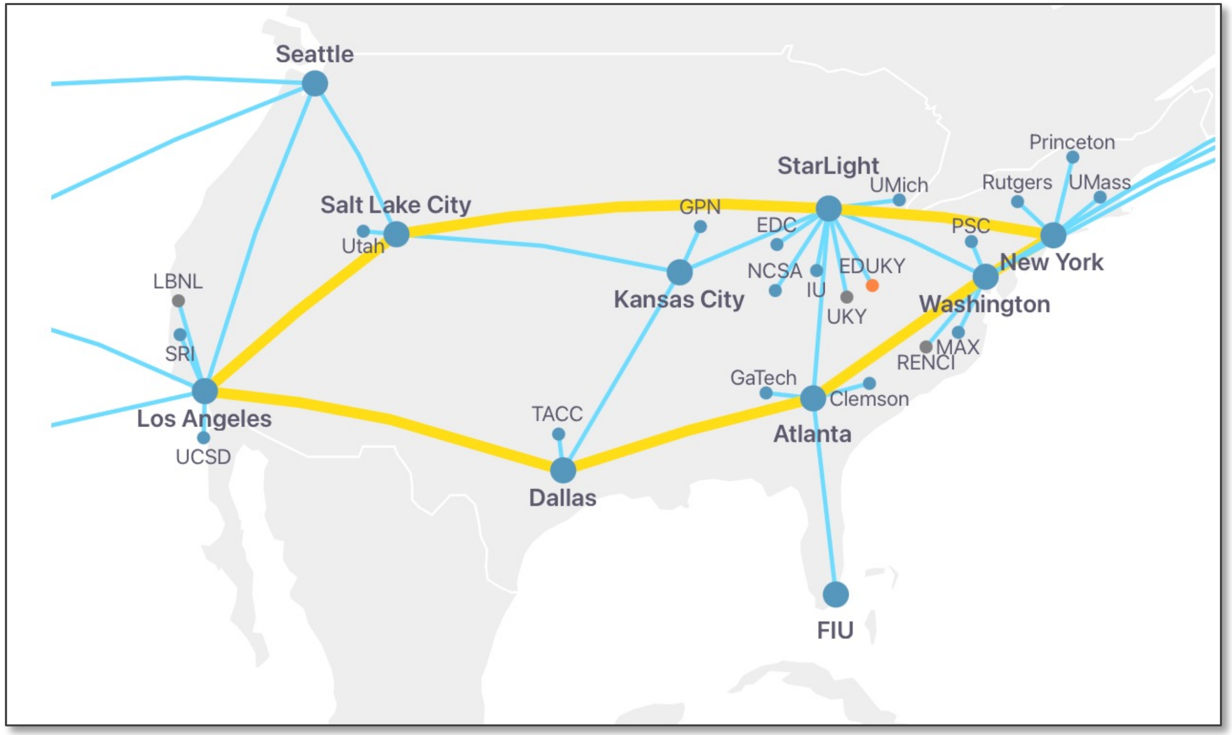
Cybertraining on SmartNICs/DPUs and End-host Stack

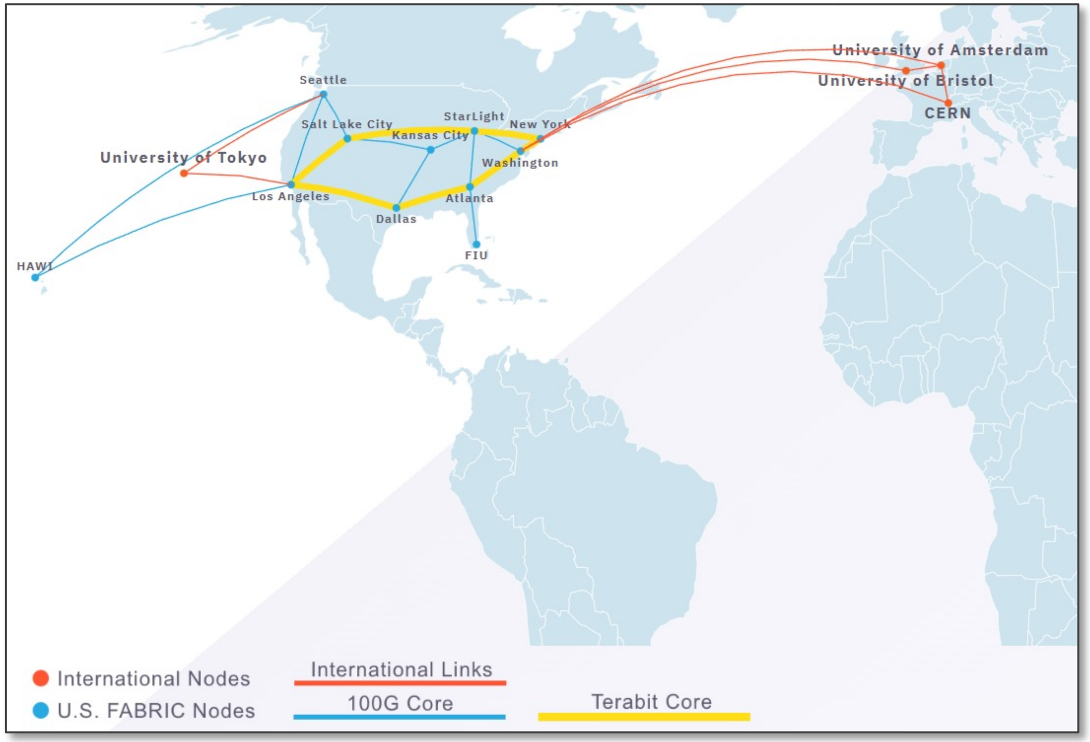
- Goal 1: Promote and facilitate the broader adoption of SmartNICs among CI professionals, CI contributors, and network owners
  - Obj. 1: Install the SmartNICs in FABRIC and the academic cloud.
  - Obj. 2: Develop vLabs on SmartNICs and related technologies.
  - Obj. 3: Disseminate the vLabs on FABRIC and the academic cloud for CI professionals

- Goal 2: Incorporate SmartNICs vLabs into educational curricula and instructional resources at various academic levels
  - Obj. 1: Facilitate adoption of vLabs at academic institutions by embedding access to FABRIC and the cloud into LMS systems and by developing a catalog system.
  - Obj. 2: Provide online and face-to-face workshops for instructors

FABRIC Testbed

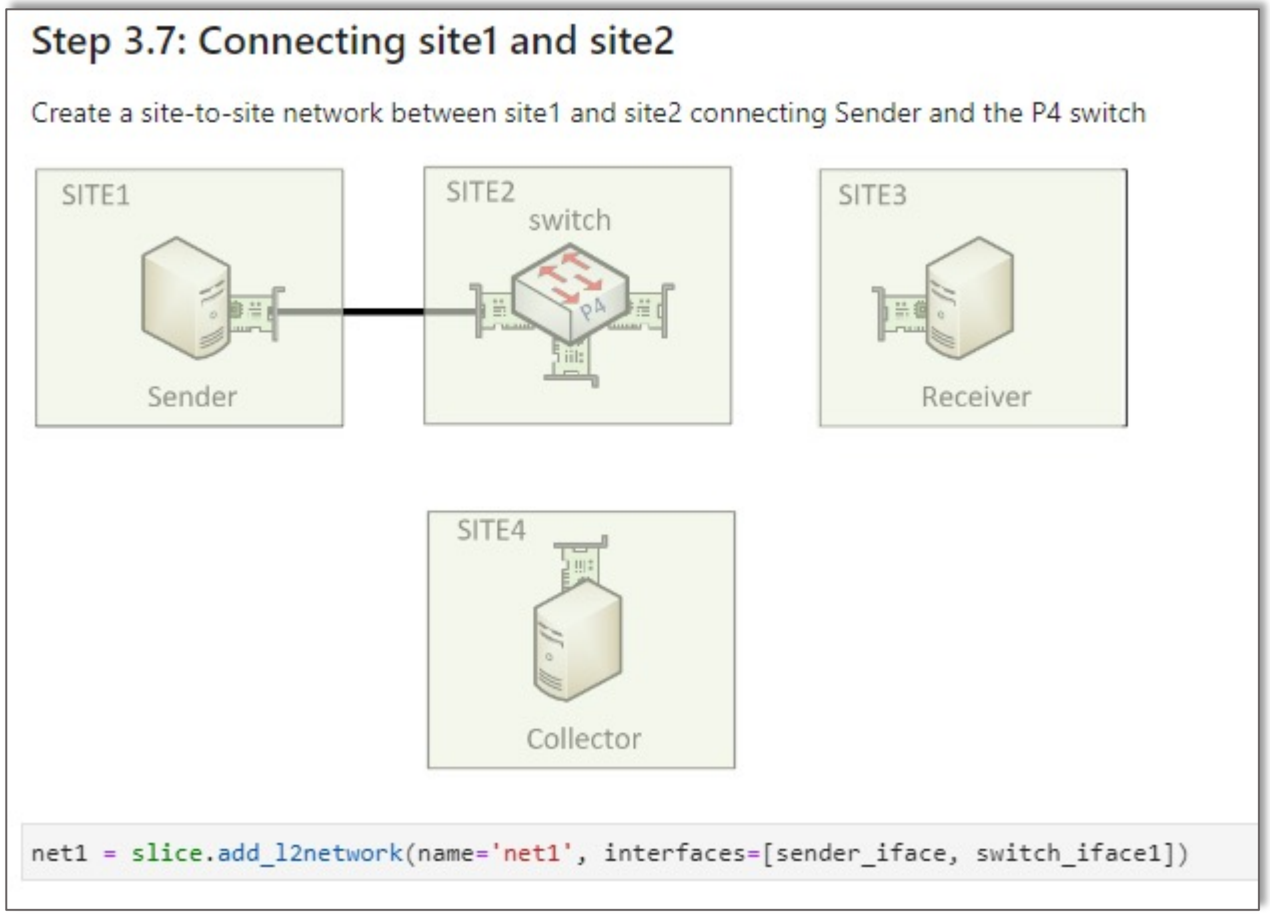
- FABRIC is an NSF-funded international infrastructure for at-scale experimentation and research
- Areas include networking, cyber, distributed computing, storage, 5G, ML, etc.
- Equipment is located at commercial collocation spaces, U.S. national labs, and campuses – 29 FABRIC sites





Cybertraining on FABRIC

- FABRIC is a real network with physical propagation delays and high-speed links
- With its integrated JupyterHub, it can be ideal for cybertraining:
  - P4 programmable switches/NICs
  - High-speed networks (SDMZ)
  - PerfSONAR
  - Measurement and telemetry
  - Cybersecurity (Zeek, Suricata, etc.)
  - Etc.



Labs on P4 Programmable Data Planes over FABRIC

- The following labs have been developed as part of the first project[1]:
  - Lab 1 – Preparing the Environment
  - Lab 2 – P4 Program Building Blocks
  - Lab 3 – Parser Implementation
  - Lab 4 – Introduction to Match-action Tables
  - Lab 5 – Populating Match-action Tables from the Control Plane
  - Lab 6 – Checksum Calculation and Packet Deparsing
- The labs are available to FABRIC users

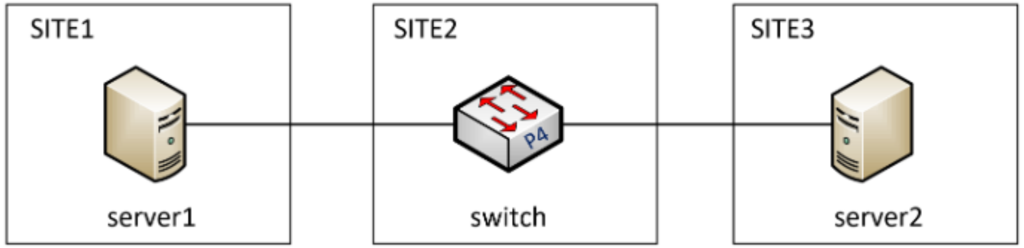
[1] https://learn.fabric-testbed.net/knowledge-base/p4-programmable-data-plane-switches-bmv2-over-fabric/

Labs on P4 Programmable Data Planes over FABRIC

Virtual Labs on P4 Programmable Data Plane Switches (BMv2)

The labs provide a hands-on experience on P4 programmable data plane switches using the Behavioral Model version 2 (BMv2) software switch. The lab series explains topics that include parsing, match-action tables, checksum verification, and others.

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

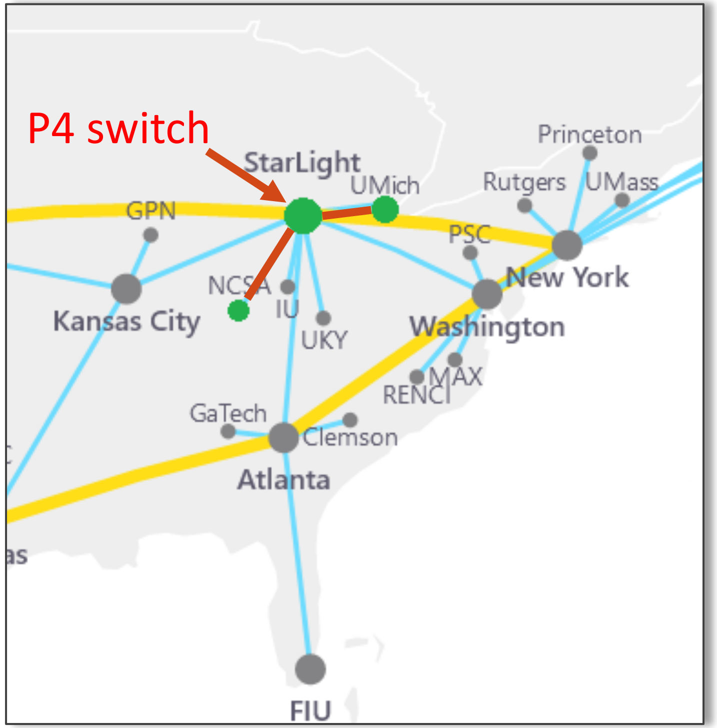


Labs:

- 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.

Throughput Test over BMv2

- BMv2 software switch is running on StarLight



Step 8.4: Starting iPerf3 on server2

```
[107]: server2.execute_thread('iperf3 -s')

[107]: <Future at 0x7fee04ab7b50 state=running>
```

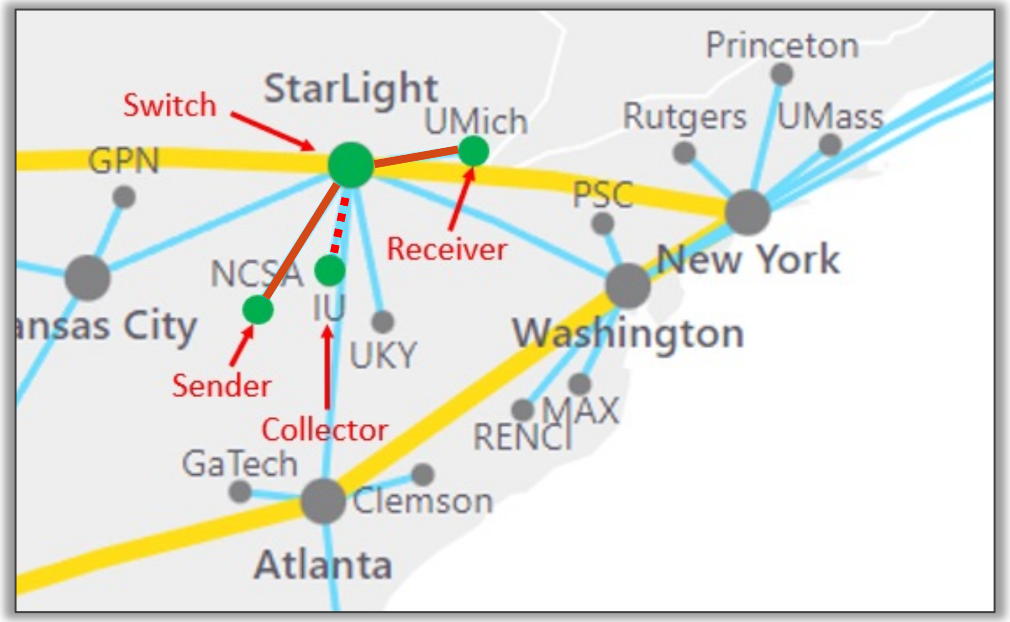
Step 8.5: Starting iPerf3 client on server1

```
[114]: server1.execute('iperf3 -c 192.168.2.10 -P 2')

Connecting to host 192.168.2.10, port 5201
[  5] local 192.168.1.10 port 57904 connected to 192.168.2.10 port 5201
[  7] local 192.168.1.10 port 57908 connected to 192.168.2.10 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  64.5 MBytes   541 Mbits/sec  427   1.25 MBytes
[  7]   0.00-1.00   sec  72.4 MBytes   607 Mbits/sec  1050   1.47 MBytes
[SUM]   0.00-1.00   sec   137 MBytes  1.15 Gbits/sec  1477
- - - - - - - - - - - - - - - - - - - - - - - - -
[  5]   1.00-2.00   sec  60.0 MBytes   503 Mbits/sec   31    952 KBytes
[  7]   1.00-2.00   sec  70.0 MBytes   587 Mbits/sec   47   1.10 MBytes
[SUM]   1.00-2.00   sec   130 MBytes  1.09 Gbits/sec   78
- - - - - - - - - - - - - - - - - - - - - - - - -
[  5]   2.00-3.00   sec  56.2 MBytes   472 Mbits/sec    0   1024 KBytes
[  7]   2.00-3.00   sec  66.2 MBytes   556 Mbits/sec    0   1.18 MBytes
[SUM]   2.00-3.00   sec   122 MBytes  1.03 Gbits/sec    0
- - - - - - - - - - - - - - - - - - - - - - - - -
```

Queue Measurement Lab

- BMv2 software switch is running on StarLight
- Microseconds granularity

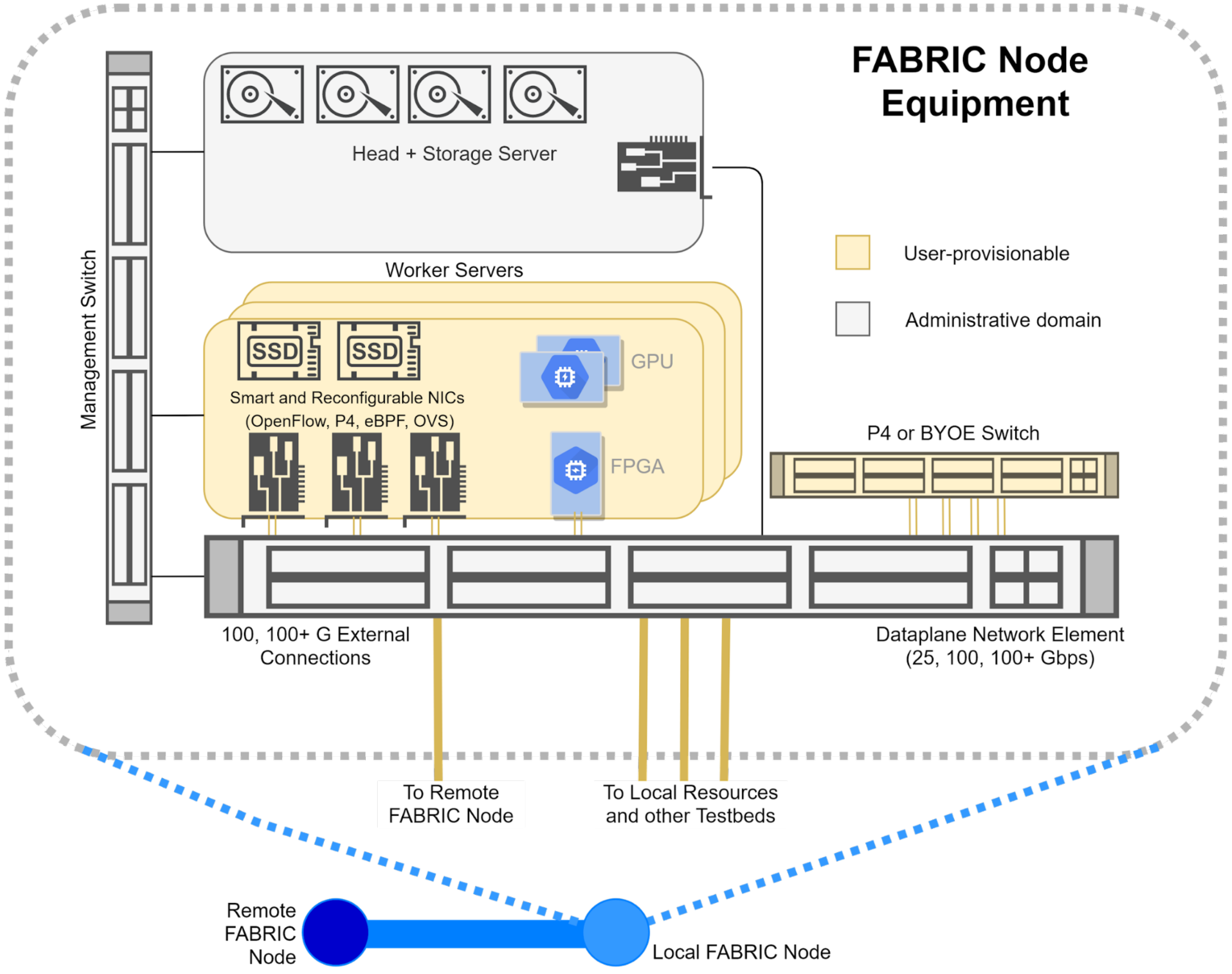


Queue Occupancy

```
384.21 us
380.21 us
395.49 us
389.79 us
383.94 us
386.61 us
375.84 us
5375.27 us
62615.75 us
132152.93 us
```

FABRIC Node

- FABRIC nodes have the following:
  - FPGA-based SmartNICs (P4-programmable)
  - P4 Tofino switches
  - Offload NICs (NVIDIA ConnectX, 100Gbps)
- The project will add new SmartNICs to FABRIC
  - NVIDIA/Mellanox Bluefield-2/3
  - Intel IPU
  - P4-programmable Xilinx FPGA
  - Maybe others, if available.

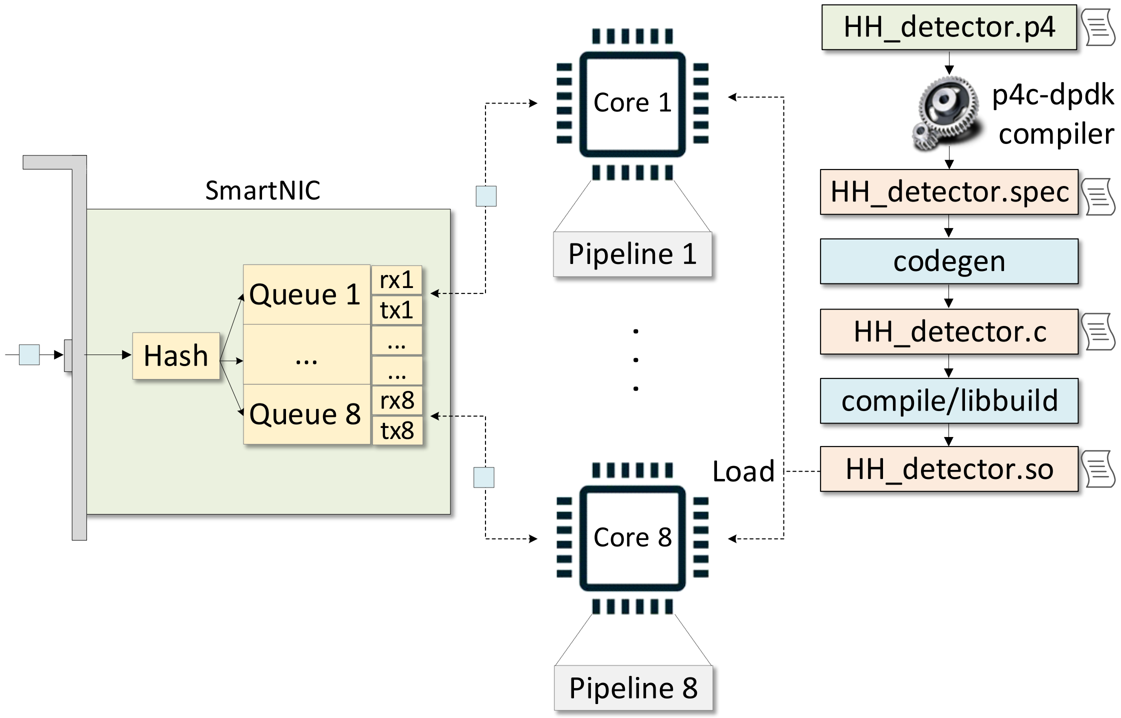


P4-DPDK Lab Library on the Academic Cloud

- The following labs have been developed for P4-DPDK:
  - Lab 1 – Introduction to P4-DPDK
  - Lab 2 – P4 Program Building Blocks with the PNA Architecture
  - Lab 3 – Parser Implementation
  - Lab 4 – Introduction to Match-action Tables (Part 1)
  - Lab 5 – Introduction to Match-action Tables (Part 2)
  - Lab 6 – Populating Match-action Tables from the Control Plane
  - Lab 7 – Checksum Calculation and Packet Deparsing
- Currently, the labs use a virtual NIC for scalability purposes

P4-DPDK Lab Library on FABRIC

- FABRIC has 15,000+ NICs (ConnectX-5/6)
- These NICs support DPDK and can be used for the libraries
- Experiments with P4-DPDK show that it is possible to achieve 100Gbps



Other Libraries to be Developed

- Advanced applications and Stateful Elements in P4-DPDK
- Implementing cybersecurity defenses using P4-DPDK
- Fine-grained measurements and telemetry using P4-DPDK
- Introduction to P4-TC
- Introduction to P4-eBPF
- Libraries related to:
  - Infrastructure Programmer Development Kit (IPDK)
  - Open Programmable Infrastructure (OPI)
  - SONIC-DASH

Other Libraries to be Developed

- Advanced applications and Stateful Elements in P4-DPDK
- Implementing cybersecurity defenses using P4-DPDK
- Fine-grained measurements and telemetry using P4-DPDK
- Introduction to P4-TC
- Introduction to P4-eBPF
- Libraries related to:
  - Infrastructure Programmer Development Kit (IPDK)
  - Open Programmable Infrastructure (OPI)
  - SONIC-DASH

We need your help! What libraries are relevant to the P4 community?
What labs would you like to see? labs feedbacks/suggestions, etc.

Access to the vLabs – Academic Cloud (USC)

- The lab libraries and their PDF contents can be found at:

https://tinyurl.com/cilab-cybertraining

- Platform link:

https://netlab.cec.sc.edu

- To obtain an account, send an email to choueiri@email.sc.edu
  - ➢ Your full name, institution and role, email address
  - ➢ Lab libraries of interest
- User guide on how to use the platform:

https://tinyurl.com/cilab-userguide

Access to the vLabs – FABRIC

- Sign-up for an account on FABRIC

https://tinyurl.com/fabric-account

- Create or enroll into a FABRIC project:

https://tinyurl.com/fabric-project

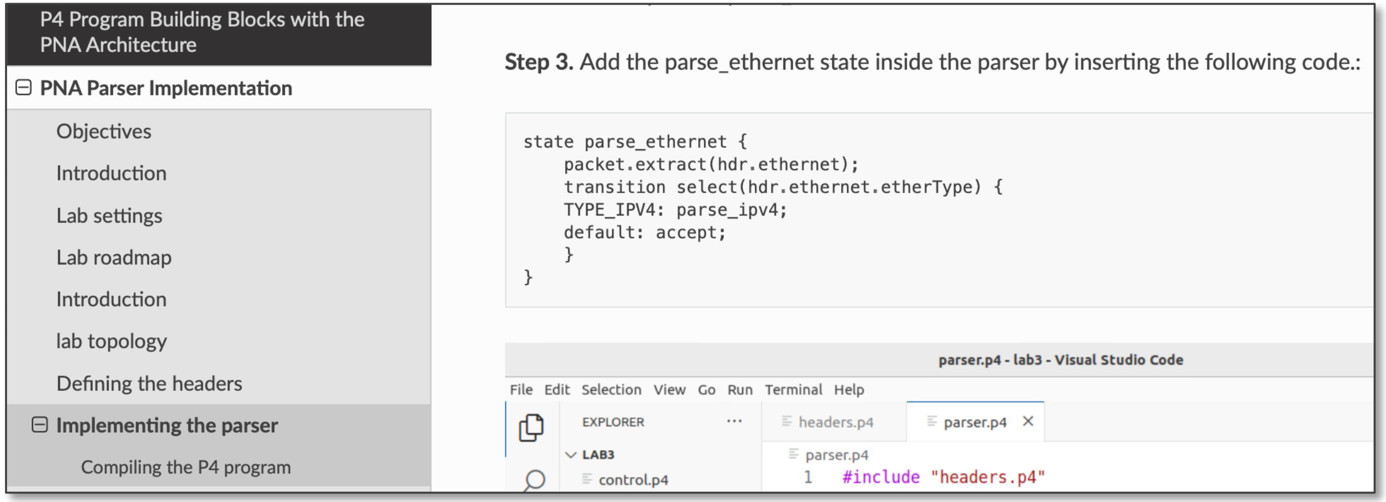
- If you are a professor, create a new project and assign your students to it
- Once you login into JupyterHub, you will see the P4 labs under the FABRIC-examples provided for you
- FABRIC can be used for teaching a class too!

https://tinyurl.com/fabric-teaching

Access the Labs via readthedocs

- Upcoming lab libraries will have a readthedocs website
- Contributions to enhance the libraires are welcomed (GitHub pull requests)
- Useful if you like to install the Virtual Machine on your device without access to the academic cloud or FABRIC
- Example library: P4-DPDK

https://tinyurl.com/cilab-readthedocs



Contact Information

UNIVERSITY OF SOUTH CAROLINA

- Elie Kfoury
  ekfoury@email.sc.edu

- Jorge Crichigno
  jcrichigno@cec.sc.edu

- CI Lab Website
  https://research.cec.sc.edu/cyberinfra/