



Lab 3: Match-Action Tables in P4

Elie Kfoury*, Jennifer Kim[^], Jorge Crichigno*

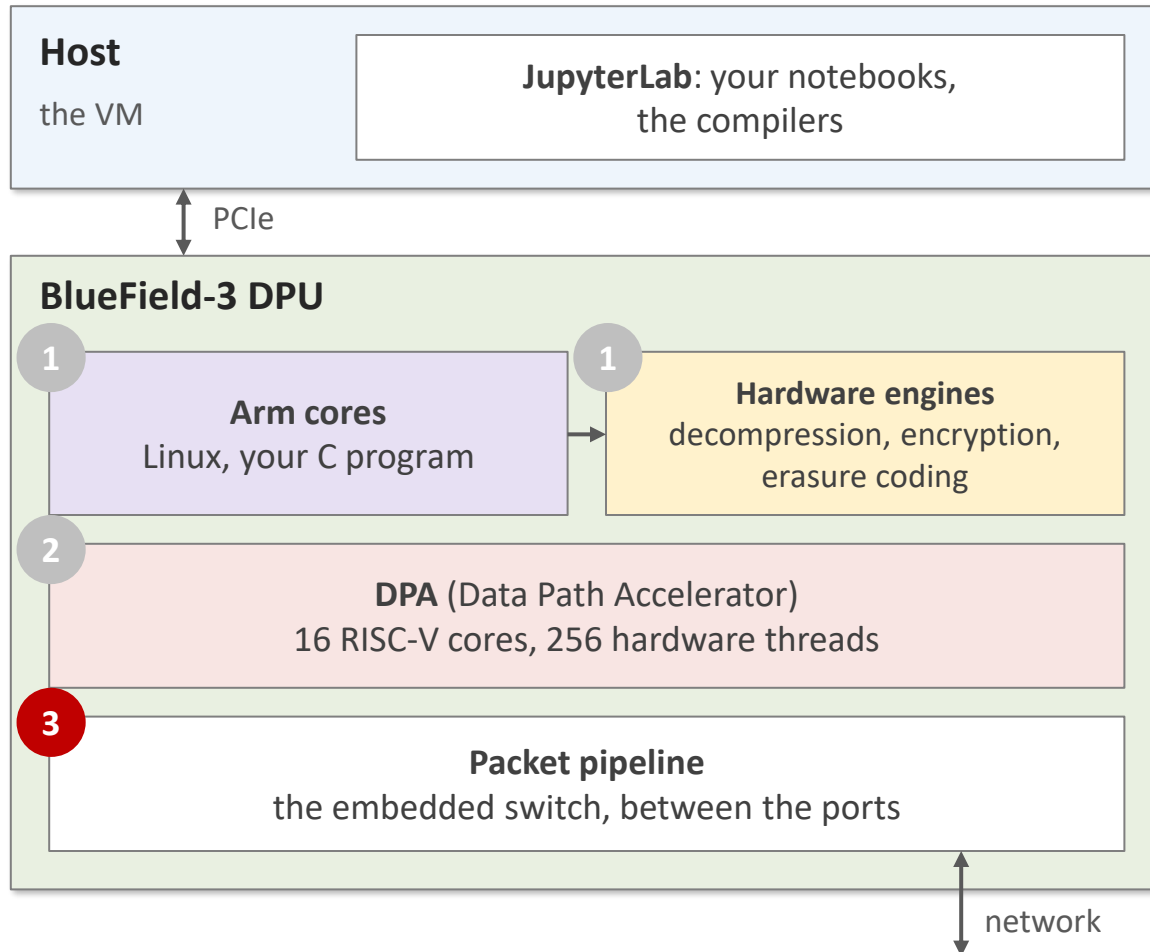
* College of Engineering and Computing, University of South Carolina
[^] Internet2

Internet2 Technology Exchange (TechEx)

October 26, 2026
Minneapolis, Minnesota

Three Labs, Three Parts of the DPU

- This lab: **Lab 3: Match-Action Tables in P4**



1 Lab 1: Hardware decompression

A C program on the Arm cores hands the work to the decompression engine

DOCA Compress · doca-highlevel Lab 1

2 Lab 2: Receiving packets on the DPA

The network card wakes a DPA thread for every packet that arrives

FlexIO · DPA Lab 5

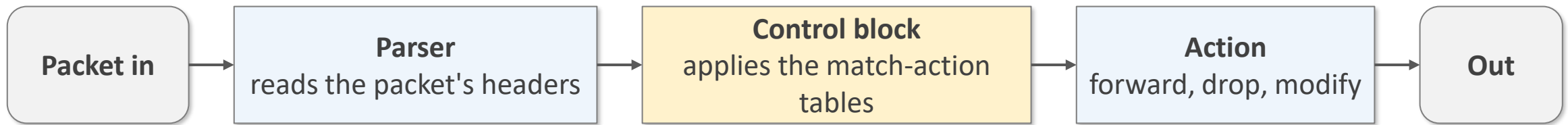
3 Lab 3: Match-action tables in P4

A P4 program becomes the pipeline that forwards or drops each packet in hardware

DOCA Pipeline Language · P4 Lab 1

What Is a Match-Action Pipeline?

- Like a switch's MAC table or a router's forwarding table, but you define the tables
- A P4 program describes the pipeline; on the DPU it runs in hardware, for every packet



A table, for example

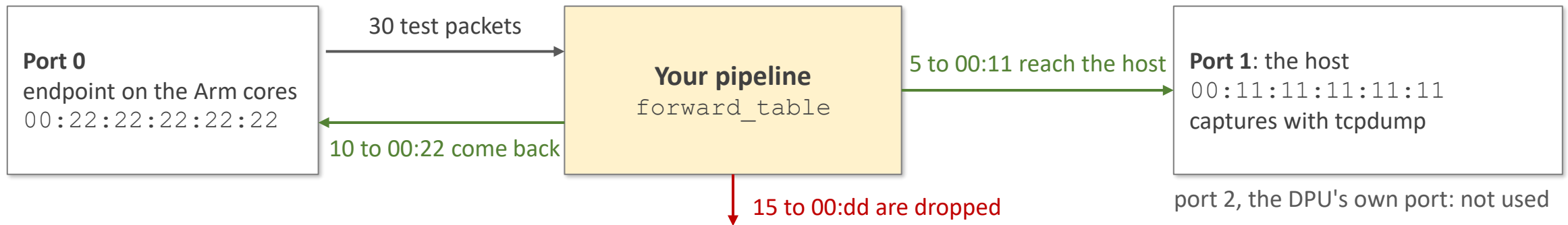
Key: destination MAC	Action	Action data
00:11:11:11:11:11	forward	port 1
00:dd:dd:dd:dd:dd	drop	—
<i>no match</i>	<i>default action</i>	—

Key	the header fields the table looks up
Action	what to do on a match: forward, drop, modify
Action data	the action's parameters, such as the port
Default action	what happens when nothing matches

On the BlueField: **DPL**, NVIDIA's P4, compiled into the hardware pipeline. Entries are fixed in the program, or added at run time through P4Runtime.

Lab 3: The Scenario

- A static layer-2 forwarding table between three ports of the DPU's pipeline
- Test packets from port 0 show what the table does with each destination MAC



What you do

- 1 Write the table in the P4 program
- 2 Compile it on the host
- 3 Take your turn: start the runtime, load the pipeline
- 4 Send the test packets, capture on the host
- 5 Unload the pipeline: your turn ends

What to expect

```
Welcome to the IPython shell for P4Runtime
the pipeline is loaded

packets received by the host: 5
only the packets for the host's MAC

192.168.0.1 > 192.168.0.2
their IP addresses come out swapped: the forward action's work
```

The P4 Program

- One table: the key is the destination MAC; five entries, and drop when nothing matches
- The control block applies the table to every packet that enters the pipeline

forward_table: exact match on the destination MAC

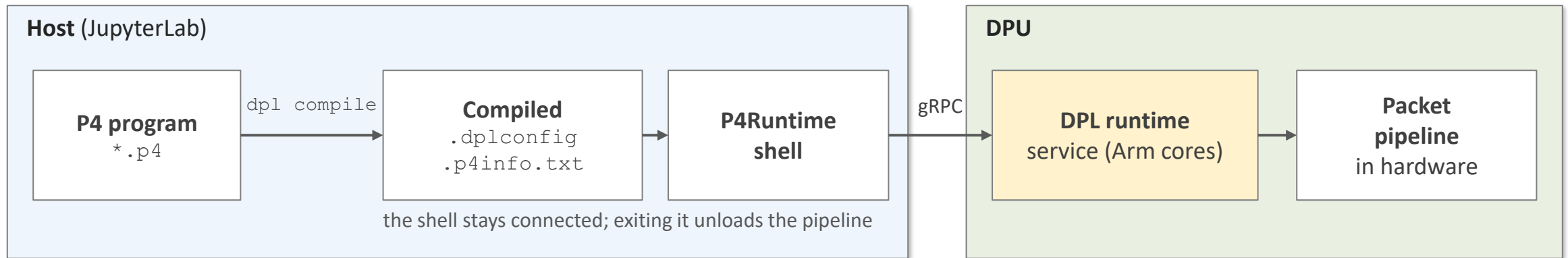
Key: destination MAC	Action	Action data
00:11:11:11:11:11	forward	port 1 (host)
00:22:22:22:22:22	forward	port 0 (sender)
00:dd:dd:dd:dd:dd	drop	—
00:aa:aa:aa:aa:aa	NoAction	—
00:bb:bb:bb:bb:bb	NoAction	—
<i>no match</i>	<i>drop (default)</i>	—

NoAction leaves the default action in place, so the 00:aa and 00:bb packets are dropped too

```
table forward_table {  
    key = { headers.ethernet.dst_addr : exact; }  
    actions = { drop; forward; NoAction; }  
    default_action = drop;  
    const entries = {  
        (48w0x001111111111) : forward(1);  
        (48w0x002222222222) : forward(0);  
        (48w0x00dddddddddd) : drop();  
        (48w0x00aaaaaaaaaa) : NoAction();  
        (48w0x00bbbbbbbbbb) : NoAction();  
    }  
}  
apply { forward_table.apply(); }
```

From Code to a Running Pipeline

- You compile on the host; a small controller loads the pipeline into the DPU's runtime service



The cells, in order

- 1 Compile** `dpl compile --target doca static_l2_forwarding.p4`
- 2 Your turn** `dpu-turn begin 10` and start the runtime
- 3 Load** the P4Runtime shell, kept running in the background
- 4 Test** send the packets, read the counters, capture on the host
- 5 Unload** end the shell, then `dpu-turn end`

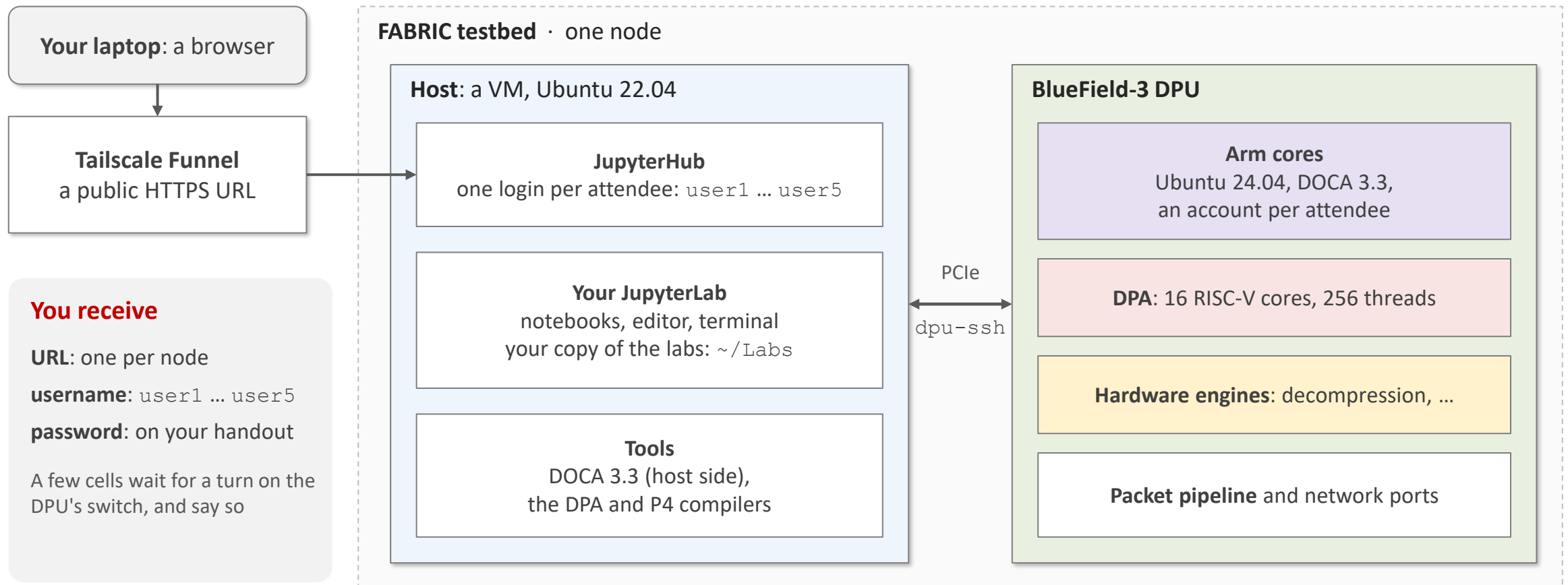
Your turn on the switch

The runtime holds one pipeline at a time, and the attendees of a node share it.

Steps 2 to 5 run in your turn, at most 10 minutes; if someone else has the switch, the cell says so and waits.

The Lab Environment

- Nothing to install: a web browser, a URL, a username and a password
- Each node is a FABRIC VM with its own BlueField-3 DPU, shared by up to five attendees



Working in JupyterLab

- Each lab is a notebook: read the text, then run the cells in order (Shift+Enter)
- A cell starting with `!` runs on the host; `dpu-ssh` '...' in it runs on the DPU's Arm cores

JupyterLab

Labs/
 doca/
 doca-highlevel/
 lib/
 Lab1/
 Lab_1_....ipynb
 lab_files/
 figs/
 Lab2/ ...
 dpa/
 p4/

1

Step 4: Running the program
Run the program on the DPU.

[5]:

```
!dpu-ssh 'cd ~/doca-highlevel/Lab1/lab_files  
         && ./build/decompress'
```

...
Hardware decompression: 64 chunks, 64.0 MiB
 in ... ms = ... MB/s
LAB 1 finished successfully

2

Step 5: Comparing with the Arm cores
bench_arm.py decompresses the same 64 chunks with zlib ...

1

File browser
your own copy of the labs; double-click a file to open it

2

Instructions
what each step does, with figures; read it before running the cell

3

Code cell
Shift+Enter runs it; it may take a few seconds

4

Output
appears under the cell; the text after it says what you should see

To write code: open the file, type the lines shown, save (Ctrl+S), run the build cell

8

The Labs: Four Families

- 17 labs; each has a notebook, the files you complete (`lab_files`) and figures

DOCA, step by step

`doca/`

a DOCA program, line by line

- 1 Initializing and configuring the DOCA environment
- 2 Device subsystem
- 3 Memory management
- 4 Progress engine and execution model
- 5 Accelerator invocation

DOCA, higher level

`doca-highlevel/`

only the lines that carry the idea

- 1 **Hardware decompression (DOCA Compress)** 1
- 2 Hardware encryption (AES-GCM)
- 3 Erasure coding
- 4 A switch with DOCA Flow
- 5 Packets through the Arm cores
- 6 Fast path and slow path

DPA

`dpa/`

built and run on the host

- 1 Initializing and configuring the DPA environment
- 2 Sharing data between host and DPA
- 3 Running multiple DPA threads
- 4 Waking a DPA thread from the network card
- 5 **Receiving live packets on the DPA** 2

P4 (DPL)

`p4/`

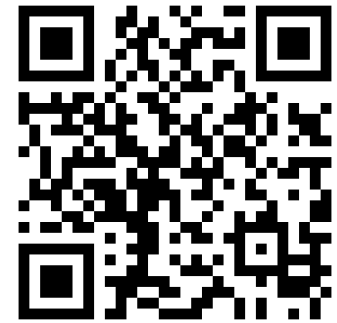
runs in the packet pipeline

- 1 **Introduction to match-action tables** 3

Your Lab Node: Links and Credentials

- Pick a node and a username, one person per username, and keep them for all three labs

Node01 FABRIC site DALL



is.gd/internet2techex_node01

`https://dpu-tutorial-10091431-r2-node01.tail9fd940.ts.net/`

usernames: user1 ... user5

Node02 FABRIC site TOKY



is.gd/internet2techex_node02

`https://dpu-tutorial-10091431-r2-node02.tail9fd940.ts.net/`

usernames: user1 ... user5

Node03 FABRIC site MASS



is.gd/internet2techex_node03

`https://dpu-tutorial-10091431-r2-node03.tail9fd940.ts.net/`

usernames: user1 ... user5

Password: smartnic_tutorial the same for every node and username