TECHNISCHE UNIVERSITÄT DRESDEN

CeTI Centre for Tactile Internet with Human-in-the-Loop

Haptic Communication Systems, Faculty of Electrical and Computer Engineering
Center for Tactile Internet with Human-in-the-Loop (CeTI), TU Dresden, Germany

P4sim: Protocol-Independent Packet Processors in ns-3

Speaker: Ma Mingyu

P4sim

P4sim: https://github.com/HapCommSys/p4sim
Artifact: https://github.com/HapCommSys/p4sim-artifact-icns3

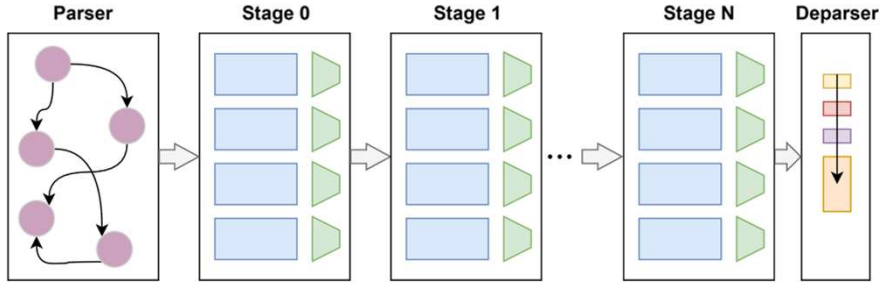
Motivation: P4 Language and Programmable Data Plane

Programming Protocol-independent Packet Processors (P4) is a domain-specific language for programming protocol independent packet forwarding.

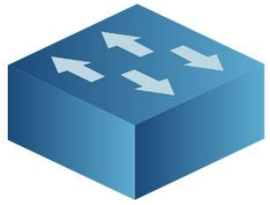
```
/* -*- P4_16 -*- */
#include <core.p4>
#include <arch_specific.p4>

control Ingress() {

    Customized
    Packet Processing
    Logic

}
...

Pipeline (Parser(), Ingress(),
Egress() ... ) main;
```

P4 Code

- Enables rapid adaptation to new protocols
- Provides fine-grained control over network behavior
- Improves flexibility and scalability



P4 Programmable Data Plane Target (e.g., Bmv2*, Tofino)



Build a "new switch" in "compilation time"

** Bmv2: Behavioral Model version 2*

Motivation: P4 Language and Programmable Data Plane

In the last past 5 years, P4 has received widespread attention in network community.

1

*

P4 Accelerates the Evolution of Networks:

a. **In-network computing:** NetCache, SwitchML …
b. **Congestion control:** HPCC, BFC …
c. **Load balancing:** HULA …
d. **Network measurement:** In-band Telemetry, Sketches …

P4 makes Network Better!

2

*

ns-3 is a discrete-event network simulator widely used for research and education, providing a flexible and modular platform for modeling internet protocols and network systems. It enables realistic simulation of **wired and wireless networks**, supporting detailed performance evaluation and reproducible experimentation.

—What if we bring P4 into ns-3?

** Figures generated by ChatGPT Sora*

Content

Related Work

Programmable switch platforms & Our Goal

P4 Platform	Tofino	Net-FPGA / Smart NIC	T4P4S	Bmv2	P4 & ns-3
Best for	Production-level deployment	Hardware acceleration	High-performance packet processing (DPDK)	Learning & debugging (Single Switch)	Efficient, scalable, and reproducible network simulation

Table: Comparison of P4 platforms: **hardware-based**, **software-based emulation**, and **simulation**.

Our goal is to provide a more practical, complete, and accurate **P4 simulation framework** by integrating P4 behavior model into ns-3—enabling scalable, cost-effective, and reproducible protocol design exploration.

Related Work

P4 Integration Attempts

The are three related work integrate P4 into ns-3:

Modules	Description	Limitations
NetDevices [6]	Generate C code using T4P4S and compile it alongside the ns-3 simulation code	• Manual adjustments and build code, low modularity and traceability
ns3-bmv2 [7]	Partially Integrates the BMv2 into the *traffic control module*	• Only single-path simulation • No routing and time-aware modeling • Not event-driven
NS4 [8, 9]	Partially Integrates the BMv2 into the ***Bridge Netdevice***	• Lacks queuing and time-aware modeling • Not event-driven • Unable to implement protocol-independent features

Table: Comparison of the modules simulating P4 in ns-3.

Overview of P4sim Design

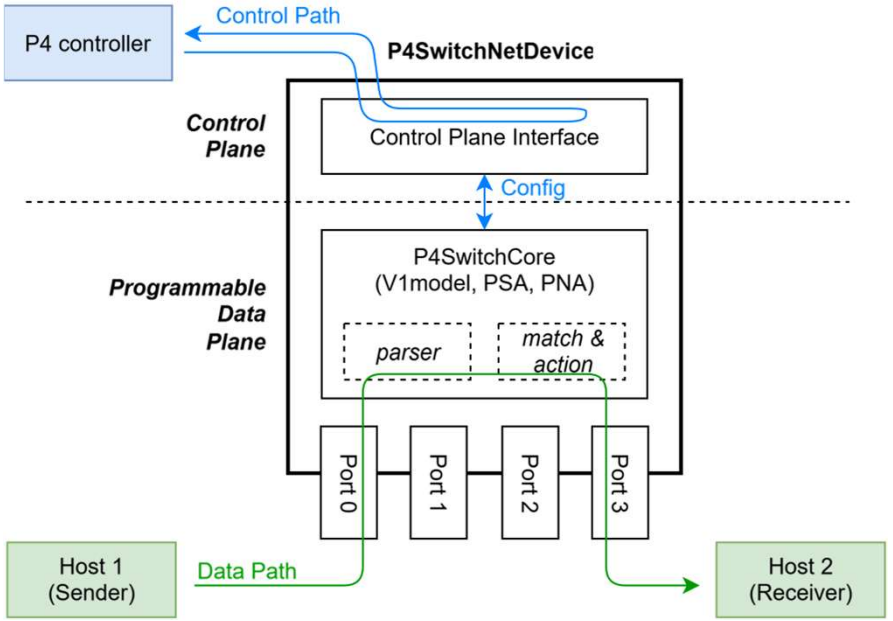
- **High-performance Event Scheduling**
  - A. Modeling P4 Switch
  - B. Queue Scheduling & Virtual Time Alignment
- **Extended P4 Support**
  - C. Supporting Modern P4 Architectures: V1model, PNA, PSA
- **Protocol-independent Design**
  - D. Enabling P4 Custom Header
- **Seamless Integration with ns-3**
  - Compatible with existing ns-3 modules
  - Tested with both Waf and CMake build systems

P4sim Design

A: Modeling P4 Switching

P4 switch design based on bridge net device, include the external bmv2's behavior model, and extend with **External Modules** like OpenFlow switch [2]:

- ❑ Hardware management part (ns-3 configure):
  - *Port management*
  - *P4 switch architecture*
  - *P4 related configuration (P4 script, flow table entries)*
  - *Buffer configuration (size & number)*
  - *Switch rate configuration with pps*
  - *…*
- ❑ Programmable switching part (P4 configure):
  - *P4 script → Initialization P4SwitchCore*
  - *Flow table entries → configure P4SwitchCore*
  - *…*



P4sim Design

A: Modeling P4 Switching

Buffers:

- Input buffer, Queue buffer

Packets metadata:

- standard_metadata defined by P4 scripts (User)
- intrinsic_metadata defined by P4 arch
- queueing_metadata defined by P4 arch
- Error information (for example parser error)

Packets processing actions register (V1model):

bit63 ..... Bit0 [Address Shift & Mask]
| ResubmitFlag (16) | LF_FieldList (16) | Clone_FieldList (16) | Clone_SessionID (16) |
| (unused) | NS_Address (16) | NS_Protocol (16) | RecirculateFlag (16) |
| User allocated in P4 scripts ..... |

External actions Add-on: hash, random, meter, counter, checksum

P4sim Design

B: Queue Scheduling & Virtual Time Alignment

Enqueue:

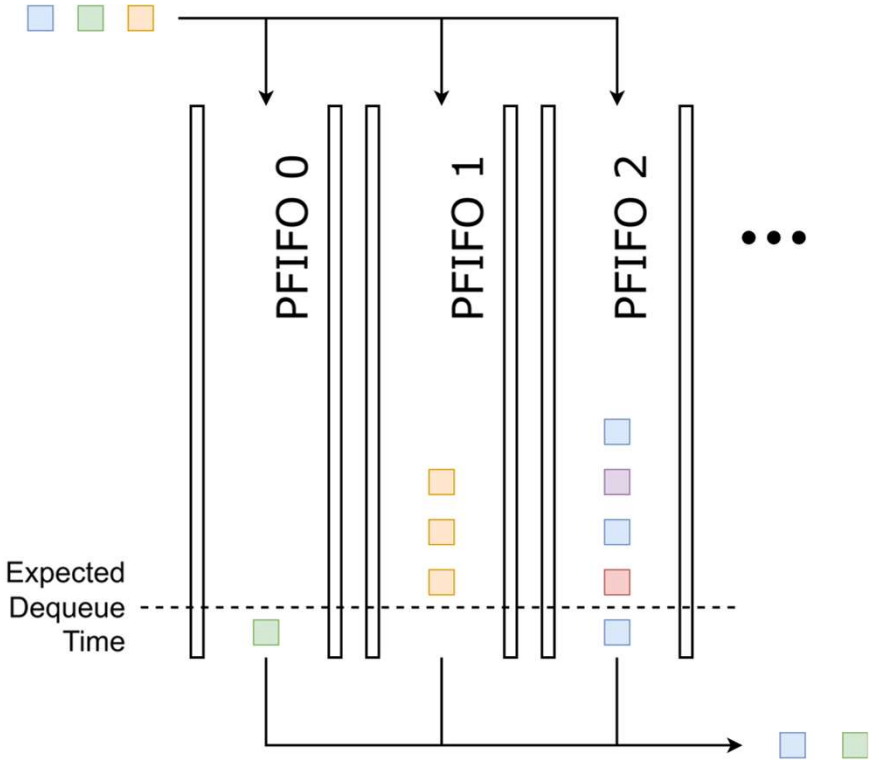
- Based on type of service and priority allocation
- Set the expected dequeue time for enqueued pkts

Dequeue:

- Packets always dequeued from PFIFO 0 first, then PFIFO 1 and lastly PFIFO 2
- Packets are dequeued at their expected dequeue time if resources are available

Dequeue Event:

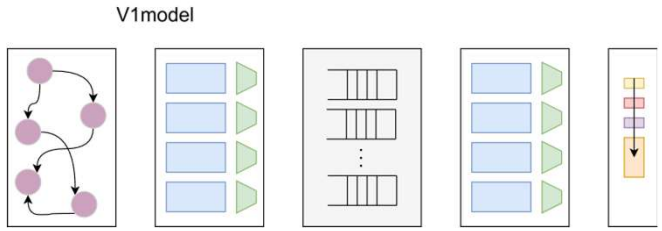
- Polling mode
- Based on switch processing rate set the events



P4sim Design

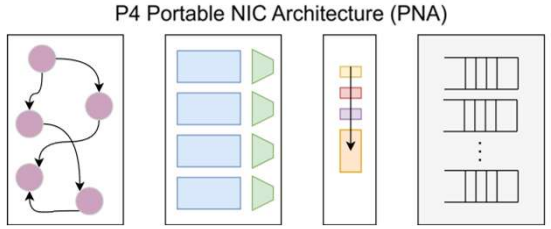
C: Supporting Modern P4 Architectures: V1model, PNA, PSA

V1model



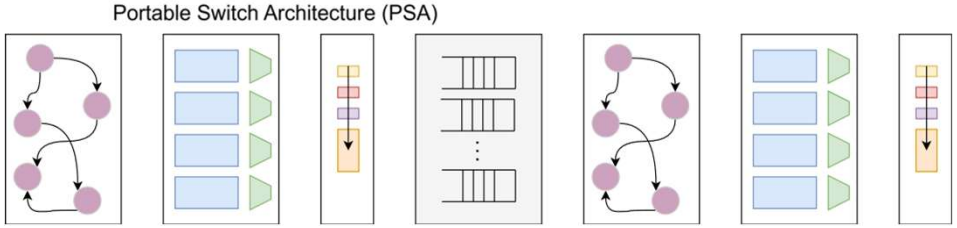
Provides a standardized description of BMv2's forwarding pipeline.

PNA



** NetFPGA SUME [10]*

PSA



** tomahawk-5 [11]*

P4sim Design

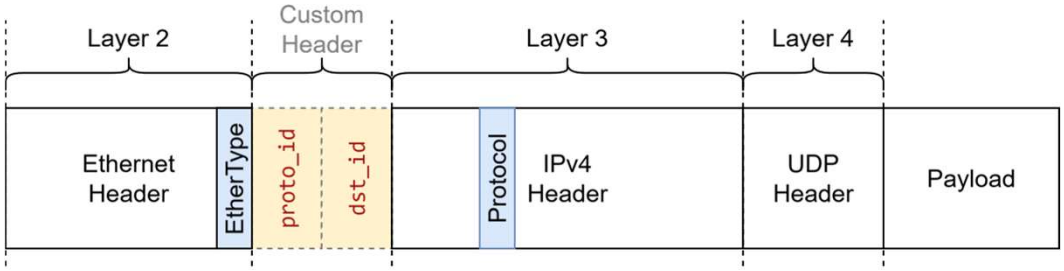
D: Protocol-independent design: Enabling P4 Custom Header

Protocol-independent data plane:

- ❑ P4 enables protocol-independent processing on **P4 switches**
- ❑ But both **sender & receiver side** in ns-3 must support the custom header format

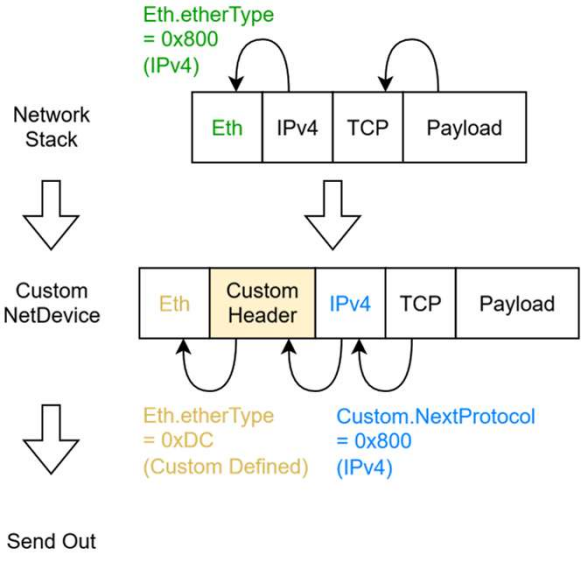
How to design a custom header? → Modify the Header!

- ❑ Define Header Layer (2, 3, 4)
- ❑ Select Placement Mode
  - ADD_BEFORE | ADD_AFTER | REPLACE
- ❑ Include Header Fields
  - *proto_id* → identifies next header type
  - Other header field

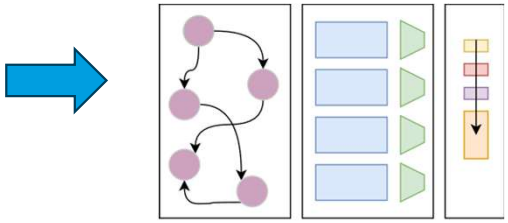


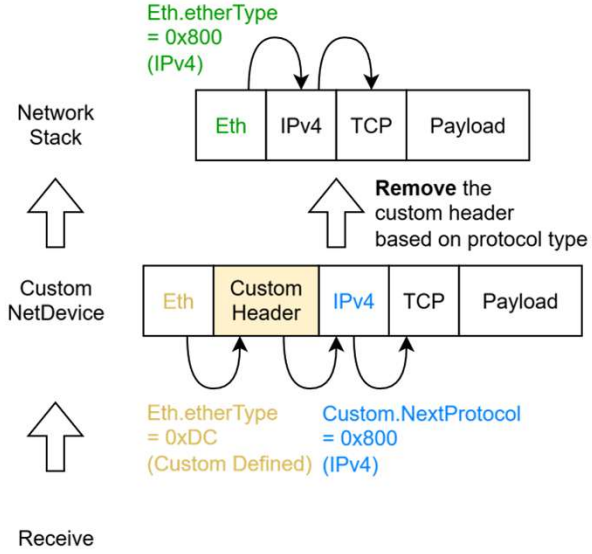
P4sim Design

D: Protocol-independent design: Enabling P4 Custom Header



In ns-3, upper-layer applications (e.g., OnOffApplication) remain unaware of the custom header, allowing maximum reuse of existing applications.





Sender Side
(Add the Custom Header)

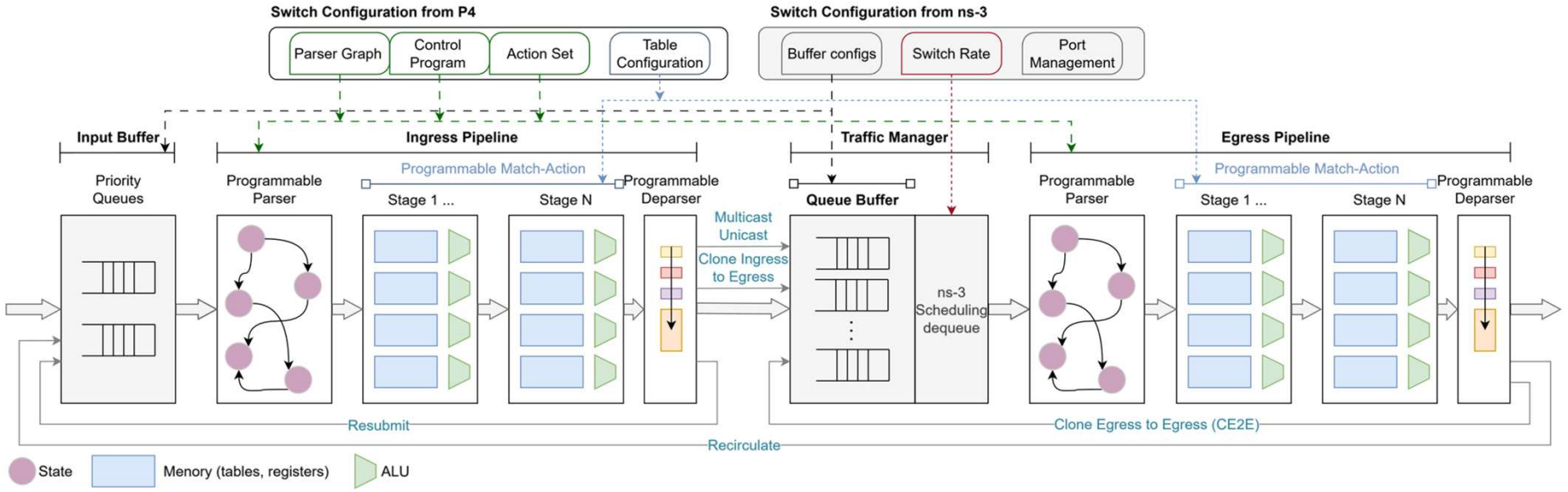
P4 Switch Processing
(with Custom Header)

Receive Side
(Remove the Custom Header)

P4sim Implementation

The Internal Structure of the P4 Switch Model (PSA) in ns-3

PSA model pkt processing pipeline + primitive actions (like Resubmit) + ns-3 configuration + P4 configuration



Two key aspects: the buffer management and scheduling mechanisms configured within ns-3, and the packet processing pipeline configured via the P4 program.

Similarly, we also deployed V1model and PNA.

Overview of Evaluation and Metrics

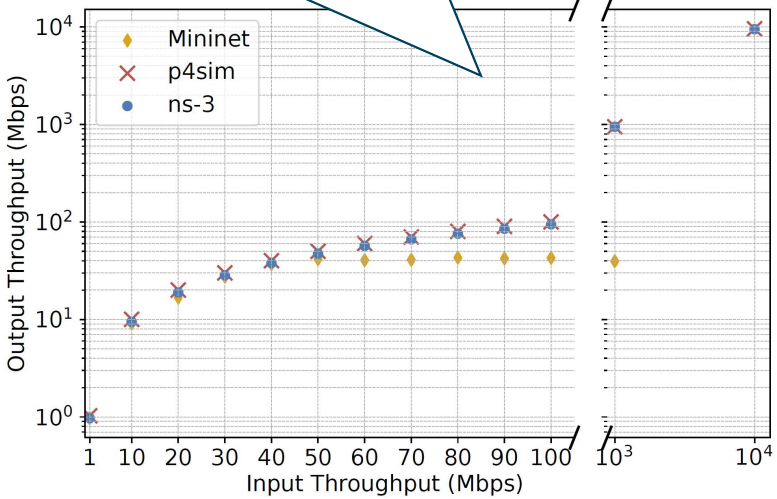
- **Detailed Goals**
  - High-performance event scheduling
  - Extended P4 support
  - Protocol-independent design
  - Seamless integration with ns-3
- **Scope**
  - One evaluation for performance and design effectiveness
  - Three use cases to show extended p4 support and protocol-independent design
- **Evaluation Metrics**
  - Forwarding throughput
  - Simulation execution time

Evaluation

Performance and Runtime Overhead in Ipv4 forwarding scenario

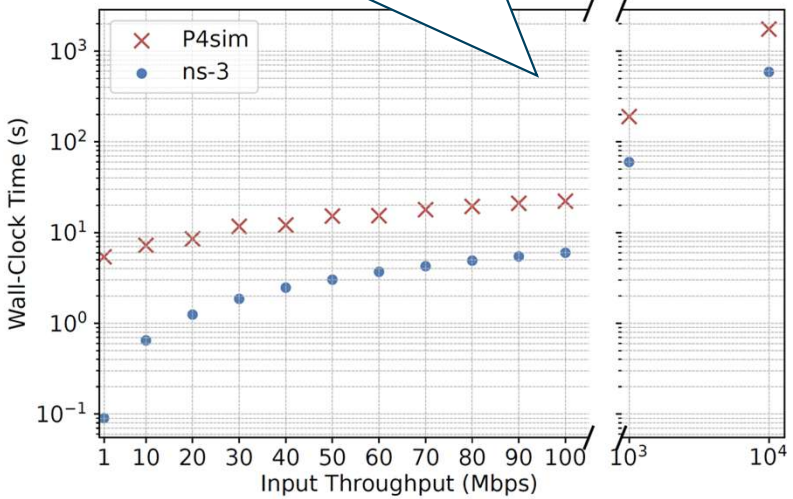
Compare the **P4sim switch** with **ns-3 bridge**, and **Mininet with BMv2 switch** in IPv4 forwarding scenario.

The BMv2 emulation is integrated and transformed into a simulation within ns-3.



✓ Support high performance

The P4 switch requires **3 times** the execution time compared to the ns-3 bridge.



✓ Simulation has acceptable efficiency

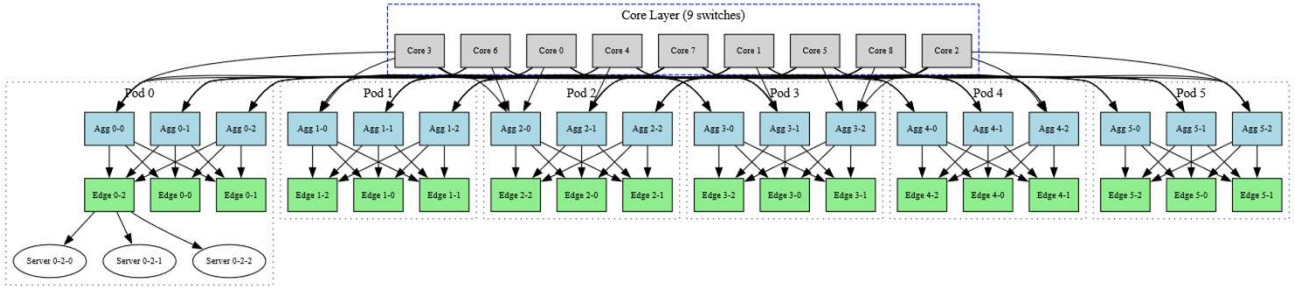
Use Cases: Easy to simulate *the large-scale network*

Large-scale network (Fat Tree)

Test Setup:

- ❑ Set topology parameter: **K = 6**
- ❑ Auto-generate network topology
  - Total 45 switches
- ❑ Auto-generate flow tables
  - Populate all switches with ARP and IPv4 entries
- ❑ Start simulation
- ❑ Tested 2s simulation use 55.601s, *(4 core, 8GB virtual machine)

Tier	Quantity	Switch ID
Core	9	0 ~ 8
Aggregation	18	9 ~ 26
Edge	18	27 ~ 44
Total Switches	**45**	—
Servers	54	—



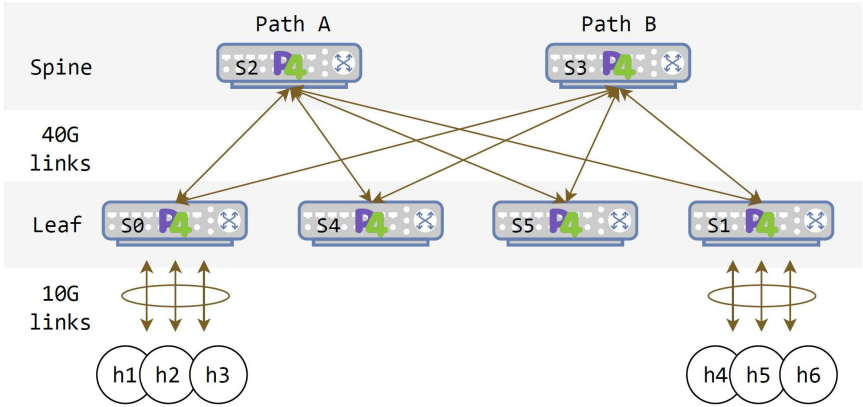
✓ Support scalable and reproducible network simulation

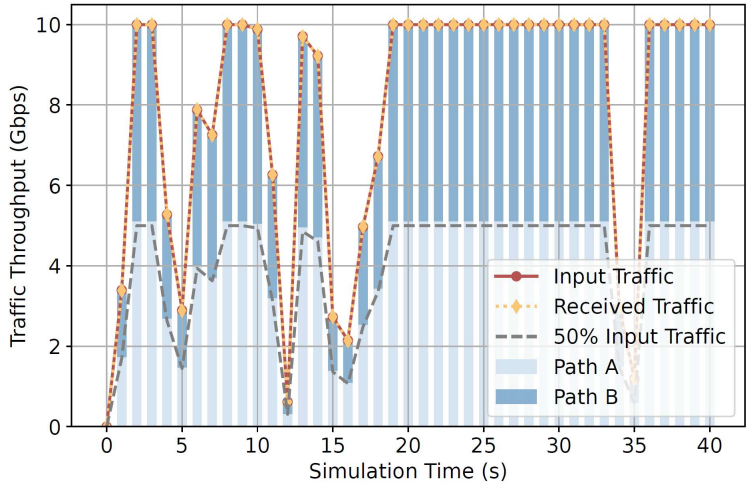
Use Cases

Load Balancing with P4 register and hash

- ❑ Host h1 → Host h4
- ❑ Traffic Pattern:
  - 1000 UDP flows
  - Per-flow rate: 10 Mbps
  - Total peak load: 10 Gbps
- ❑ Result:
  - Near-equal traffic distribution across both paths
  - Confirms the efficacy of hash-based ECMP strategy

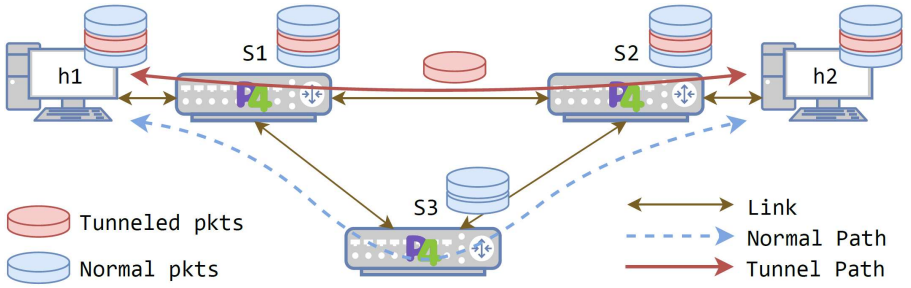
✓ High performance with event scheduling
✓ Extended P4 Support





Use Cases

Tunneling forwarding



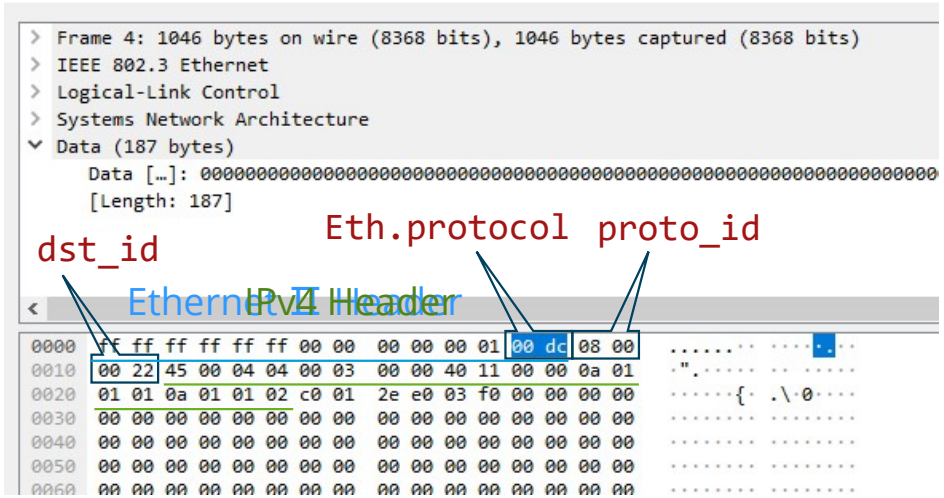
Setup→

- P4 script define the header:

```
header myTunnel_t {
    bit<16> proto_id;
    bit<16> dst_id;
}
```

- ns-3 script define the header:

```
CustomHeader myTunnelHeader;
myTunnelHeader.SetLayer(HeaderLayer::LAYER_3);
myTunnelHeader.SetOperator(ADD_BEFORE);
myTunnelHeader.AddField("proto_id", 16);
myTunnelHeader.AddField("dst_id", 16);
myTunnelHeader.SetField("proto_id", 0x0800);
myTunnelHeader.SetField("dst_id", 0x22);
```



✓ Support protocol-independent design

Current examples

Simulation Parameters for P4 switch

In a P4 switch, the functionality is defined by the user's P4 program and the configuration of the flow table. But the following parameters can also be configured in ns-3 `ns3::P4SwitchNetDevice`.

Parameter	Description
EnableTracing	Enable or disable tracing in the switch
EnableSwap	Enable or disable swapping of the P4 configuration
P4SwitchArch	Switch architecture: 0 for v1model, 1 for PSA, 2 for PNA
JsonPath	Path to the compiled P4 JSON file (*.json)
FlowTablePath	Path to the flow table configuration file
InputBufferSizeLow	Input buffer size for low-priority packets (external packets)
InputBufferSizeHigh	Input buffer size for high-priority packets (internal packets)
QueueBufferSize	Total size of the queue buffer
SwitchRate	Switch processing rate in packets per second (pps)
ChannelType	Channel type: 0 for CSMA, 1 for point-to-point (P2P), default is CSMA

Table: Basic simulation configure parameters

We will add all the tests for the `example` in p4sim, check dir `examples_test_result`:

Example Name	Description	Architecture
basic_tunnel [1]	Tunnel with custom header	V1model
firewall [2]	A basic stateful firewall	V1model
ipv4_forward	Basic forwarding based on `ip_dst`	V1model
load_balance [3]	Load balancing in spine-leaf network	V1model
p4_basic [4]	Basic forwarding based on `ip_dst`	V1model
qos	Forwarding with priority	V1model
simple_pna	IPv4 forwarding with PNA architecture	PNA
simple_psa	IPv4 forwarding with PSA architecture	PSA
simple_v1model	IPv4 forwarding with V1model architecture	V1model

Table: examples of the usecases, from Github: **p4sim-artifact-icns3**

Conclusion and Future Work

P4sim: an enhanced P4-driven simulation framework

- ❑ High-speed, large-scale programmable data plane networks
- ❑ Extended P4 support with V1model, PSA and PNA architectures
- ❑ Protocol independent custom headers design
- ❑ Seamless integration external modules from Bmv2 into ns-3

Evaluation shows:

- ❑ Efficiency and precision in packet processing and queue management
- ❑ Improves simulation fidelity and scalability

Future work:

- ❑ Integrate the link state awareness
- ❑ More use cases and examples

P4sim

Artifacts

P4sim: https://github.com/HapCommSys/p4sim
Artifact: https://github.com/HapCommSys/p4sim-artifact-icns3

References

[1] White Paper: Timestamping and Clock Synchronisation in P4-Programmable Platforms.

[2] Chaves L J, Garcia I C, Madeira E R M. Ofswitch13: Enhancing ns-3 with openflow 1.3 support[C]//Proceedings of the 2016 Workshop on ns-3. 2016: 33-40.

[3] Yu X, Huaxin Z, Zhijun S. Design and implementation of switch module for NS-3[C]//Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools. 2009: 1-10.

[4] Li Y, Miao R, Liu H H, et al. HPCC: High precision congestion control[M]//Proceedings of the ACM special interest group on data communication. 2019: 44-58.

[5] Ben Basat R, Ramanathan S, Li Y, et al. PINT: Probabilistic in-band network telemetry[C]//Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication. 2020: 662-680.

[6] mrosan. 2017. mrosan/P4SwitchNetDevice. https://github.com/mrosan/ P4SwitchNetDevice

[7] PIFO-TM ns3-bmv2. https://github.com/ PIFO-TM/ns3-bmv2

[8] Chengze Fan, Jun Bi, Yu Zhou, Cheng Zhang, and Haisu Yu. 2017. NS4: A P4-driven Network Simulator. In Proceedings of the SIGCOMM Posters and Demos (Los Angeles, CA, USA) (SIGCOMM Posters and Demos '17). Association for Computing Machinery, New York, NY, USA, 105–107. doi:10.1145/3123878.3132002

[9] Jiasong Bai, Jun Bi, Peng Kuang, Chengze Fan, Yu Zhou, and Cheng Zhang. 2018. NS4: Enabling Programmable Data Plane Simulation. In Proceedings of the Symposium on SDN Research (Los Angeles, CA, USA) (SOSR '18). Association for Computing Machinery, New York, NY, USA, Article 12, 7 pages. doi:10.1145/3185467.3185470

[10]NetFPGA SUME https://digilent.com/reference/programmable-logic/netfpga-sume/start?srsltid=AfmBOopC0yOIwFaz6ChotloSovV1TFDC-lnZz99yu2oiluXjJYKu4pt0

[11] https://www.nextplatform.com/2022/08/16/like-a-drumbeat-broadcom-doubles-ethernet-bandwidth-with-tomahawk-5/

[12] P4sim github link: https://github.com/HapCommSys/p4sim Artifact: https://github.com/HapCommSys/p4sim-artifact-icns3

TECHNISCHE UNIVERSITÄT DRESDEN

CeTI Centre for Tactile Internet with Human-in-the-Loop

Topic: P4sim: Simulating programmable switches in ns-3

Haptic Communication Systems, Faculty of Electrical and Computer Engineering
Center for Tactile Internet with Human-in-the-Loop (CeTI), TU Dresden, Germany

Thanks for your attention!

Speaker: Ma Mingyu

P4sim Design

Three types of switch modeling approaches

Existing switch modeling approaches in ns-3 can be broadly categorized into three types:

- **Dedicated Switch Models** (e.g., iSLIP, CICB [3])
  - Custom scheduling/buffering models
  - Simulates internal data paths, not programmable logic
- **Specialized Behavior Models** (e.g., ECN [4], INT [5])
  - Simulate specific behaviors only
  - Lack support for general-purpose programmable pipelines
- **External Modules**
  - Easy integration with real-world control plane APIs
  - Reuse mature external libraries and protocol implementations (e.g., OpenFlow [2])
  - Lower development effort when only high-level switch behavior is needed

P4 already have the behavior model (Bmv2) libraries → External Module

P4sim Design

Queue Scheduling & Virtual Time Alignment

- Virtual Time in ns-3
  - ns-3 uses discrete-event simulation with a virtual time scheduler
  - All events (packet arrivals, queueing, transmissions) are timestamped and executed without real wall-clock time
  - This ensures precise timing control and reproducibility in simulations
- Align P4 packet processing with ns-3's virtual time:
  - Schedule dequeue as a virtual-time event
- Key Differences from Traditional Switch Models:
  - Programmable switches have **dynamic behaviors** (recirculation, drop, resubmit)
  - Packet output **time and order** may change depending on the P4 program, making timing unpredictable without virtual time alignment
- Solution: **Reschedule the event** (e.g., due to recirculation or dependency) **for the next switch processing cycle**, preserving correct timing semantics.

Evaluation

Use Cases: Load Balancing

Network Topo

- Source: Host h1
- Traffic Pattern:
  - 1000 UDP flows
  - Per-flow rate: 10 Mbps
  - Total peak load: 10 Gbps
  - Traffic model: *OnOffApplication*
  - ON/OFF periods: Exponentially distributed
- Result:
  - Near-equal traffic distribution across both paths
  - Confirms the efficacy of hash-based ECMP strategy

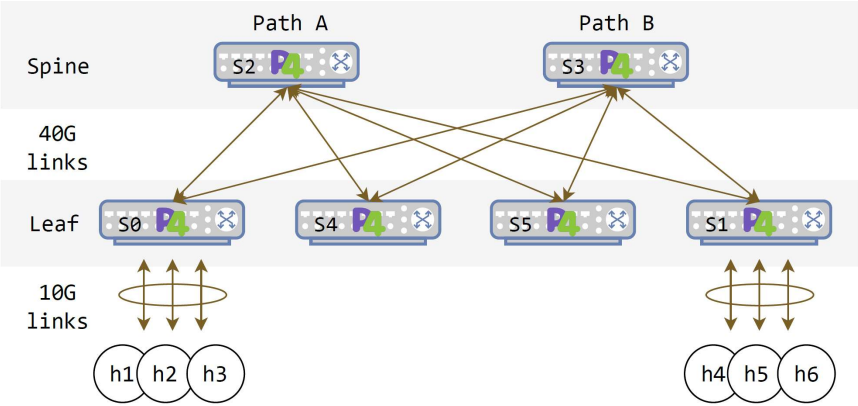


Figure 6: Spine-Leaf Network Topology for ECMP Load Balancing

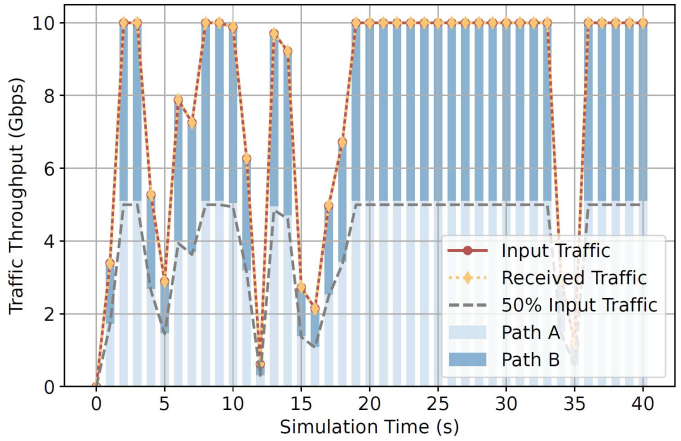


Figure 7: Throughput of Hosts and Paths in Load-Balancing System

P4sim Design

Simulation Time Representation in ns-3 & P4sim

Enqueue:

- Enqueued into bands based on type of service and priority allocation on P4
- Calculate the expected dequeue time T_{exDeq}

Dequeue:

- Packets always dequeued from PFIFO 0 first. Then PFIFO 1 and lastly PFIFO 2
- Within each band, packets are eligible for dequeue at their T_{exDeq}, but actual dequeue occurs only if resources permit ($Rate_{switch}$).

$$T_{exDeq} = T_{Enq} + \frac{1}{Rate_{queue}}$$

T_{exDeq}: The expected dequeue time for packet
T_{Enq}: The enqueue timestamp
$Rate_{queue}$: The rate settings of the queue

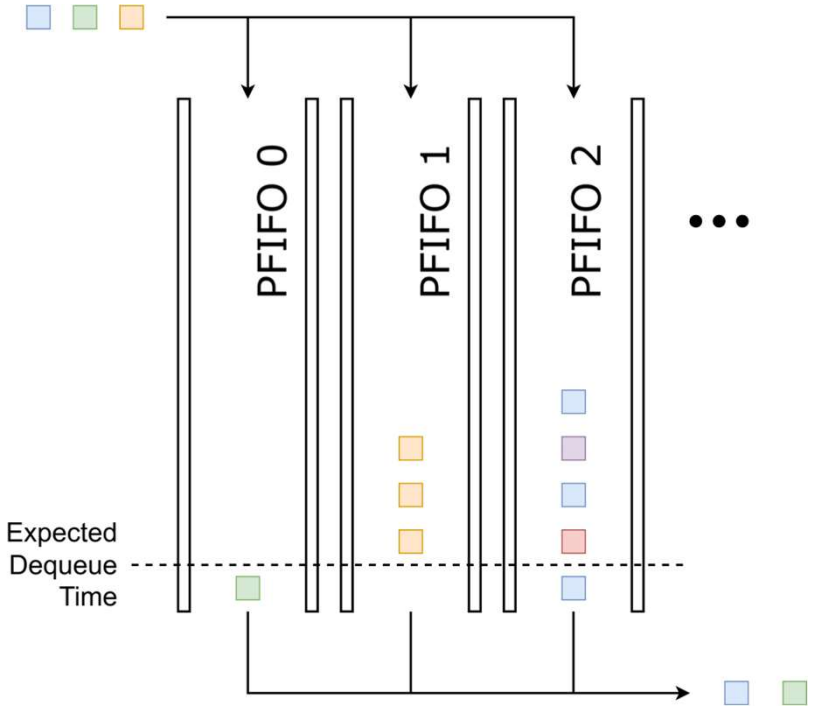


Figure: Per-Queue Rate Configurable PFIFO model (from Bmv2)

P4sim Implementation

The Overview of the P4sim

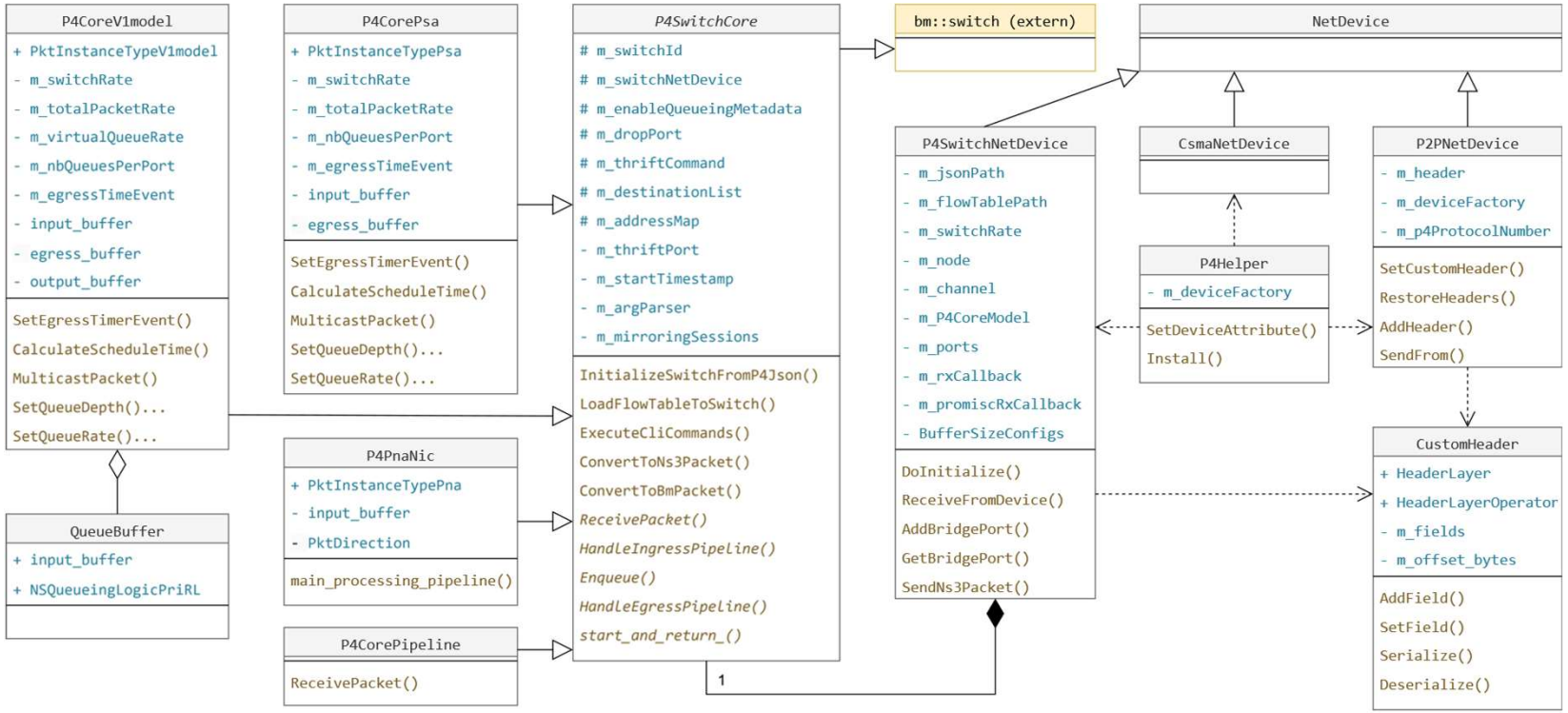


Figure 4: UML Diagram of P4 Switch Implementation