Connecting ideas, anticipating the future:
collaborative innovation for 5G and 6G networks.

II INTERNATIONAL WORKSHOP
xGMobile

Organized by:

xGMobile Centro de Competência EMBRAPII Inatel em Redes 5G e 6G

Inatel

FAPEMIG

EMBRAPII

GOVERNO DE MINAS
AQUI O TREM PROSPERA.

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO

GOVERNO DO BRASIL
DO LADO DO POVO BRASILEIRO

Engineering for the Unknown:
Open-Source 6G Experimentation Toolbox for Protocol Innovation and High-Fidelity Evaluation

Christian Esteve Rothenberg (chesteve@dca.fee.unicamp.br)

SMARTNESS / University of Campinas (UNICAMP)

Organized by:

xGMobile Inatel FAPEMIG EMBRAPII GOVERNO DE MINAS AQUI O TREM PROSPERA. MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO GOVERNO DO BRASIL DO LADO DO POVO BRASILEIRO

Agenda

- **6G R&D Landscape**
  - Challenges with Existing Testb**eds**
- **The Toolbox**
  - Network emulation: P7
  - Traffic generation: PIPO-TG, P4R
  - (Selected) Use Cases
  - Future Directions
  - Community & Impact
- **Conclusions**

6G R&D Landscape

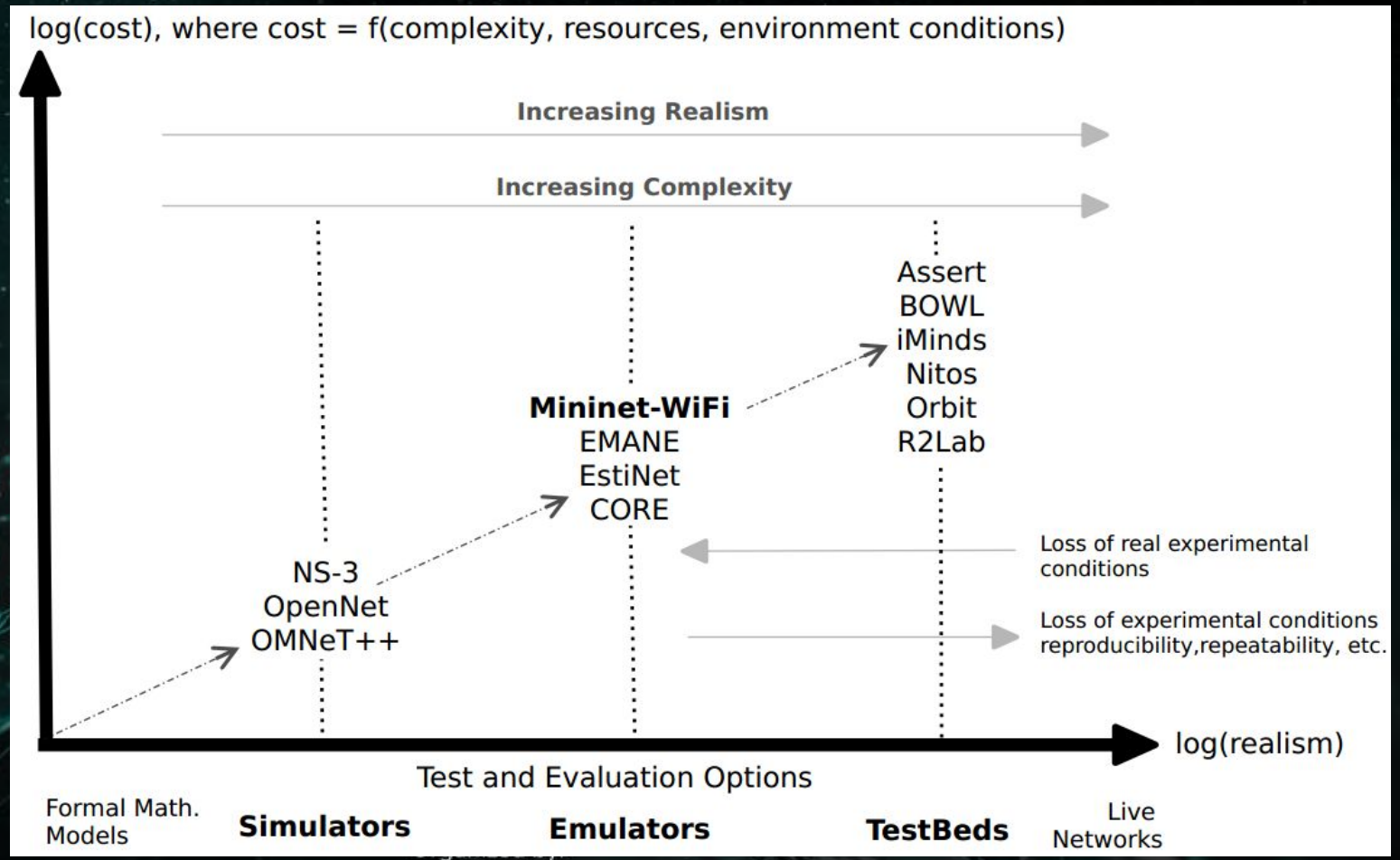
- The path (R&D&Standardization) toward 6G introduces profound **uncertainty**.
- Moving beyond simple speed increases to fundamentally undefined **protocol stack**(s?).
  - **Protocol Stacks:** Future headers are undefined.
  - **Programmable 6G Core:** New abstractions e.g. semantic networking, intent-aware functions, ….
  - **Cognitive 6G Core:**
    Shift to AI-native, NWDAF++ exploration
  - **Control:** Real-time control loops, predictive mobility & slicing controllers, etc.

Challenges with Existing Testbeds

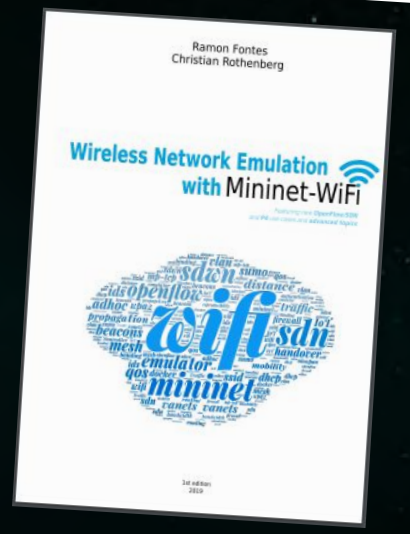
- Inflexible 5G implementations
- Hard to test new headers
- Limited AI/telemetry integration
- Unrealistic workload support
- Restricted accessibility:
  - Cost-related
  - Knowledge-related

Experimental environments

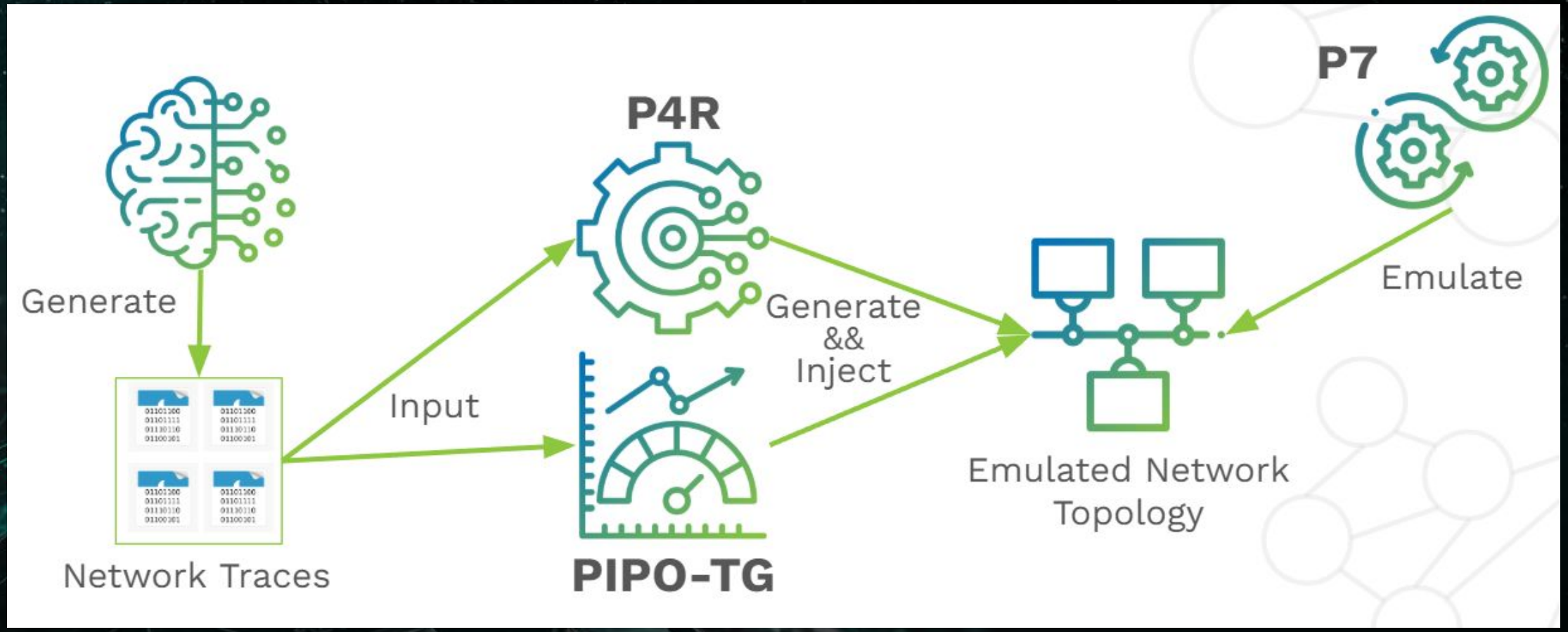
Testbeds, Emulators, Simulators?



https://mininet-wifi.github.io/book/



Open-Source 6G Experimentation Toolbox



Experimental environments

Testbeds, **Emulators**, Simulators?

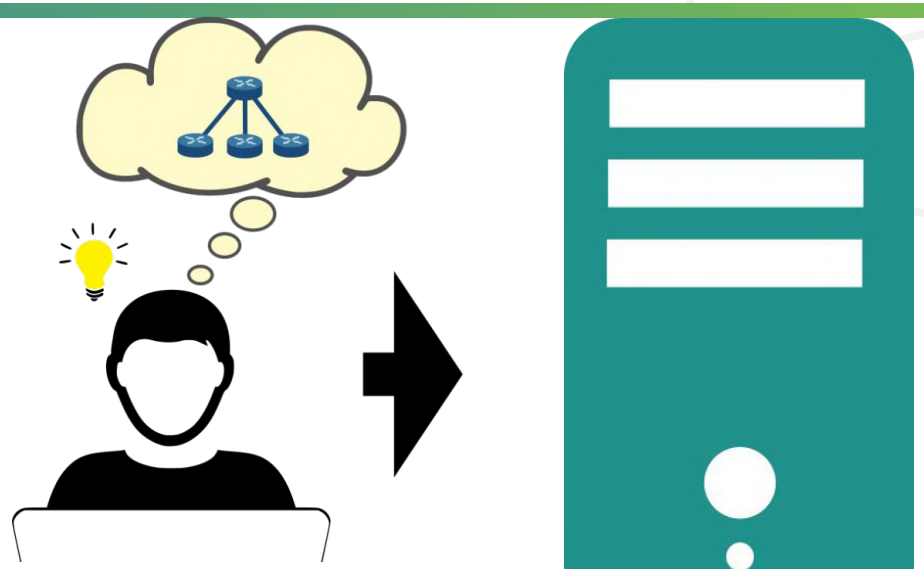
Mininet

Mininet-WiFi

Emulator for Software-Defined Wireless Networks

SMARTNESS

```
# Define 6G Topology Script
topo = Topology()

# Add Nodes
ue = topo.add_host('UE_1')
upf = topo.add_switch('UPF_Core')

# Add Link with 6G Impairments
topo.add_link(ue, upf,
  bw='100G',
  delay='1ms',
  jitter='0.1ms')

topo.compile_p4()
```



How can we create more realistic network scenarios exploring the programmable capabilities of modern switch architectures?

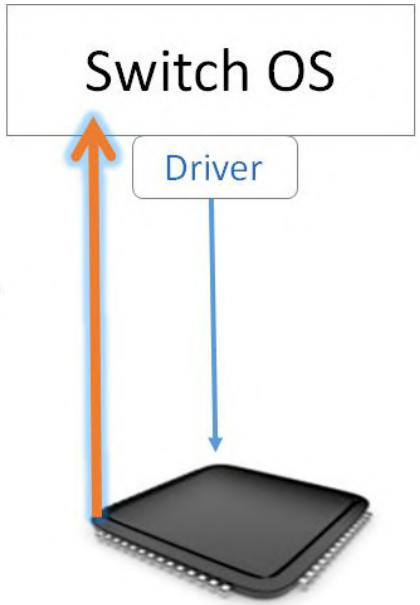
Using P7! The P4 Programmable Patch Panel.

Background

Programming Protocol-independent Packet Processors

Traditional Switch

"This is how I process packets …"



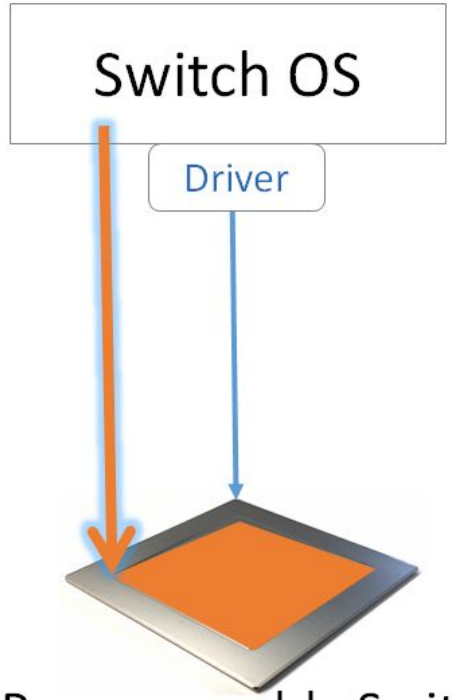
Fixed-function switch

Programmable Switch

"This is precisely how you must process packets"

```
table int_table {
  reads {
    ip.protocol;
  }
  actions {
    export_queue_latency;
  }
}
```

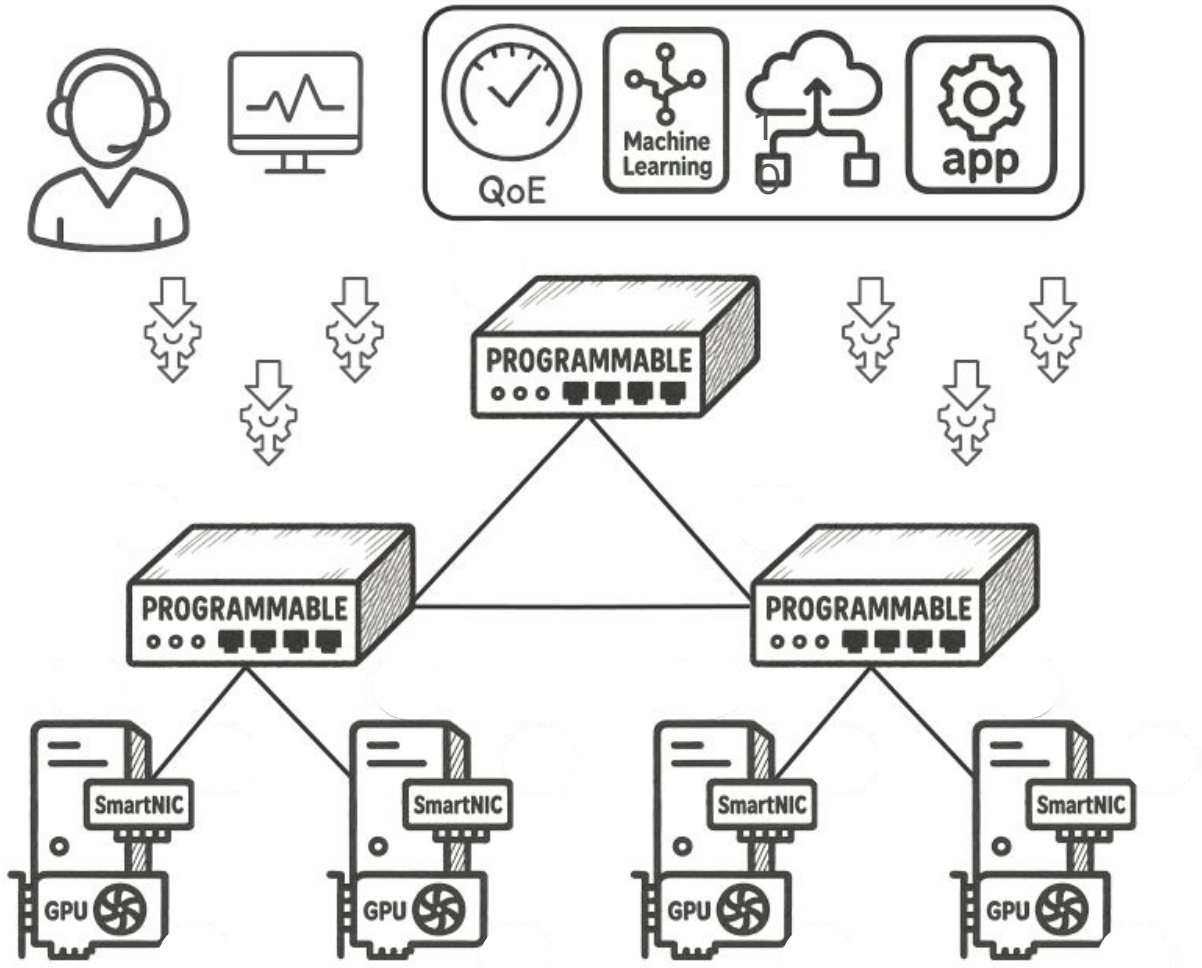
```
action export_queue_latency (sw_id) {
  add_header(int_header);
  modify_field(int_header.kind, TCP_OPTION_INT);
  modify_field(int_header.len, TCP_OPTION_INT_LEN);
  modify_field(int_header.sw_id, sw_id);
  modify_field(int_header.q_latency,
               intrinsic_metadata.deq_timedelta);
  add_to_field(tcp.dataOffset, 2);
  add_to_field(ipv4.totalLen, 8);
  subtract_from_field(ingress_metadata.tcpLength,
                      12);
}
```

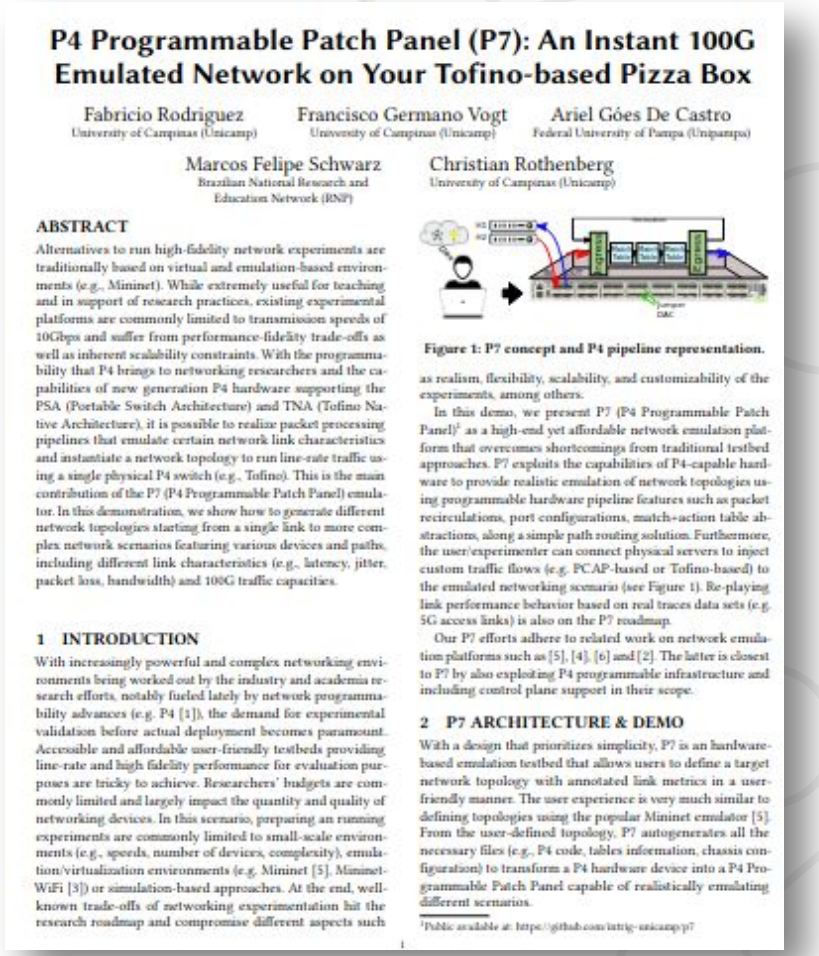


Programmable Switch

P7: P4 Programmable Patch Panel

High-fidelity Network Emulation





P4 Programmable Patch Panel (P7): An Instant 100G Emulated Network on Your Tofino-based Pizza Box

Fabricio Rodriguez, University of Campinas (Unicamp)
Francisco Germano Vogt, University of Campinas (Unicamp)
Ariel Góes De Castro, Federal University of Pampa (Unipampa)
Marcos Felipe Schwarz, Brazilian National Research and Education Network (RNP)
Christian Rothenberg, University of Campinas (Unicamp)

ABSTRACT

Alternatives to run high-fidelity network experiments are traditionally based on virtual and emulation-based environments (e.g., Mininet). While extremely useful for teaching and in support of research practices, existing experimental platforms are commonly limited to transmission speeds of 10Gbps and suffer from performance-fidelity trade-offs as well as inherent scalability constraints. With the programmability that P4 brings to networking researchers and the capabilities of new generation P4 hardware supporting the PSA (Portable Switch Architecture) and TNA (Tofino Native Architecture), it is possible to realize packet processing pipelines that emulate certain network link characteristics and instantiate a network topology to run line-rate traffic using a single physical P4 switch (e.g., Tofino). This is the main contribution of the P7 (P4 Programmable Patch Panel) emulator. In this demonstration, we show how to generate different network topologies starting from a single link to more complex network scenarios featuring various devices and paths, including different link characteristics (e.g., latency, jitter, packet loss, bandwidth) and 100G traffic capacities.

1 INTRODUCTION

With increasingly powerful and complex networking environments being worked out by the industry and academia research efforts, notably fueled lately by network programmability advances (e.g. P4 [1]), the demand for experimental validation before actual deployment becomes paramount. Accessible and affordable user-friendly testbeds providing line-rate and high fidelity performance for evaluation purposes are tricky to achieve. Researchers' budgets are commonly limited and largely impact the quantity and quality of networking devices. In this scenario, preparing an running experiments are commonly limited to small-scale environments (e.g., speeds, number of devices, complexity), emulation/virtualization environments (e.g. Mininet [5], Mininet-WiFi [3]) or simulation-based approaches. At the end, well-known trade-offs of networking experimentation hit the research roadmap and compromise different aspects such as realism, flexibility, scalability, and customizability of the experiments, among others.

Figure 1: P7 concept and P4 pipeline representation.

In this demo, we present P7 (P4 Programmable Patch Panel)[1] as a high-end yet affordable network emulation platform that overcomes shortcomings from traditional testbed approaches. P7 exploits the capabilities of P4-capable hardware to provide realistic emulation of network topologies using programmable hardware pipeline features such as packet recirculations, port configurations, match-action table abstractions, along a simple path routing solution. Furthermore, the user/experimenter can connect physical servers to inject custom traffic flows (e.g. PCAP-based or Tofino-based) to the emulated networking scenario (see Figure 1). Re-playing link performance behavior based on real traces data sets (e.g. 5G access links) is also on the P7 roadmap.

Our P7 efforts adhere to related work on network emulation platforms such as [5], [4], [6] and [2]. The latter is closest to P7 by also exploiting P4 programmable infrastructure and including control plane support in their scope.

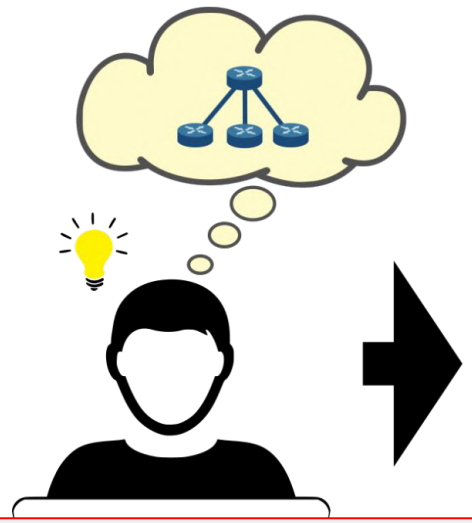
2 P7 ARCHITECTURE & DEMO

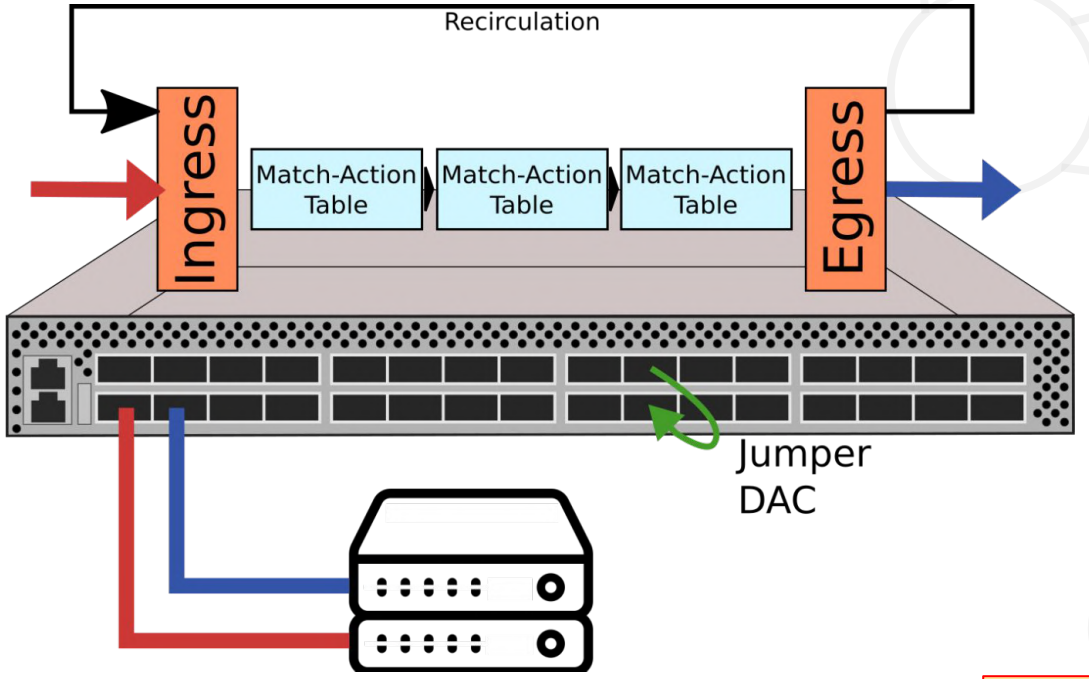
With a design that prioritizes simplicity, P7 is an hardware-based emulation testbed that allows users to define a target network topology with annotated link metrics in a user-friendly manner. The user experience is very much similar to defining topologies using the popular Mininet emulator [5]. From the user-defined topology, P7 autogenerates all the necessary files (e.g., P4 code, tables information, chassis configuration) to transform a P4 hardware device into a P4 Programmable Patch Panel capable of realistically emulating different scenarios.

[1]Public available at: https://github.com/intrig-unicamp/p7

Source: P7 SIGCOMM Demo, 2022

P7: P4 Programmable Patch Panel





P7 is a high-fidelity 100G traffic network emulation, including various link characteristics such as latency, jitter, packet loss, and bandwidth, as well as the option to customize network topologies.

Everything implemented in a single P4 switch

Link Emulation

P4/TNA implementation approaches

Link Connectivity	Intern Recirculation + internal Tag
Latency [ms]	Internal timer + recirculation TM + Pipelines recirculation
Jitter [ms]	Hash to determine recirculation times Lookup table with mathematical functions
Packet loss [%]	Random function to determine packets discard probability Realistic packet loss model
Bandwidth	Rate limit TNA TM feature Port configuration and shaping

This are the available link metrics and how were implemented

Link Emulation

P4/TNA implementation approaches

Link Connectivity	Intern Recirculation + internal Tag
Latency [ms]	Internal timer + recirculation TM + Pipelines recirculation
Jitter [ms]	Hash to determine recirculation times Lookup table with mathematical functions
Packet loss [%]	Random function to determine packets discard probability Realistic packet loss model
Bandwidth	Rate limit TNA TM feature Port configuration and shaping

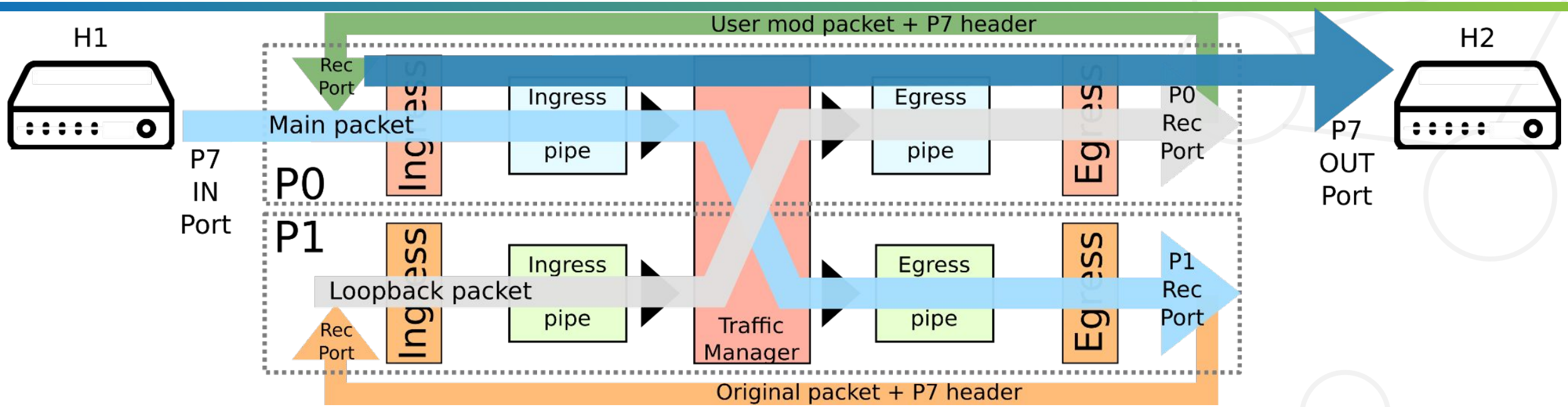
Limitation

- Switches are treated as a simple forwarding nodes without instantiate a custom P4 code

Solution

- Allocate a dedicated pipe for user-defined P4 code and incorporate it into each emulated nod

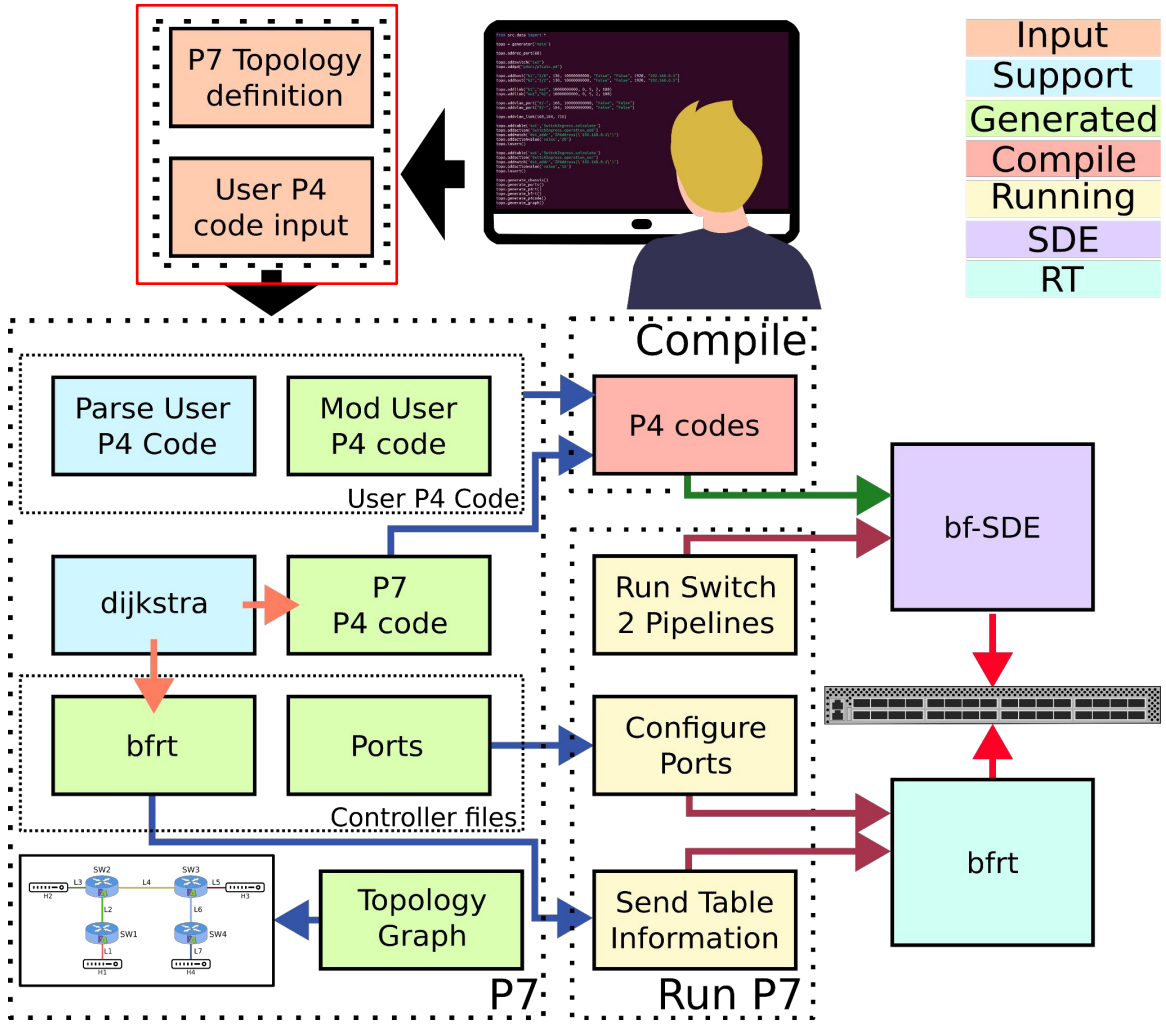
P7 Multiple Pipelines Design



We propose a solution where a dedicated pipe runs the P7 P4 code, and a separate pipe runs the user-defined P4 code

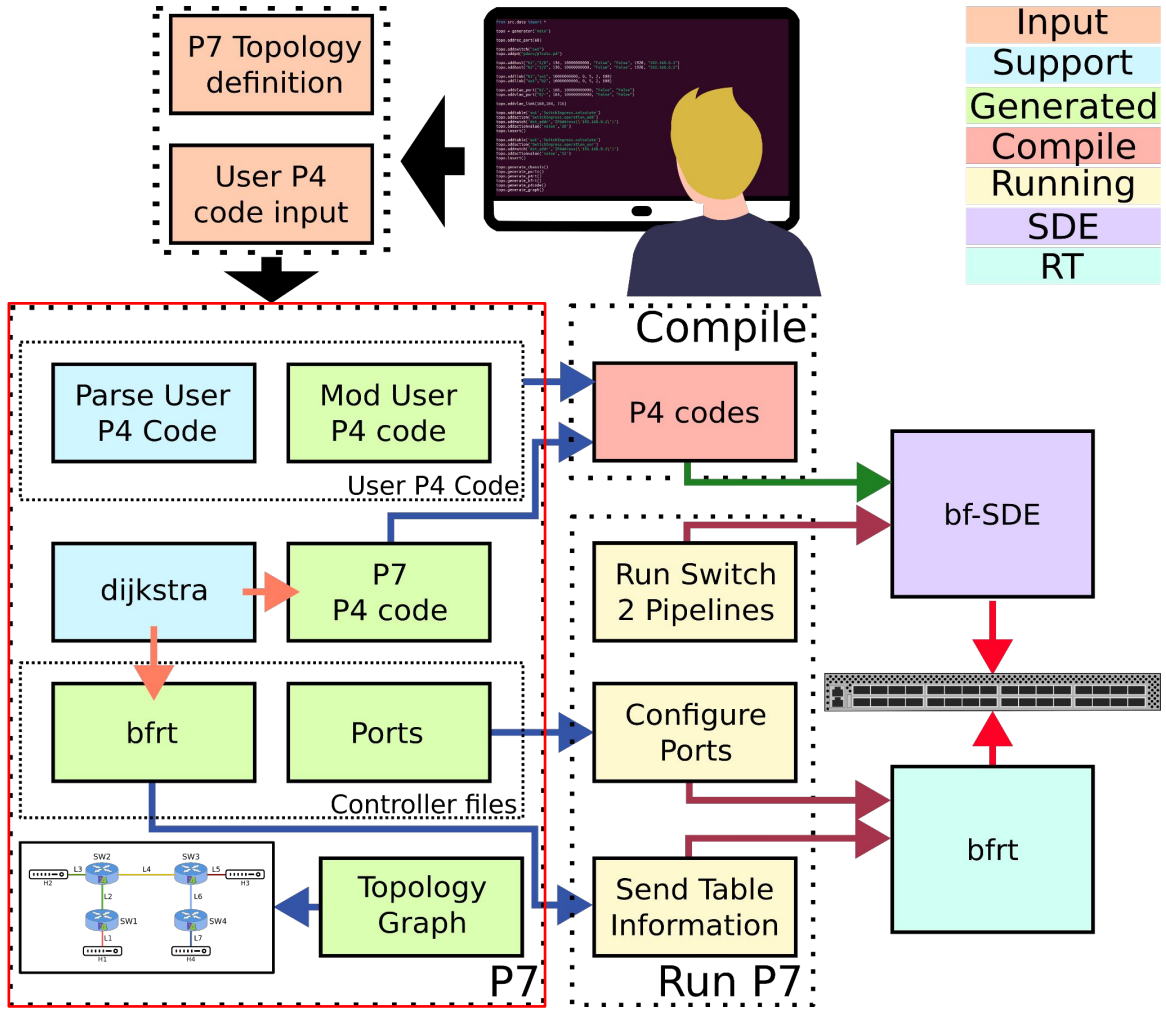
We send the packet in the P7 pipe (P0) to the pipe where the user-defined P4 code is running (P1) using recirculation

P7 Architecture



The user defines the topology and sets a custom P4 code.

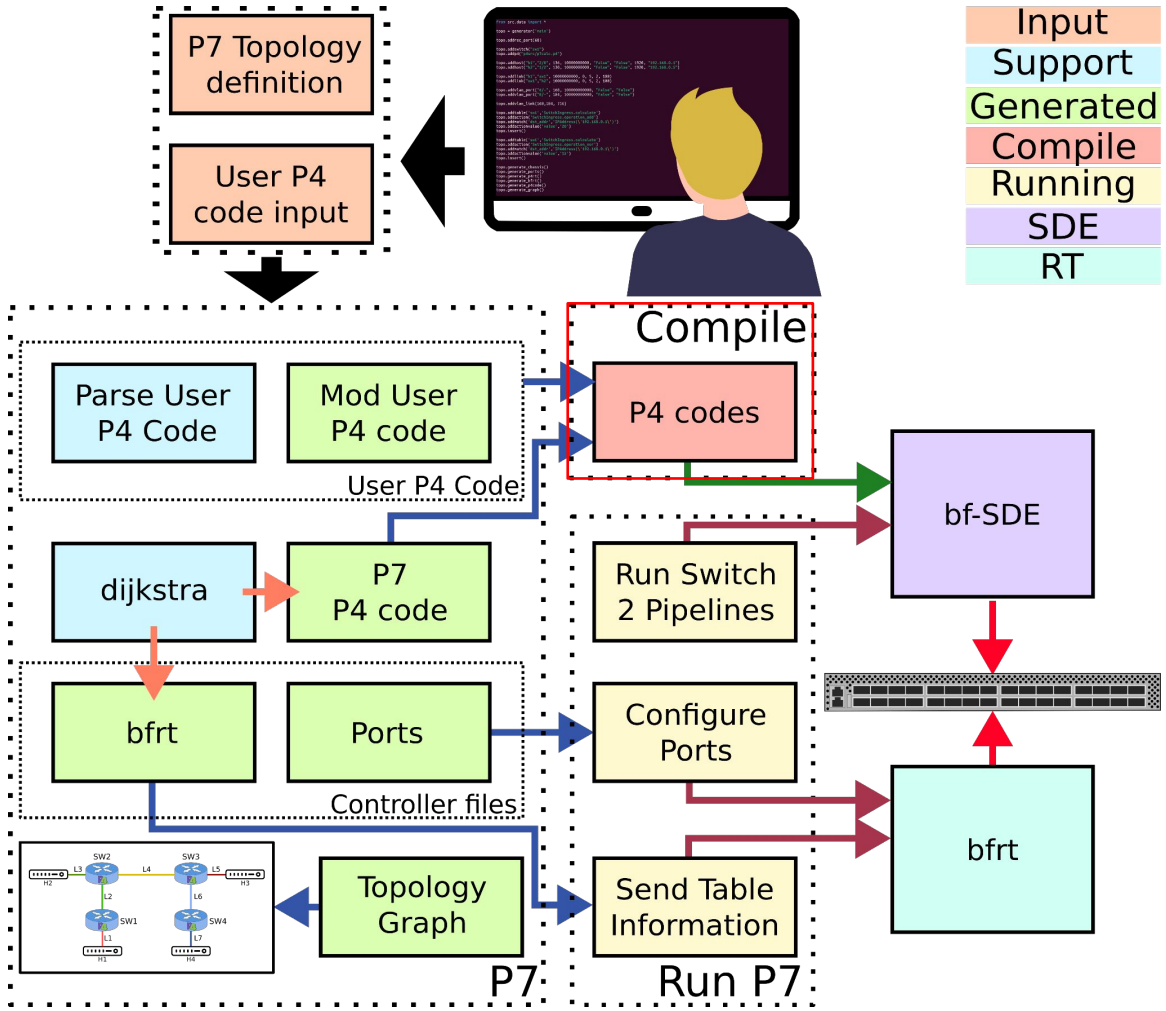
P7 Architecture



P7 processes the data and generates the necessary files:

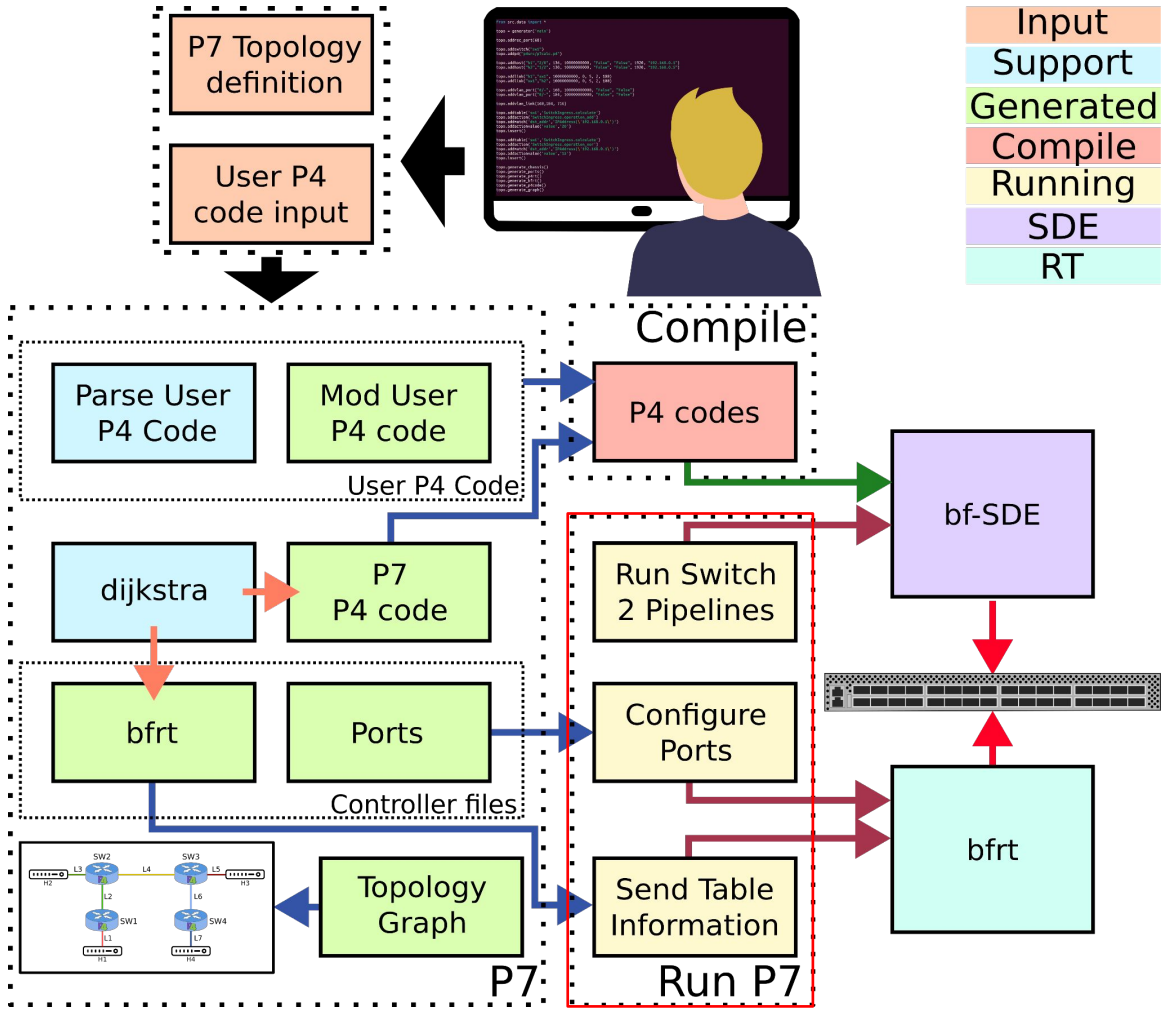
- P7 P4 code
- User P4 code
- Tables information
- Ports configuration

P7 Architecture



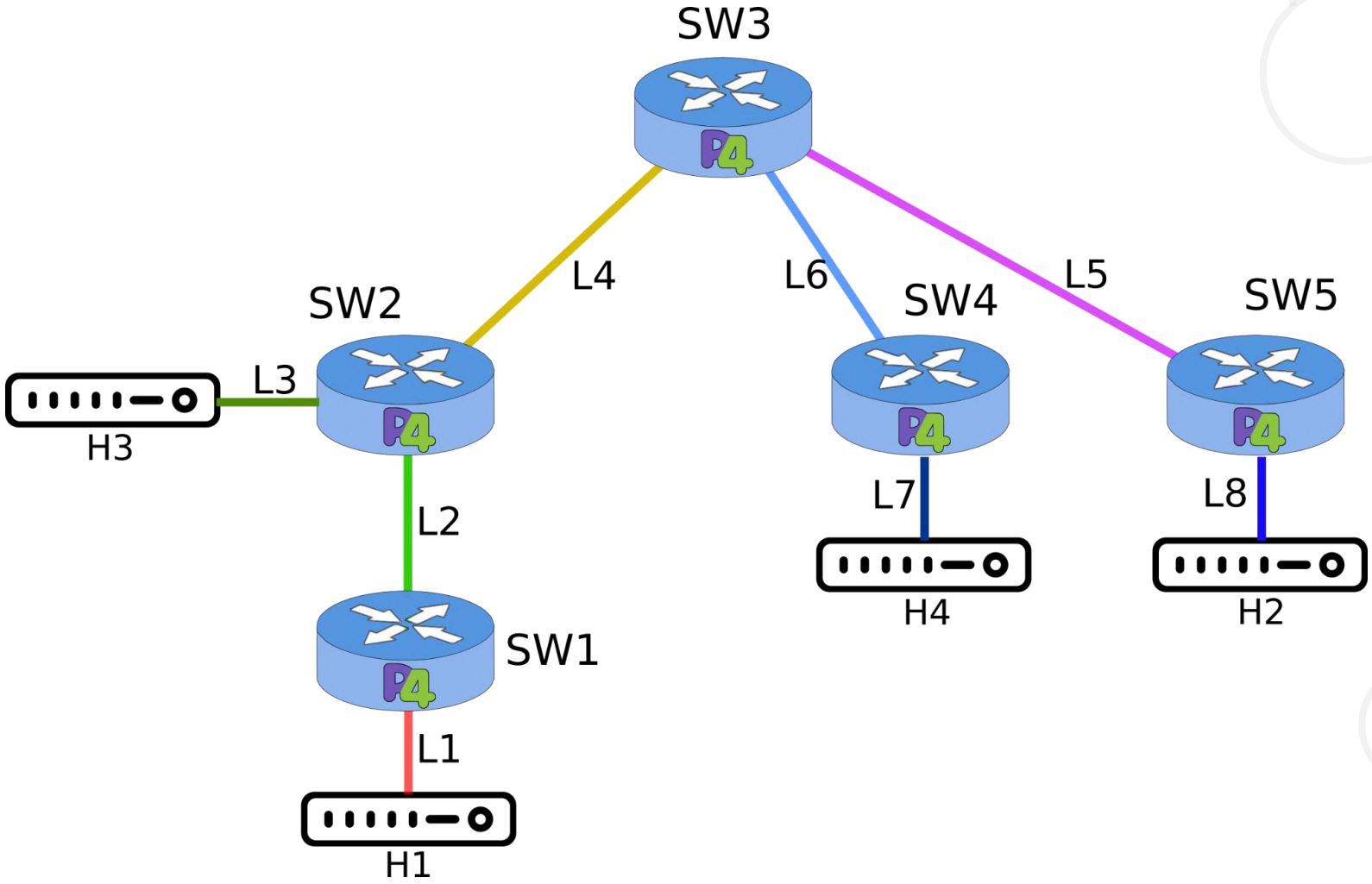
The user needs to compile both P4 codes

P7 Architecture

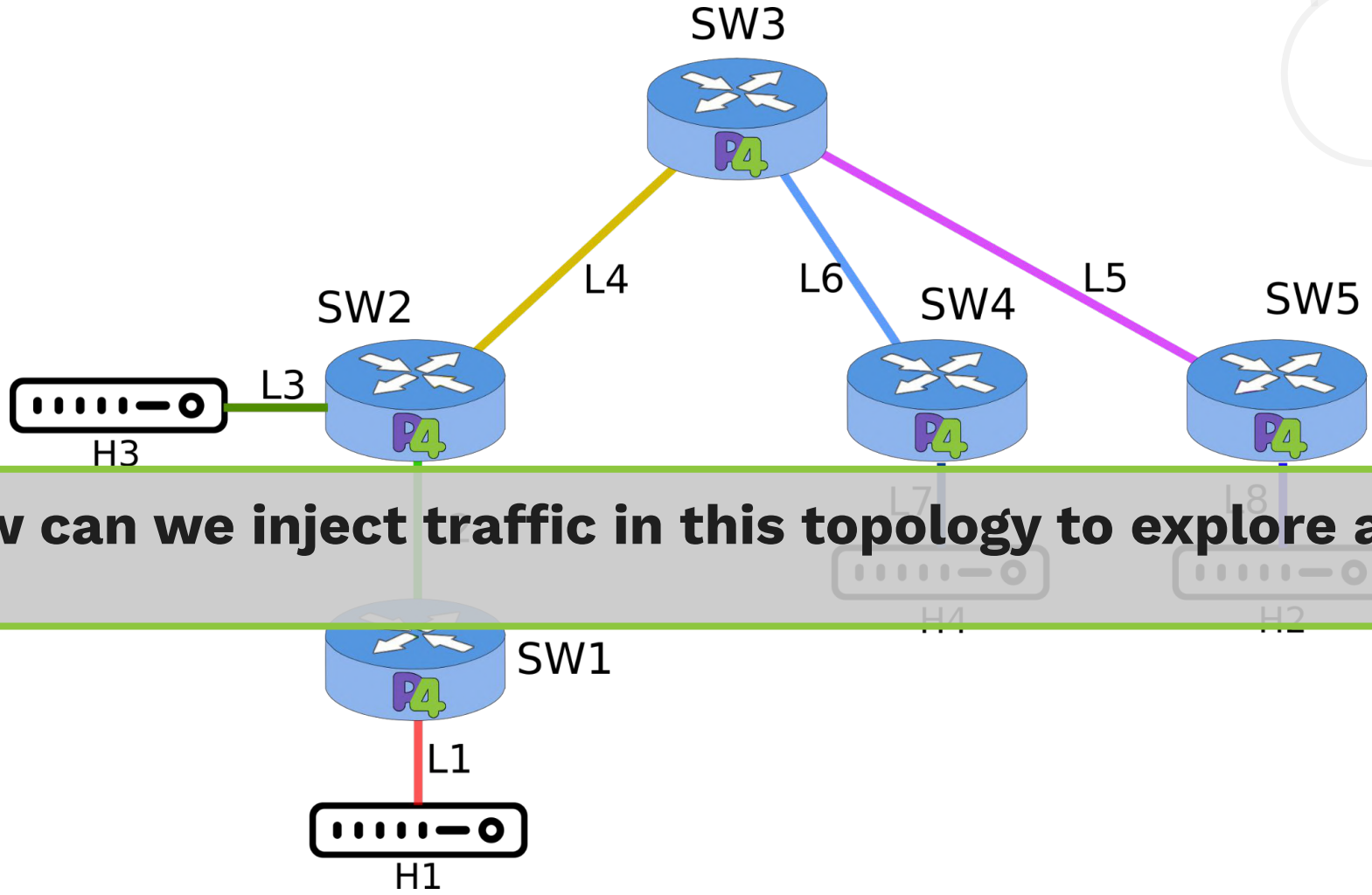


Finally, the user can run the switch with both P4 codes and send the tables and ports configuration using the bfrt.

Example of P7 topology

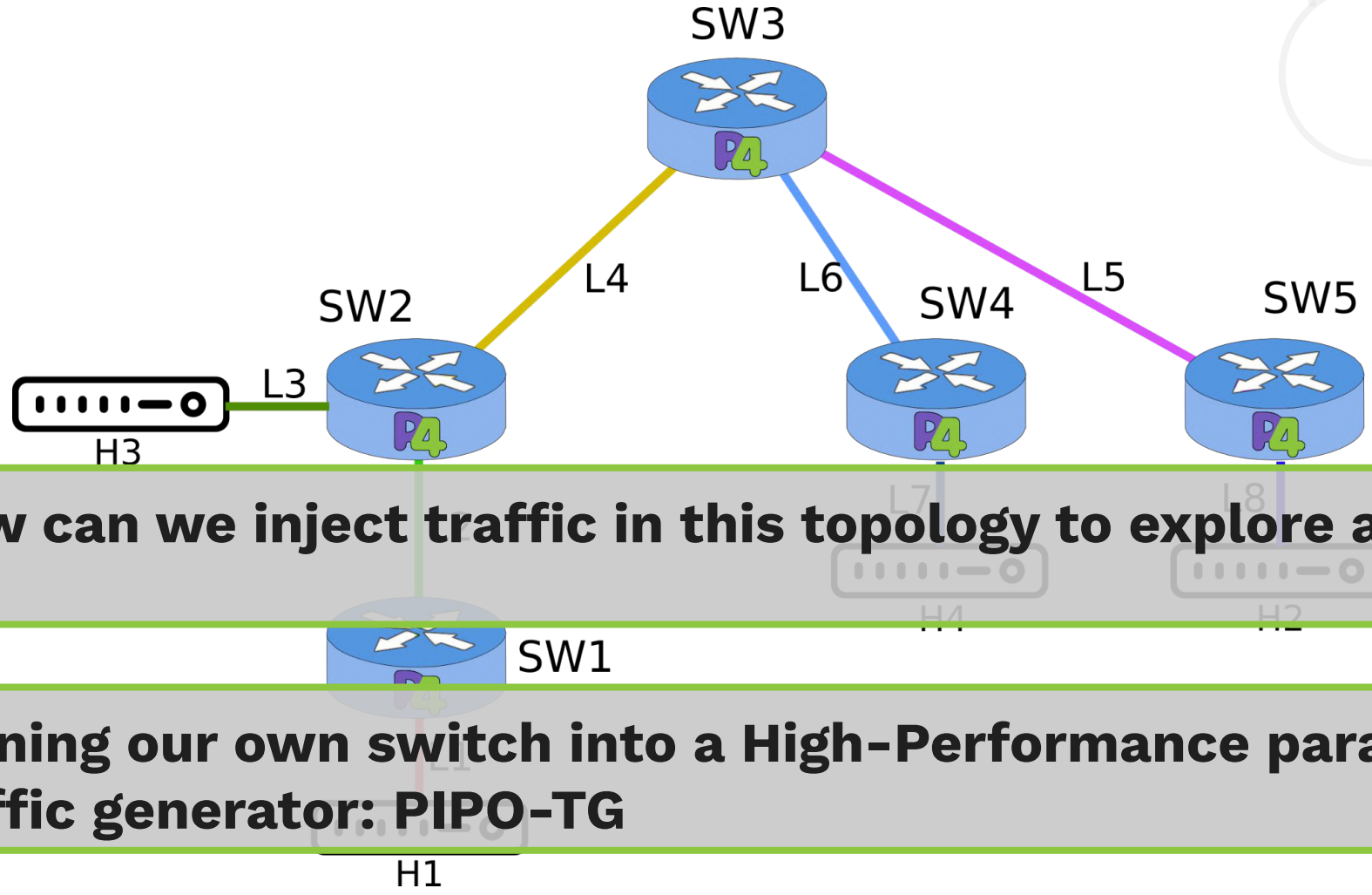


Example of P7 topology



How can we inject traffic in this topology to explore at 100Gb/s?

Example of P7 topology

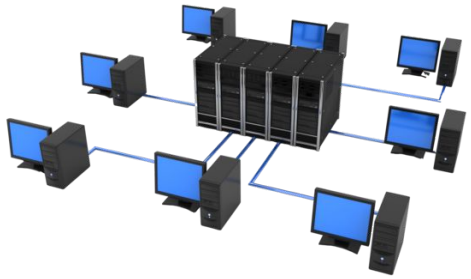


How can we inject traffic in this topology to explore at 100Gb/s?

Turning our own switch into a High-Performance parameterizable traffic generator: PIPO-TG

PIPO-TG: Programmable HW Traffic Generation

Problem, Challenges, State-of-art



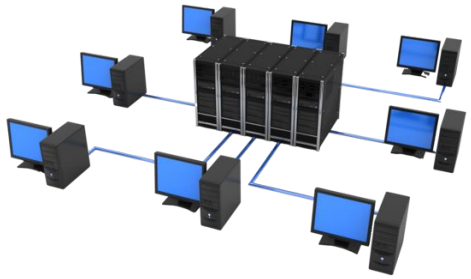
How we can reproduce traffic patterns?

How inject this traffic into the network?

Network Testing

PIPO-TG: Programmable HW Traffic Generation

Problem, Challenges, State-of-art



How we can reproduce traffic patterns?

How inject this traffic into the network?

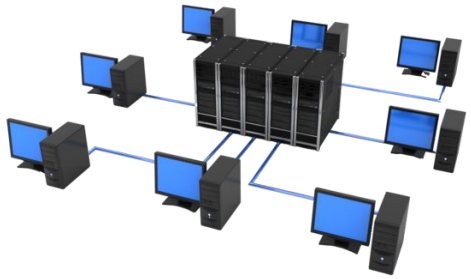
Network Testing

Software:

- Iperf
- TCP Replay
- TRex

PIPO-TG: Programmable HW Traffic Generation

Problem, Challenges, State-of-art



How we can reproduce traffic patterns?

How inject this traffic into the network?

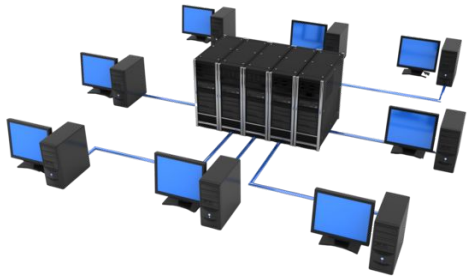
Network Testing

Software:
- Iperf
- TCP Replay
- TRex

However...
- Performance
- Inaccuracy

PIPO-TG: Programmable HW Traffic Generation

Problem, Challenges, State-of-art



How we can reproduce traffic patterns?

How inject this traffic into the network?

Network Testing

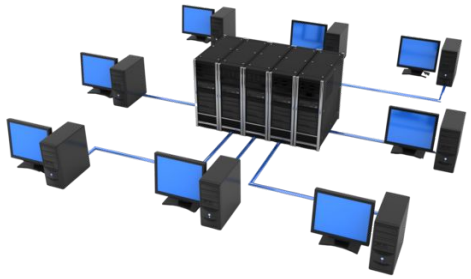
Software:
- Iperf
- TCP Replay
- TRex

However...
- Performance
- Inaccuracy

Hardware:
- FPGA-based
- ConnectX-5

PIPO-TG: Programmable HW Traffic Generation

Problem, Challenges, State-of-art



How we can reproduce traffic patterns?

How inject this traffic into the network?

Network Testing

Software:
- Iperf
- TCP Replay
- TRex

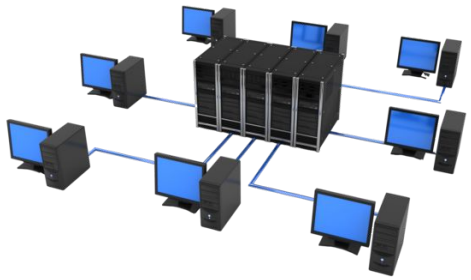
However...
- Performance
- Inaccuracy

Hardware:
- FPGA-based
- ConnectX-5

However...
- Scalability and Cost
- Inflexibility

PIPO-TG: Programmable HW Traffic Generation

Problem, Challenges, State-of-art



How we can reproduce traffic patterns?

How inject this traffic into the network?

P4TG: 1 Tb/s Traffic Generation for Ethernet/IP Networks

STEFFEN LINDNER, MARCO HÄBERLE, (Graduate Student Member, IEEE), AND MICHAEL MENTH, (Senior Member, IEEE)

HyperTester: High-Performance Network Testing Driven by Programmable Switches

Dai Zhang, Yu Zhou, Zhaowei Xi, Yangyang Wang, Mingwei Xu, *Senior Member, IEEE*, and Jianping Wu, *Fellow, IEEE*

Network Testing

Software:
- Iperf
- TCP Replay
- TRex

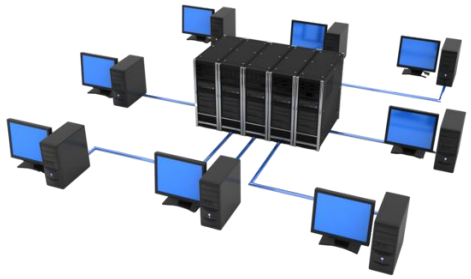
However...
- Performance
- Inaccuracy

Hardware:
- FPGA-based
- ConnectX-5

However...
- Scalability and Cost
- Inflexibility

PIPO-TG: Programmable HW Traffic Generation

Problem, Challenges, State-of-art



How we can reproduce traffic patterns?

How inject this traffic into the network?

P4TG: 1 Tb/s Traffic Generation for Ethernet/IP Networks

STEFFEN LINDNER, MARCO HÄBERLE, (Graduate Student Member, IEEE), AND MICHAEL MENTH, (Senior Member, IEEE)

HyperTester: High-Performance Network Testing Driven by Programmable Switches

Dai Zhang, Yu Zhou, Zhaowei Xi, Yangyang Wang, Mingwei Xu, *Senior Member, IEEE*, and Jianping Wu, *Fellow, IEEE*

Limitations:
- Flexibility
- Server dependency
- not open-source
- limited traffic patterns

Network Testing

Software:
- Iperf
- TCP Replay
- TRex

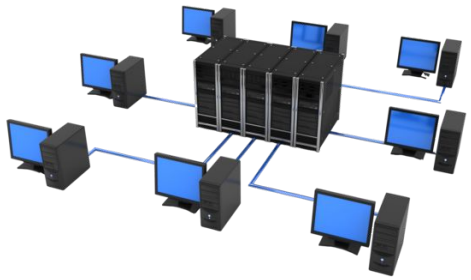
However...
- Performance
- Inaccuracy

Hardware:
- FPGA-based
- ConnectX-5

However...
- Scalability and Cost
- Inflexibility

PIPO-TG: Programmable HW Traffic Generation

Problem, Challenges, State-of-art



How we can reproduce traffic patterns?

How inject this traffic into the network?

P4TG: 1 Tb/s Traffic Generation for Ethernet/IP Networks

STEFFEN LINDNER, MARCO HÄBERLE, (Graduate Student Member, IEEE), AND MICHAEL MENTH, (Senior Member, IEEE)

HyperTester: High-Performance Network Testing Driven by Programmable Switches

Dai Zhang, Yu Zhou, Zhaowei Xi, Yangyang Wang, Mingwei Xu, *Senior Member, IEEE*, and Jianping Wu, *Fellow, IEEE*

Limitations:
- Flexibility
- Server dependency
- not open-source
- limited traffic patterns

Network Testing

Software:
- Iperf
- TCP Replay
- TRex

However...
- Performance
- Inaccuracy

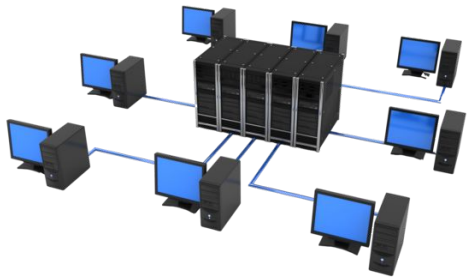
Hardware:
- FPGA-based
- ConnectX-5

However...
- Scalability and Cost
- Inflexibility

How can we combine the hardware and software generation benefits?

PIPO-TG: Programmable HW Traffic Generation

Problem, Challenges, State-of-art



How we can reproduce traffic patterns?

How inject this traffic into the network?

P4TG: 1 Tb/s Traffic Generation for Ethernet/IP Networks

STEFFEN LINDNER, MARCO HÄBERLE, (Graduate Student Member, IEEE), AND MICHAEL MENTH, (Senior Member, IEEE)

HyperTester: High-Performance Network Testing Driven by Programmable Switches

Dai Zhang, Yu Zhou, Zhaowei Xi, Yangyang Wang, Mingwei Xu, *Senior Member, IEEE*, and Jianping Wu, *Fellow, IEEE*

Limitations:
- Flexibility
- Server dependency
- not open-source
- limited traffic patterns

Network Testing

Software:
- Iperf
- TCP Replay
- TRex

However...
- Performance
- Inaccuracy

Hardware:
- FPGA-based
- ConnectX-5

However...
- Scalability and Cost
- Inflexibility

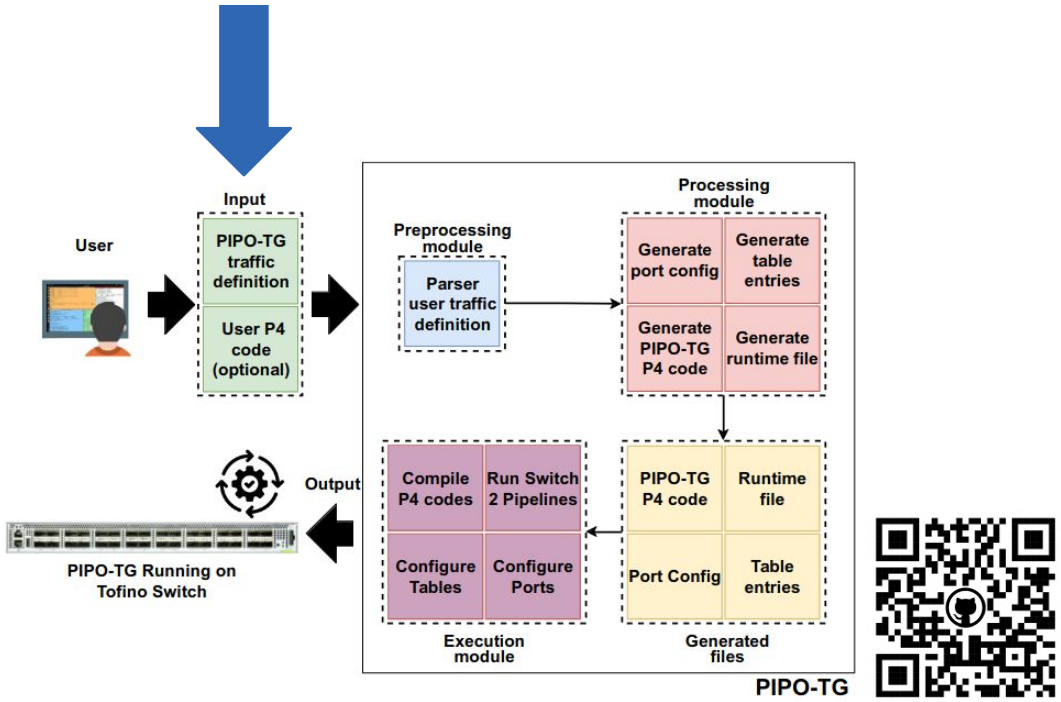
How can we combine the hardware and software generation benefits?

PIPO-TG: Parametrizable High-Performance Traffic Generator

PIPO-TG: Programmable HW Traffic Generation

High-performance flexible traffic generation using a P4/Tofino Switch

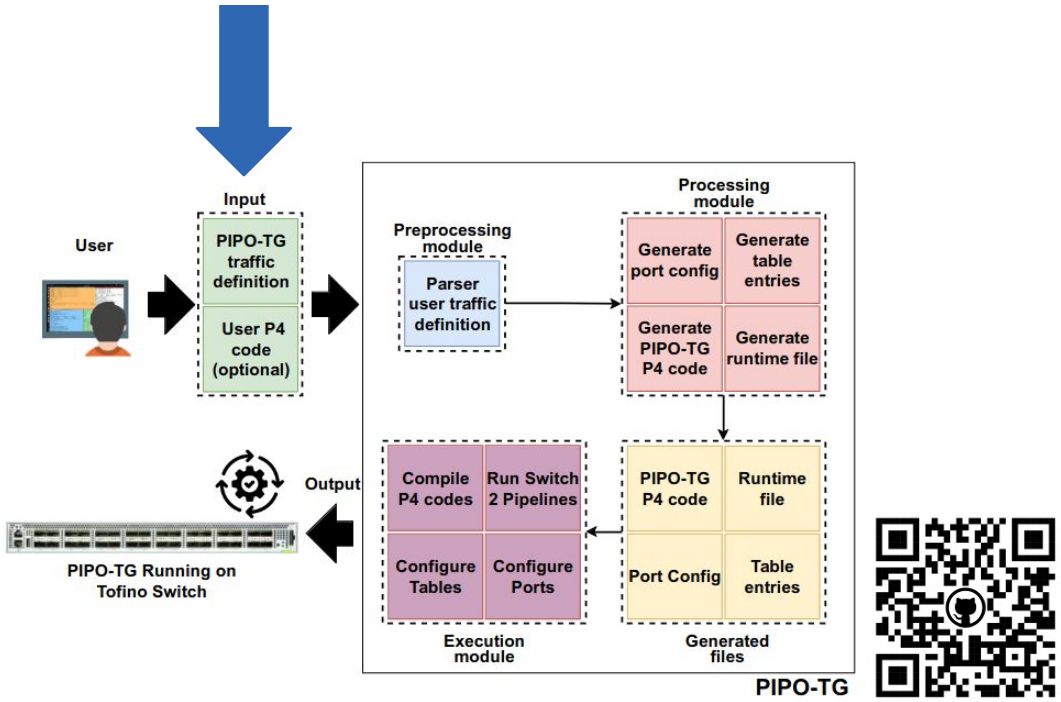
```
myTG = PipoGenerator()                                  #instatiate the traffic generator
myTG.addGenerationPort(68)                              #define the generation port
myTG.addOutputPort(5, 160, "100G")                      #physical port, port ID(D_P), portBW
myTG.addTroughput(max=500, min=100, interval=4)         #courve (a) - 4 seconds

myTG.addTroughput(max=500, min=100, interval=8)         #courve (a) - 8 seconds

myTG.addIP(ip_src="192.168.1.10", ip_dst="192.168.2.20")    #set teh IP headers source and destination address
myTG.generate()                                         #start traffic generation
```



PIPO-TG: Programmable HW Traffic Generation

High-performance flexible traffic generation using a P4/Tofino Switch

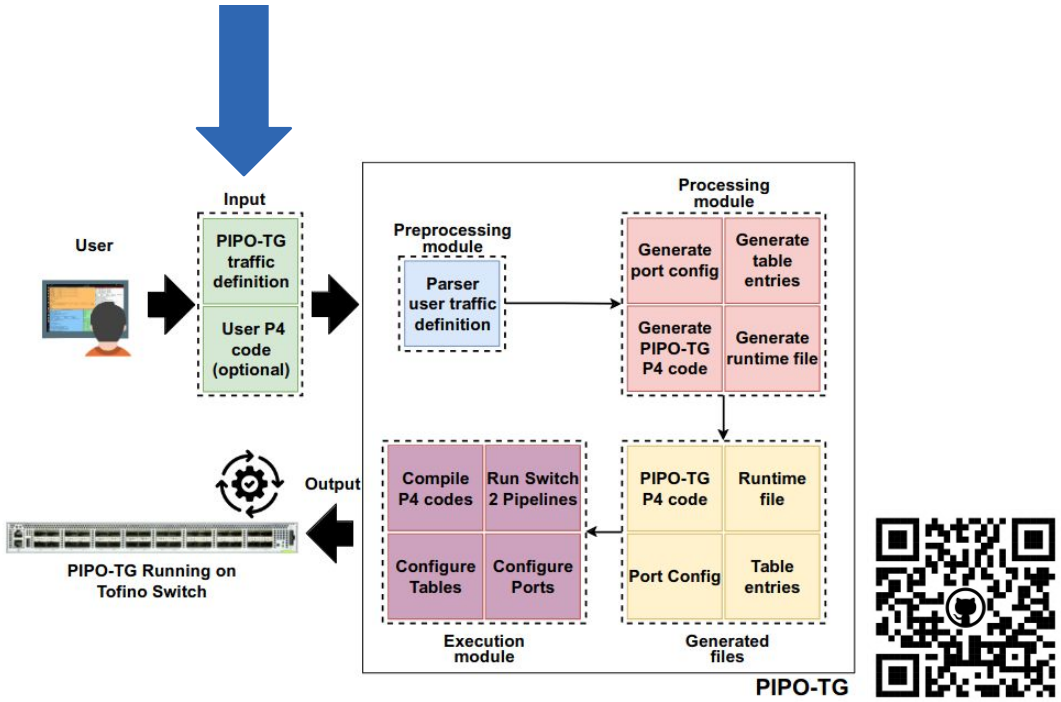
```
myTG = PipoGenerator()                                          #instatiate the traffic generator
myTG.addGenerationPort(68)                                      #define the generation port
myTG.addOutputPort(5, 160, "100G")                              #physical port, port ID(D_P), portBW
myTG.addTroughput(max=500, min=100, interval=4)                 #courve (a) - 4 seconds

myTG.addTroughput(max=500, min=100, interval=8)                 #courve (a) - 8 seconds

myTG.addIP(ip_src="192.168.1.10", ip_dst="192.168.2.20")        #set teh IP headers source and destination address
myTG.generate()                                                 #start traffic generation
```

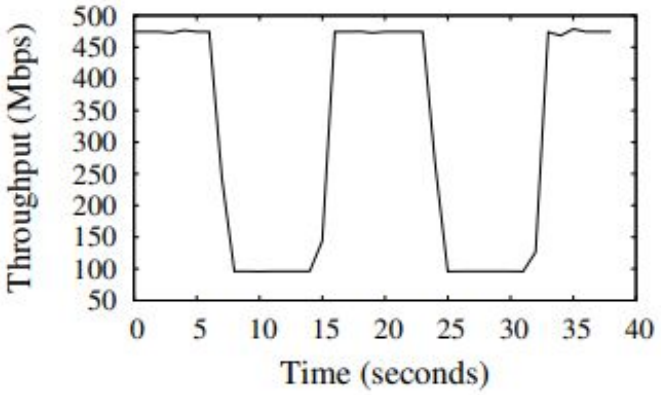


PIPO-TG: Programmable HW Traffic Generation

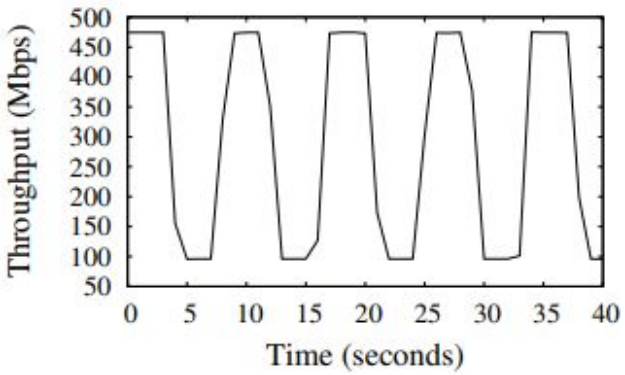
High-performance flexible traffic generation using a P4/Tofino Switch

```
myTG = PipoGenerator()                                          #instatiate the traffic generator
myTG.addGenerationPort(68)                                      #define the generation port
myTG.addOutputPort(5, 160, "100G")                              #physical port, port ID(D_P), portBW
myTG.addTroughput(max=500, min=100, interval=4)                 #courve (a) - 4 seconds

myTG.addTroughput(max=500, min=100, interval=8)                 #courve (a) - 8 seconds

myTG.addIP(ip_src="192.168.1.10", ip_dst="192.168.2.20")        #set teh IP headers source and destination address
myTG.generate()                                                 #start traffic generation
```





(b) 8-second square wave shape.

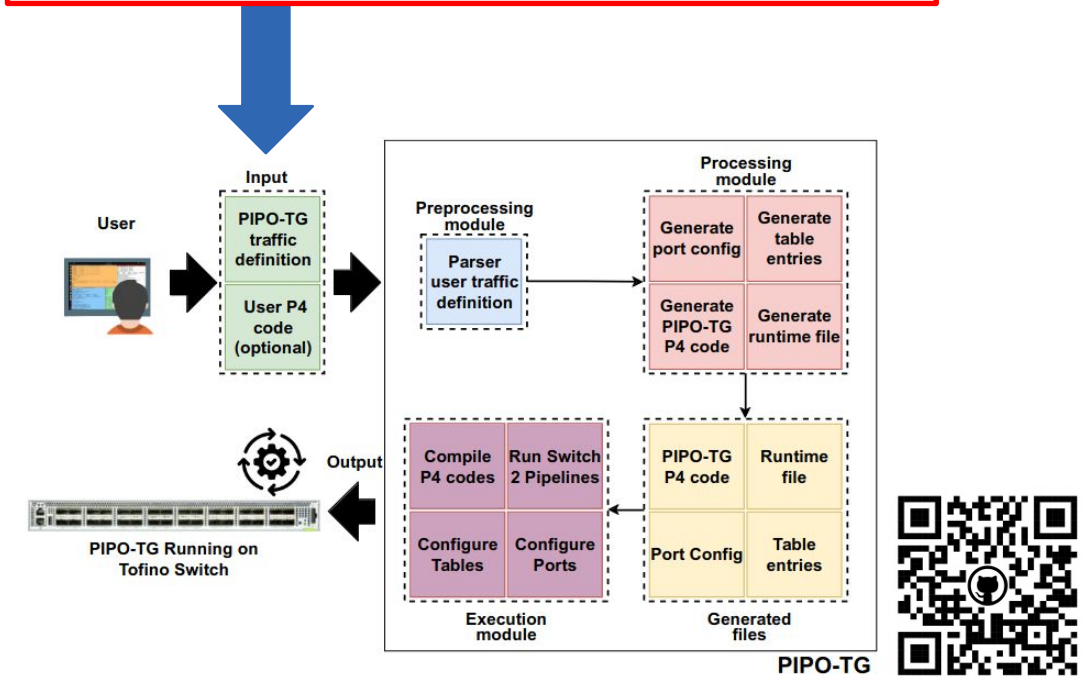


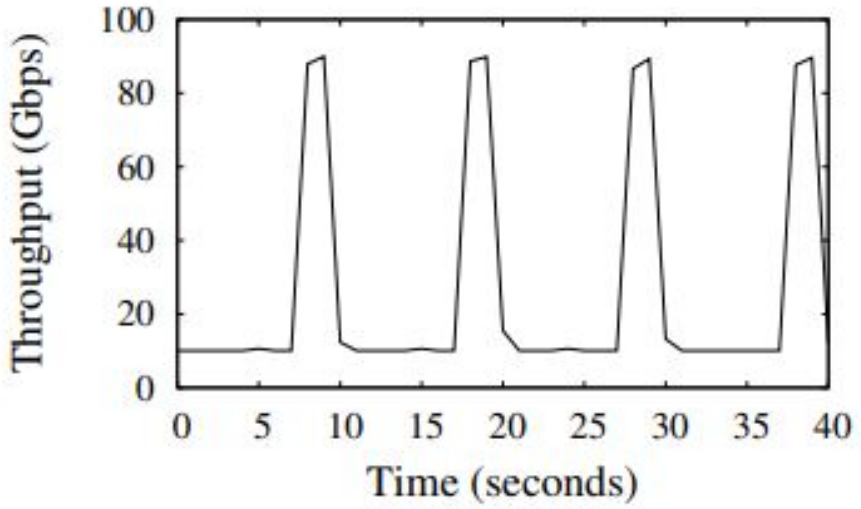
(a) 4-second square wave shape.

PIPO-TG: Programmable HW Traffic Generation

High-performance flexible traffic generation using a P4/Tofino Switch

```
myTG = PipoGenerator()                          #instatiate the traffic generator
myTG.addGenerationPort(68)                      #define the generation port
myTG.addOutputPort(5, 160, "100G")              #physical port, port ID(D_P), portBW
myGenerator.addIP(ip_dst = "10.0.0.2")             #set IP header with destination address

myTG.addVariance([10000, 90000], [8, 2])     #([Throughputs], [Intervals])
myTG.generate()                                 #start traffic generation
```

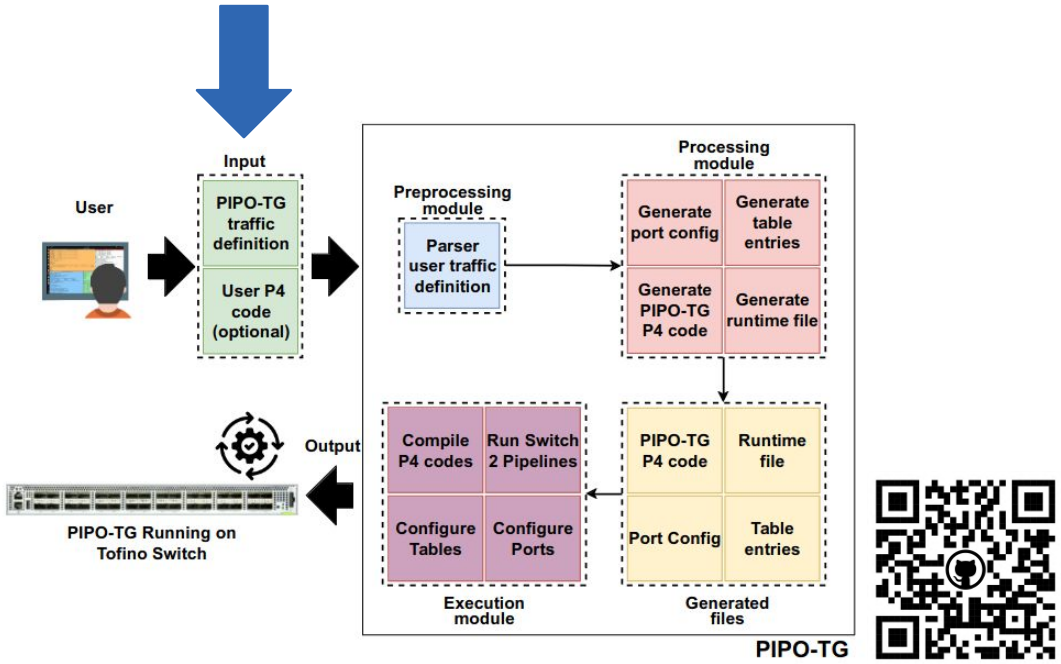


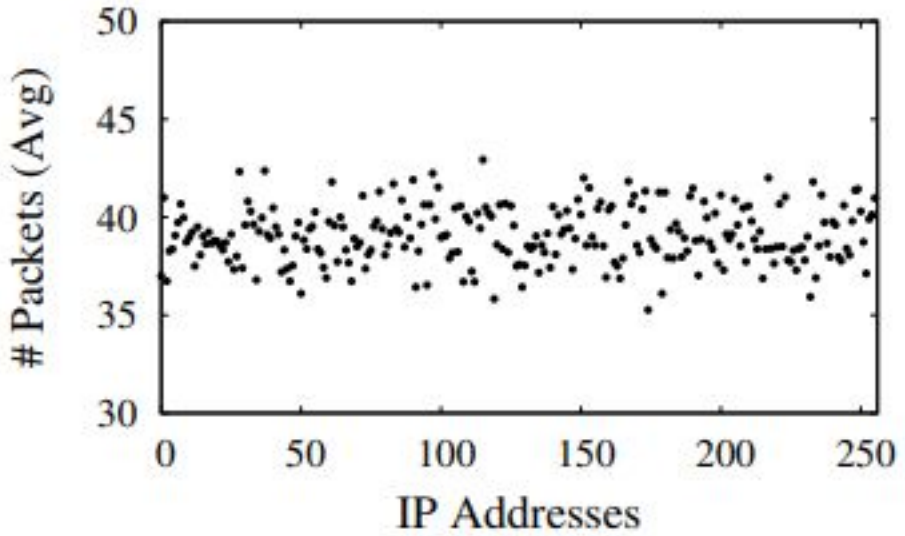


PIPO-TG: Programmable HW Traffic Generation

High-performance flexible traffic generation using a P4/Tofino Switch

```
myTG = PipoGenerator()                          #instatiate the traffic generator
myTG.addGenerationPort(68)                      #define the generation port
myTG.addOutputPort(5, 160, "100G")              #physical port, port ID(D_P), portBW

myTG.addThroughput(10000,"meter")          #define throughput(Mbps) and the type(port_shaping or meter)
myTG.addIP(src="192.168.1.0", srcRandom = True, srcMask = 24, dst="192.168.2.2")
myTG.generate()                                 #start traffic generation
```

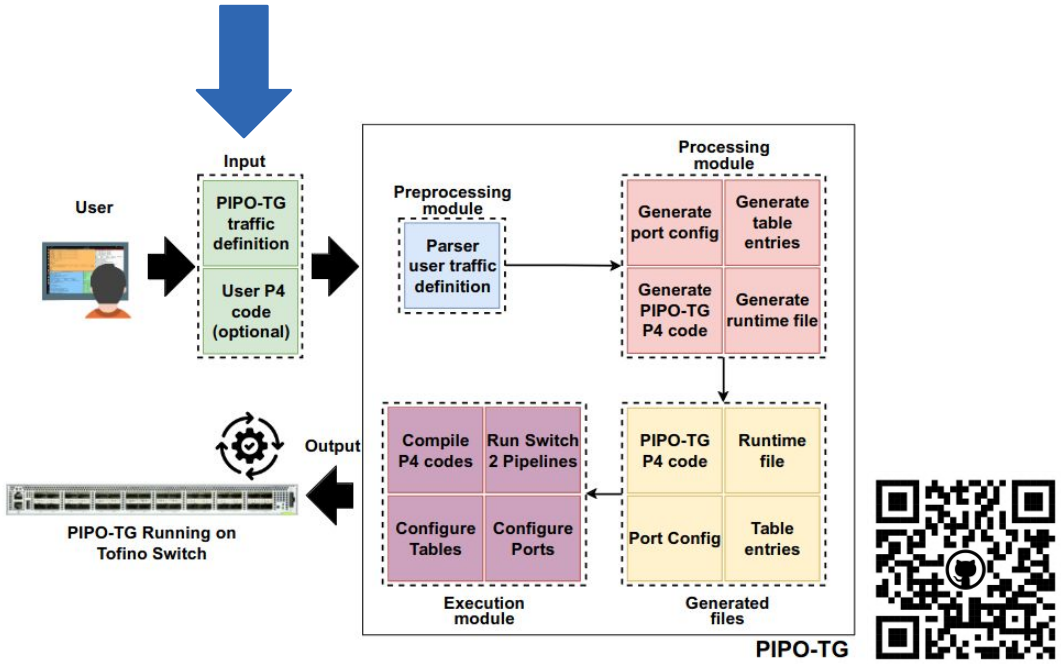




PIPO-TG: Programmable HW Traffic Generation

High-performance flexible traffic generation using a P4/Tofino Switch

```
myTG = PipoGenerator()                                  #ins
myTG.addGenerationPort(68)                              #def
myTG.addOutputPort(5, 160, "100G")                      #phy

myTG.addThroughput(10000,"meter")                  #define thro
myTG.addIP(src="192.168.1.0", srcRandom = True, srcMask
myTG.generate()                                         #sta
```



Main Features:

- Packet crafting;
- Throughput pattern definition;
- Common protocols;
- Custom protocols;
- Packet size definition and distribution;
- Workload Assay.
- User P4 code support

PIPO-TG: Programmable HW Traffic Generation

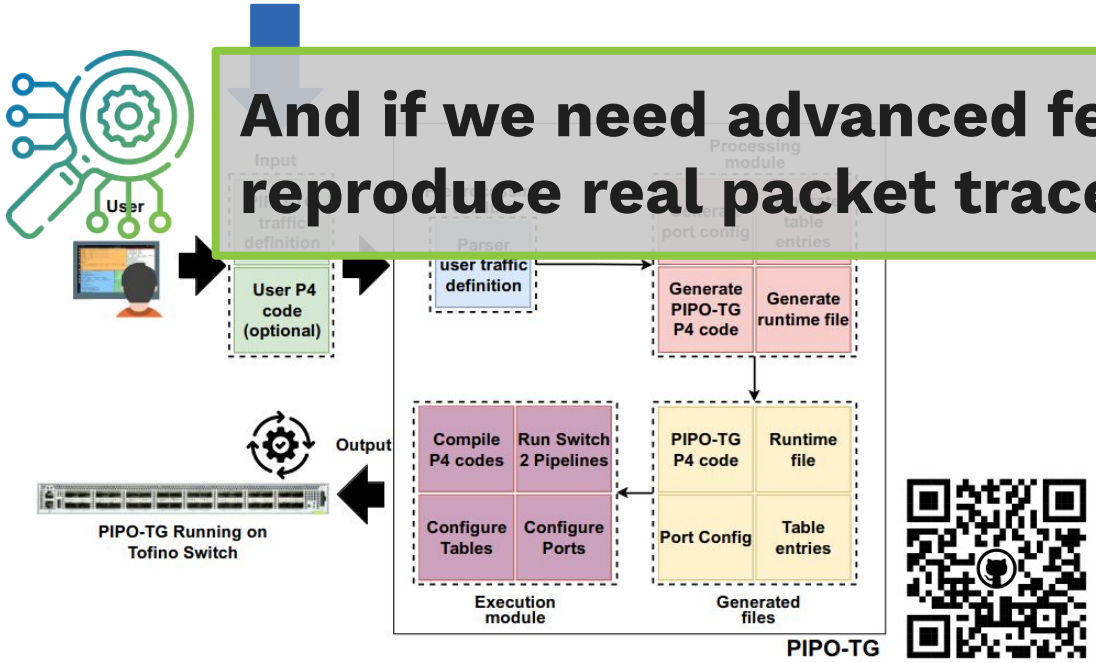
High-performance flexible traffic generation using a P4/Tofino Switch

```
myTG = PipoGenerator()                                    #ins
myTG.addGenerationPort(68)                                #def
myTG.addOutputPort(5, 160, "100G")                        #phy

myTG.addThroughput(10000,"meter")                  #define thro
myTG.addIP(src="192.168.1.0", srcRandom = True, srcMask
myTG.generate()                                           #sta
```

Main Features:

- Packet crafting;
- Throughput pattern definition;
- Common protocols;
- Custom protocols;
- Packet 2 [illegible] distribution;
- Workload Assay.
- User P4 code support

And if we need advanced features like TCP connections and reproduce real packet traces?



PIPO-TG: Programmable HW Traffic Generation

High-performance flexible traffic generation using a P4/Tofino Switch

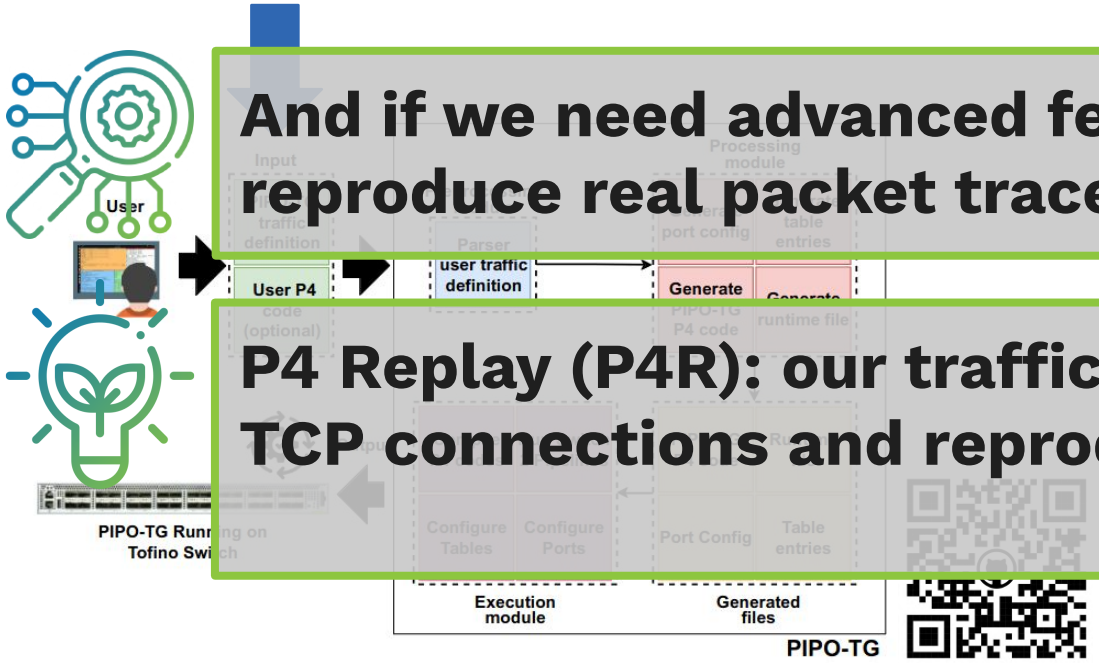
```
myTG = PipoGenerator()                                  #ins
myTG.addGenerationPort(68)                              #def
myTG.addOutputPort(5, 160, "100G")                      #phy

myTG.addThroughput(10000,"meter")                  #define thro
myTG.addIP(src="192.168.1.0", srcRandom = True, srcMask
myTG.generate()                                         #sta
```

Main Features:

- Packet crafting;
- Throughput pattern definition;
- Common protocols;
- Custom protocols;
- Packet 2 and finish time distribution;
- Workload Assay.
- User P4 code support

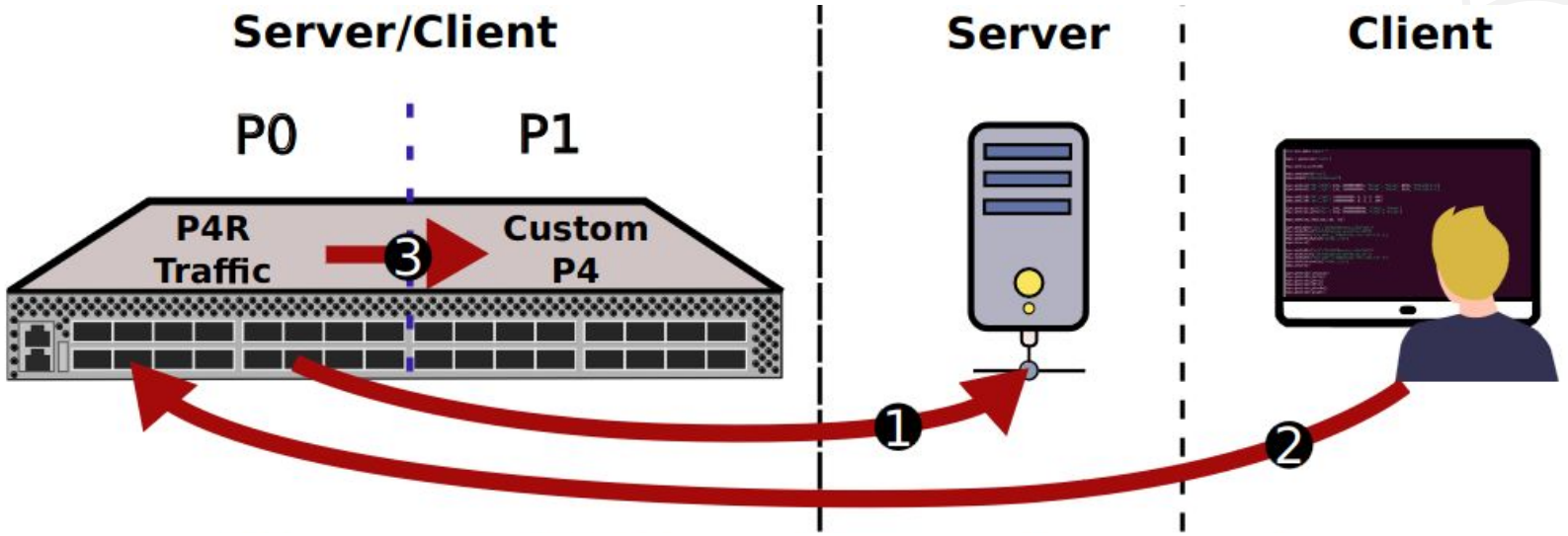
And if we need advanced features like TCP connections and reproduce real packet traces?

P4 Replay (P4R): our traffic generated designed to establish real TCP connections and reproduce PCAPs using the P4 switch



P4 Replay (P4R)

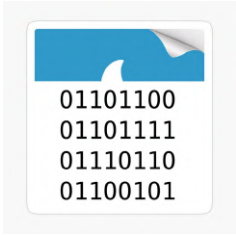
Operation modes



1. **Client Mode:** P4R can reproduce PCAPs or establish TCP connections with a connected server.
2. **Server Mode:** P4R responds to TCP connections from connected clients.
3. **Internal Mode:** P4R can send packet traces or TCP connections to test an user P4 code running in parallel, in another pipeline.

P4 Replay (P4R)

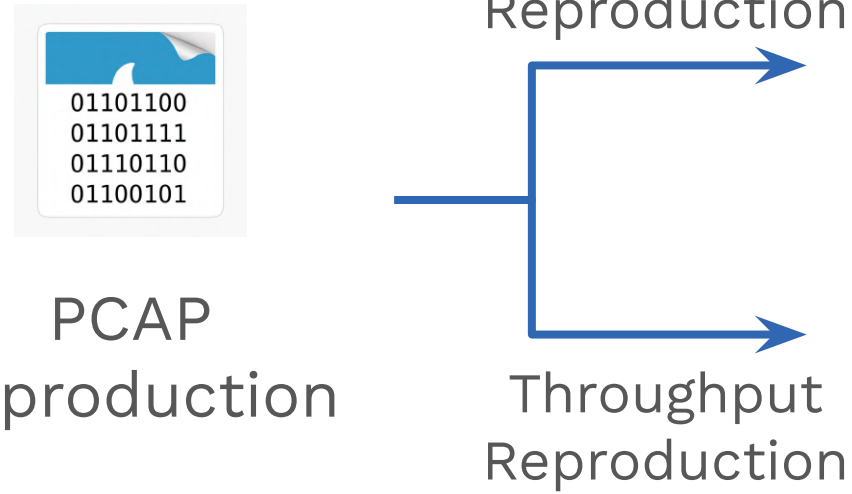
P4R capabilities



PCAP
Reproduction

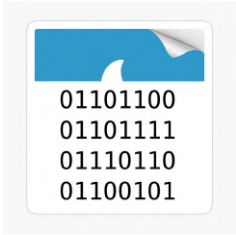
P4 Replay (P4R)

P4R capabilities



P4 Replay (P4R)

P4R capabilities



PCAP Reproduction

Original Reproduction

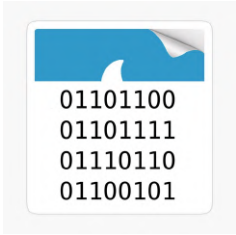
- High accuracy in inter-packet arrival times

Throughput Reproduction

- High throughput, reproducing the PCAP in a repetition mode up to 100 Gbps per-port

P4 Replay (P4R)

P4R capabilities



PCAP Reproduction

Original Reproduction

- High accuracy in inter-packet arrival times

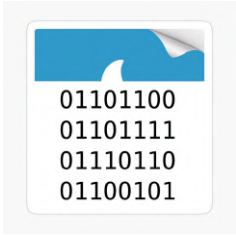
Throughput Reproduction

- High throughput, reproducing the PCAP in a repetition mode up to 100 Gbps per-port

Stateful Connections

P4 Replay (P4R)

P4R capabilities



PCAP Reproduction

Original Reproduction

- High accuracy in inter-packet arrival times

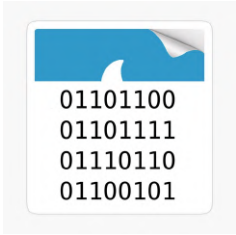
Throughput Reproduction

- High throughput, reproducing the PCAP in a repetition mode up to 100 Gbps per-port

Stateful Connections

P4 Replay (P4R)

P4R capabilities



PCAP Reproduction

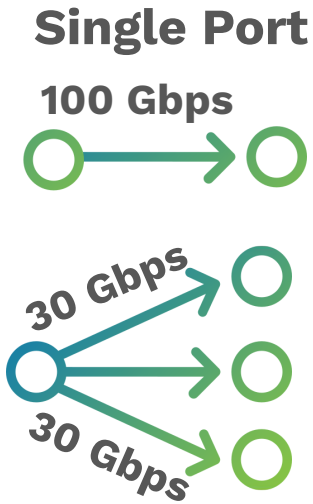
Original Reproduction

- High accuracy in inter-packet arrival times

Throughput Reproduction

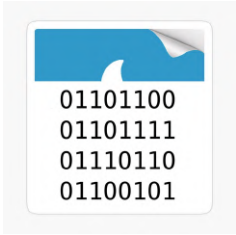
- High throughput, reproducing the PCAP in a repetition mode up to 100 Gbps per-port

Stateful Connections



P4 Replay (P4R)

P4R capabilities



PCAP Reproduction

Original Reproduction

- High accuracy in inter-packet arrival times

Throughput Reproduction

- High throughput, reproducing the PCAP in a repetition mode up to 100 Gbps per-port

Stateful Connections

Single Port

100 Gbps

30 Gbps

30 Gbps

30 Gbps

Multiple Ports

30 Gbps

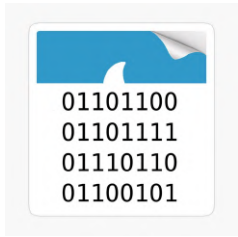
30 Gbps

30 Gbps

30 Gbps

P4 Replay (P4R)

P4R capabilities



PCAP Reproduction

Original Reproduction

- High accuracy in inter-packet arrival times

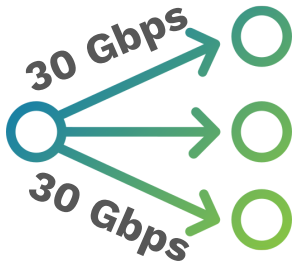
Throughput Reproduction

- High throughput, reproducing the PCAP in a repetition mode up to 100 Gbps per-port

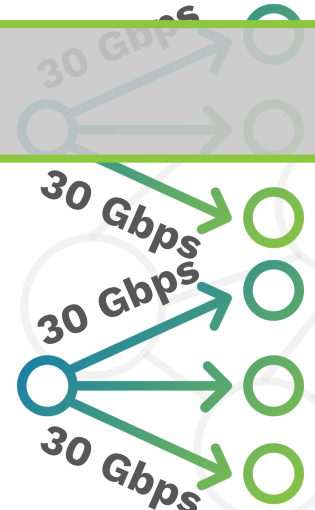
Which packet traces can we reproduce?

Stateful Connections

Single Port

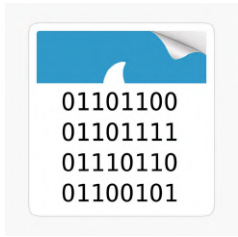


Multiple Ports



P4 Replay (P4R)

P4R capabilities



PCAP Reproduction

Original Reproduction

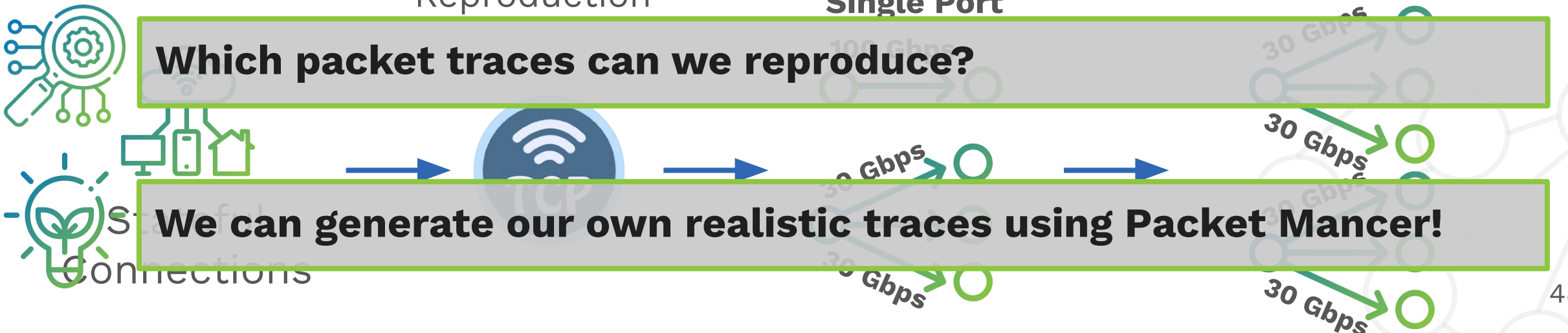
- High accuracy in inter-packet arrival times

Throughput Reproduction

- High throughput, reproducing the PCAP in a repetition mode up to 100 Gbps per-port

Which packet traces can we reproduce?

We can generate our own realistic traces using Packet Mancer!



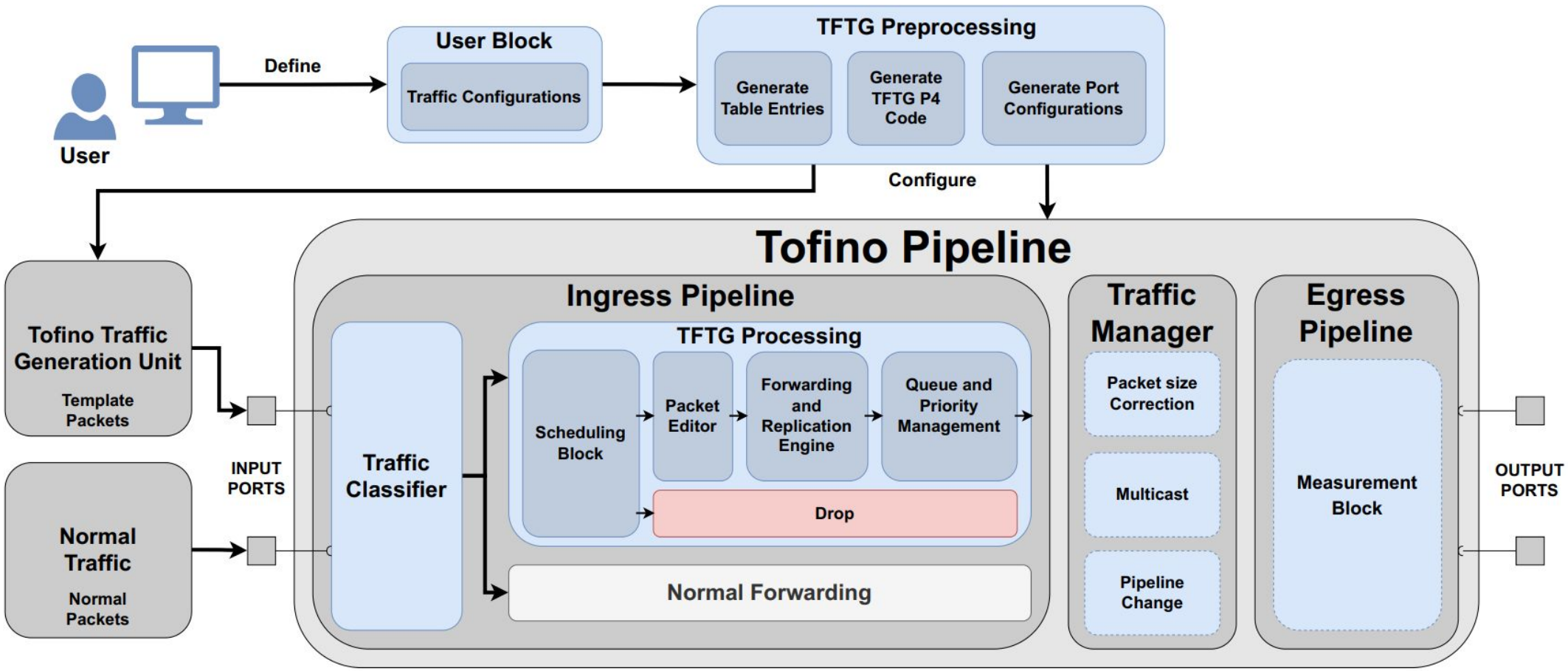
SMARTNESS

Use cases & Further Reading

(Selected) for 6G research

Time-fidelity Traffic Generation

Work-in-progress

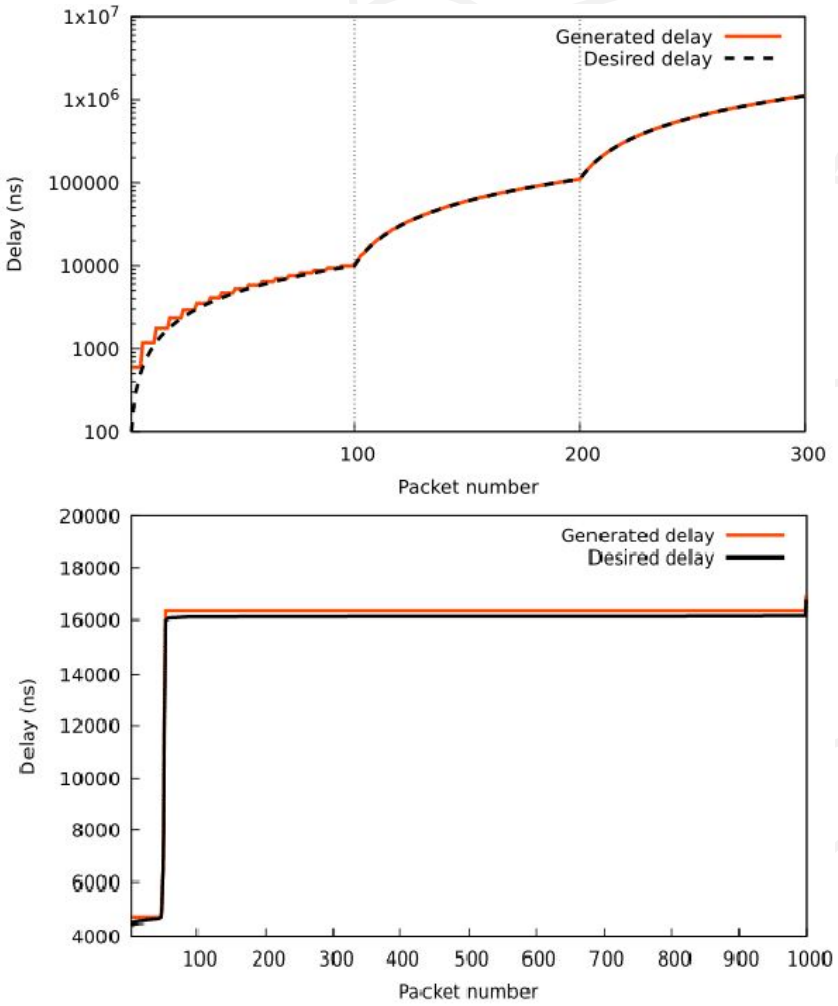


Example of PIPO-TG use case

Reproducing TSN delays with PIPO-TG

Real delay measurements from EU SNS Deterministic6G project:
(www.github.com/DETERMINISTIC6G/deterministic6g_data)

```
<histogram>
<bin low="1500ns">3</bin>
<bin low="2000ns">13</bin>
<bin low="2500ns">20</bin>
<bin low="1000ns">0</bin>
</histogram>
```



Example of P7 use case

Path-aware networking IN a Tofino BoX (PINT-BoX)

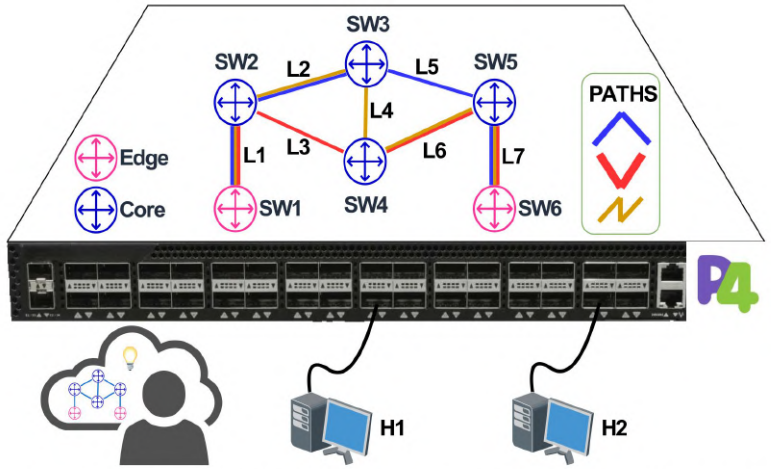


TABLE I

TABLE WITH THE TTL WEIGHTS OF EACH CORRESPONDING SWITCH

TTL Weight	SW1	SW2	SW3	SW4	SW5	SW6
Request	4	5	6	7	8	9
Reply	10	11	12	13	14	15

TABLE II

CUSTOMIZED NETWORK LINK METRICS FOR THE DEMO IN EACH PATH.

Path	Link	Bandwidth	Loss	Latency	Cumulative TTL	
					Request	Reply
Path 1	L5	10G	3%	5 ms	32	62
Path 2	L3	10G	1%	15 ms	33	63
Path 3	L4	10G	2%	10 ms	39	75

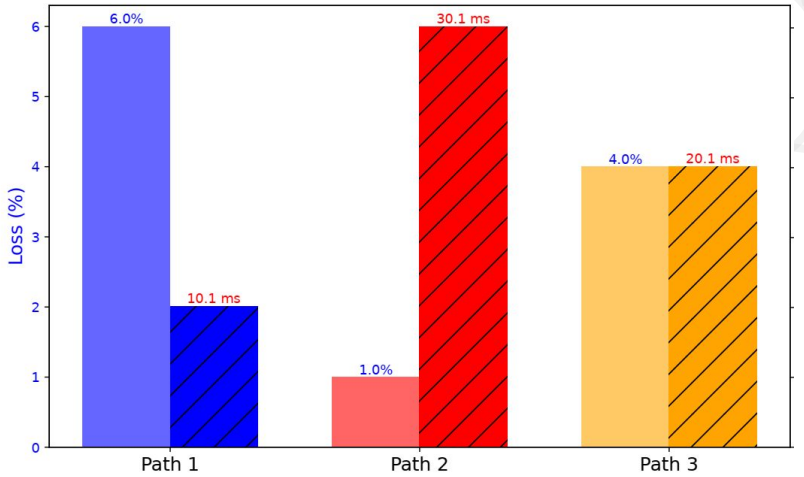
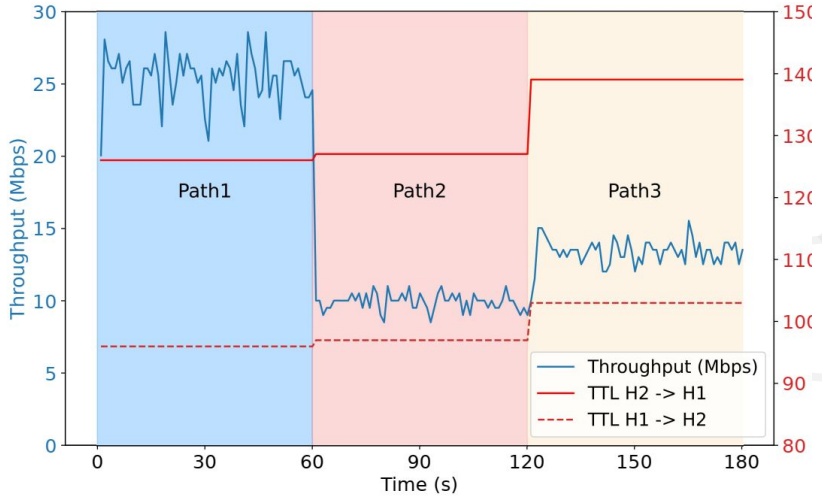
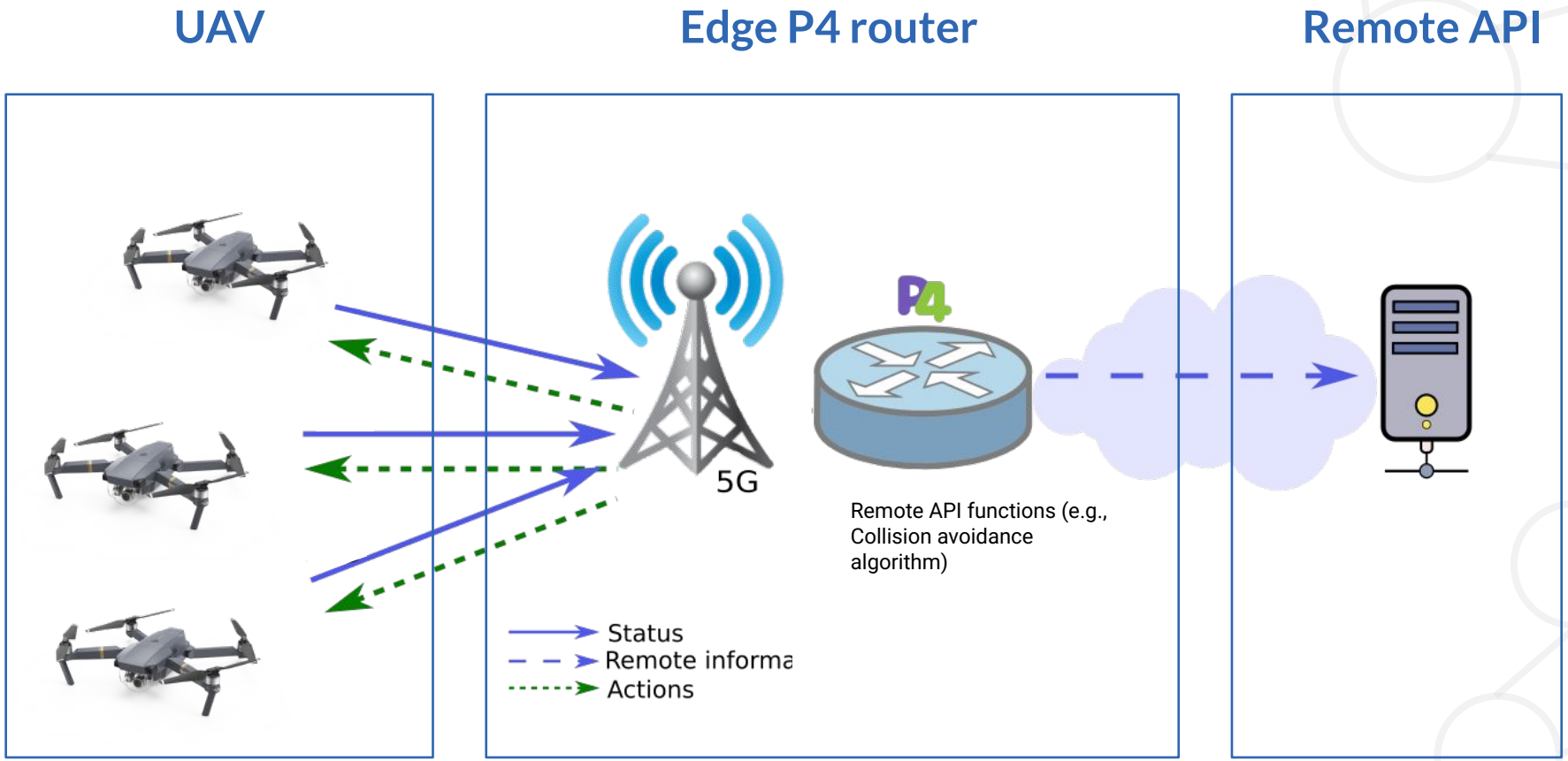


Fig. 2. Loss and Latency measurements over different path



Example of P7 use case

In-network collision avoidance



F. E. R. Cesen et al., "**Harnessing P4 for In-Network Unmanned Aerial Vehicle Collision Avoidance**," IEEE NetSoft 2025

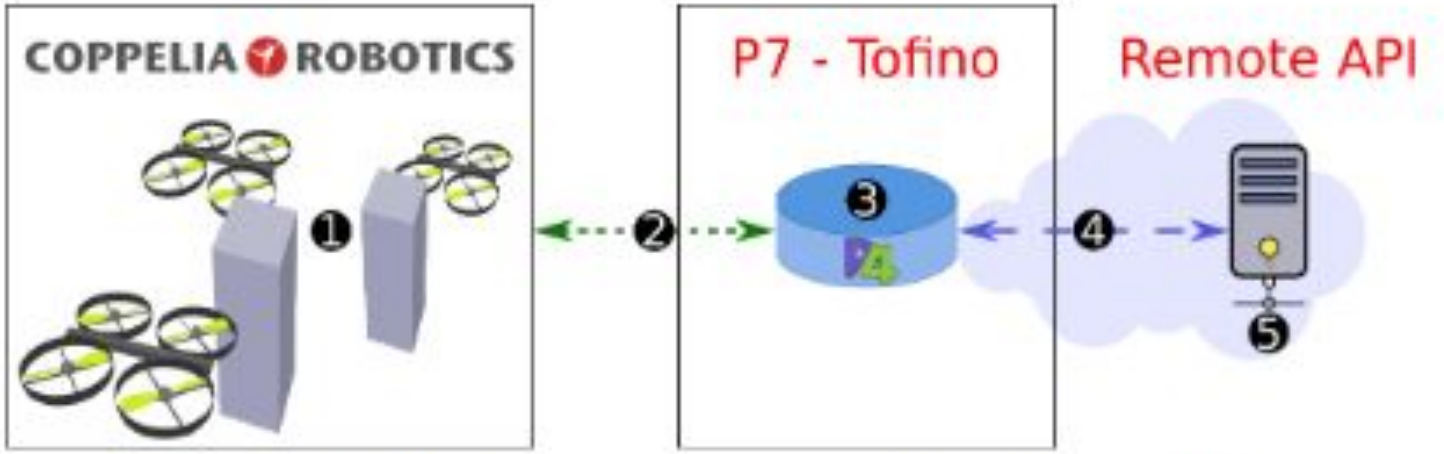
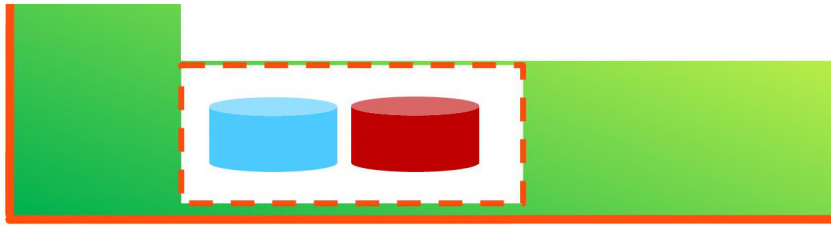
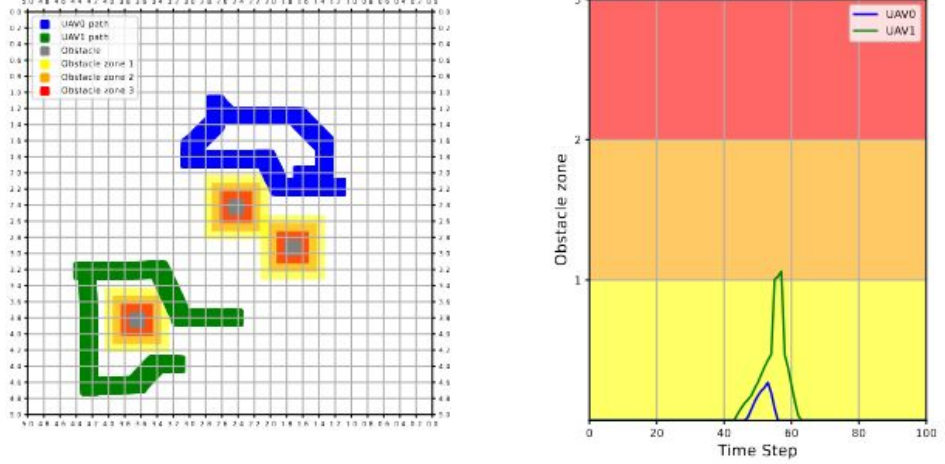


Fig. 5: Collision avoidance algorithm + P7.

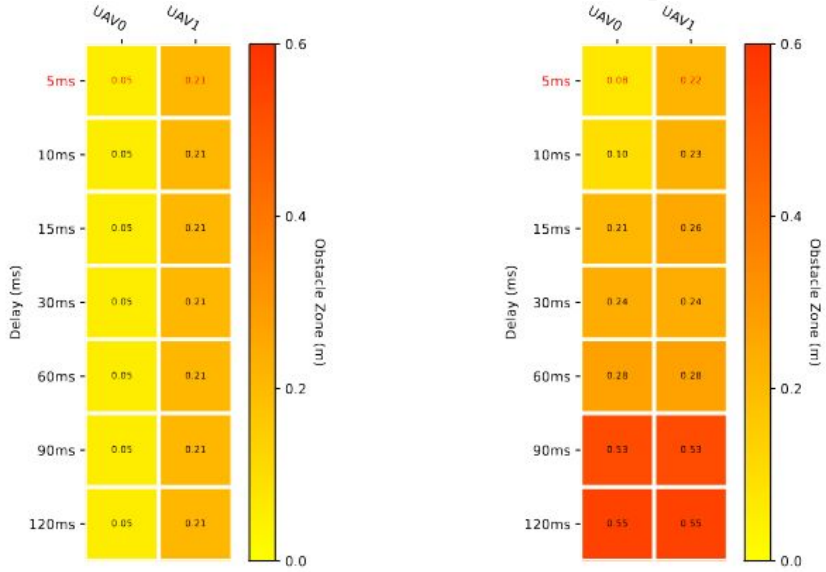
HW





(a) UAV path. (b) Collision detection.

Fig. 6: In-network collision avoidance scenario.

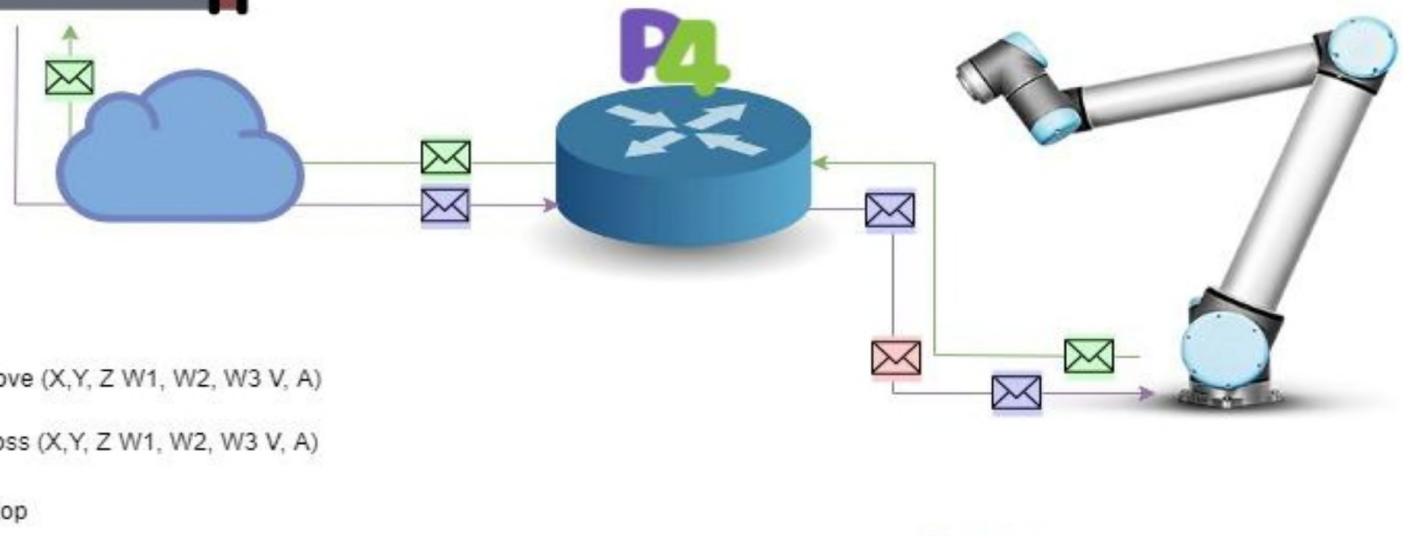


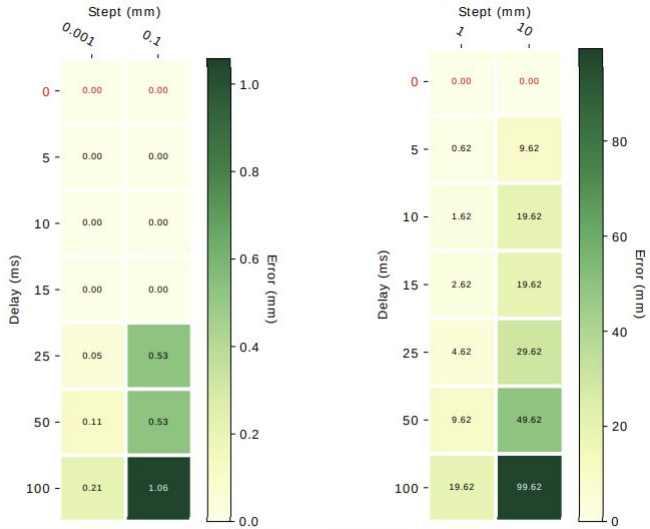
(a) In-network control. (b) Remote API control.

Fig. 9: In-network vs. remote control. Velocity $= 0.66m/s$.

F. E. R. Cesen et al., "**Towards Low Latency Industrial Robot Control in Programmable Data Planes**," IEEE NetSoft 2020

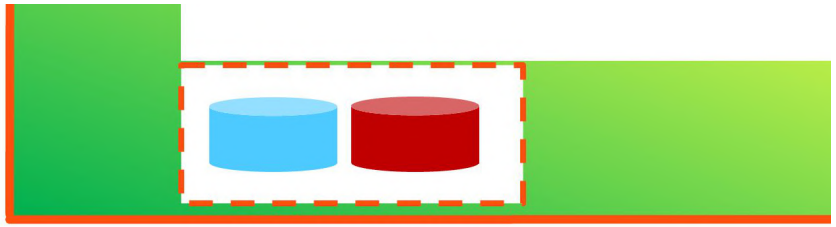
SMARTNESS





(a) Steps of 0.001 - 0.1 mm (b) Steps of 1 - 10 mm

HW



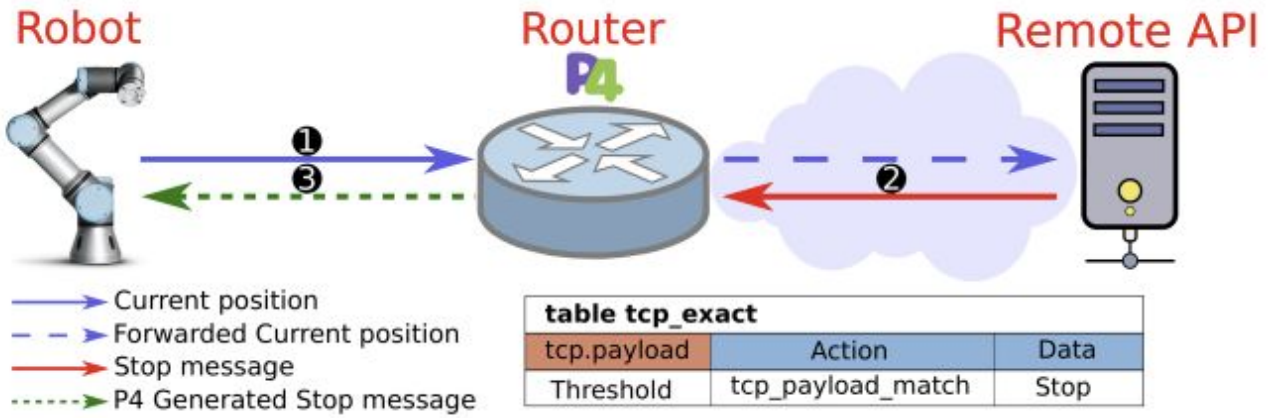
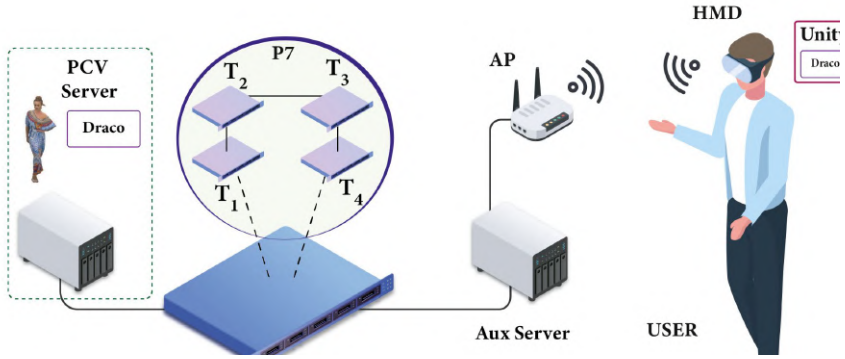
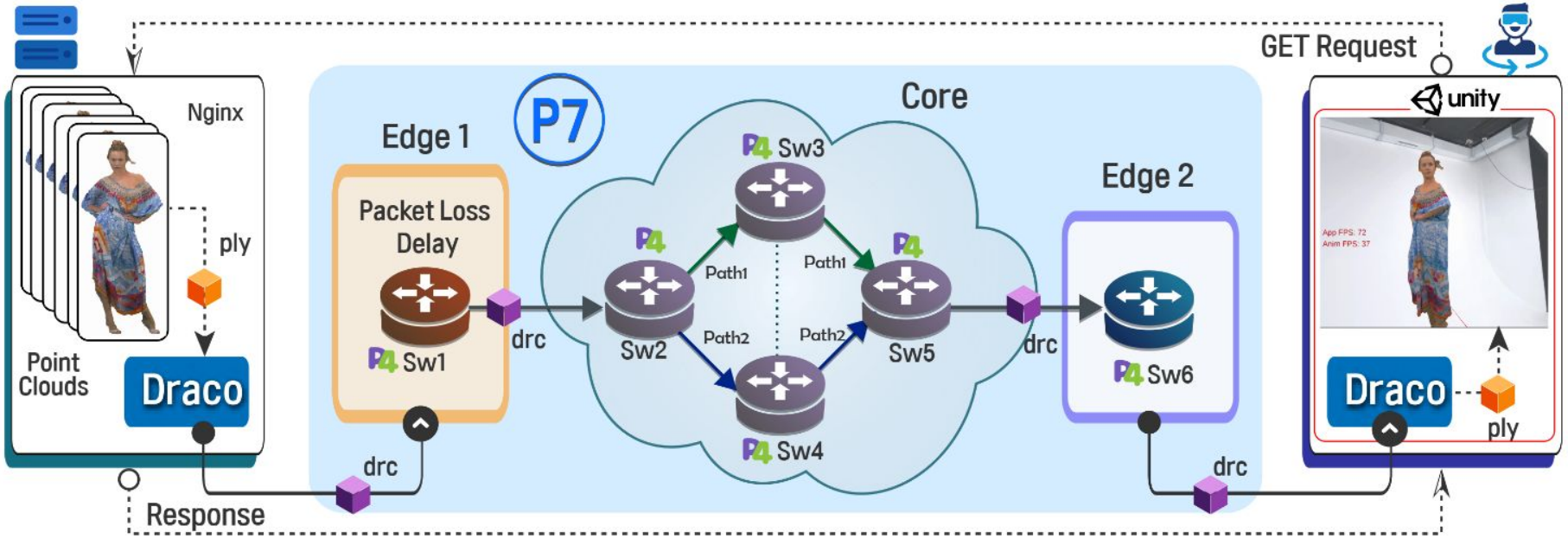


table tcp_exact		
tcp.payload	Action	Data
Threshold	tcp_payload_match	Stop

Holographic-Type Communication (HTC)

P7-based Experiment Testbed setup



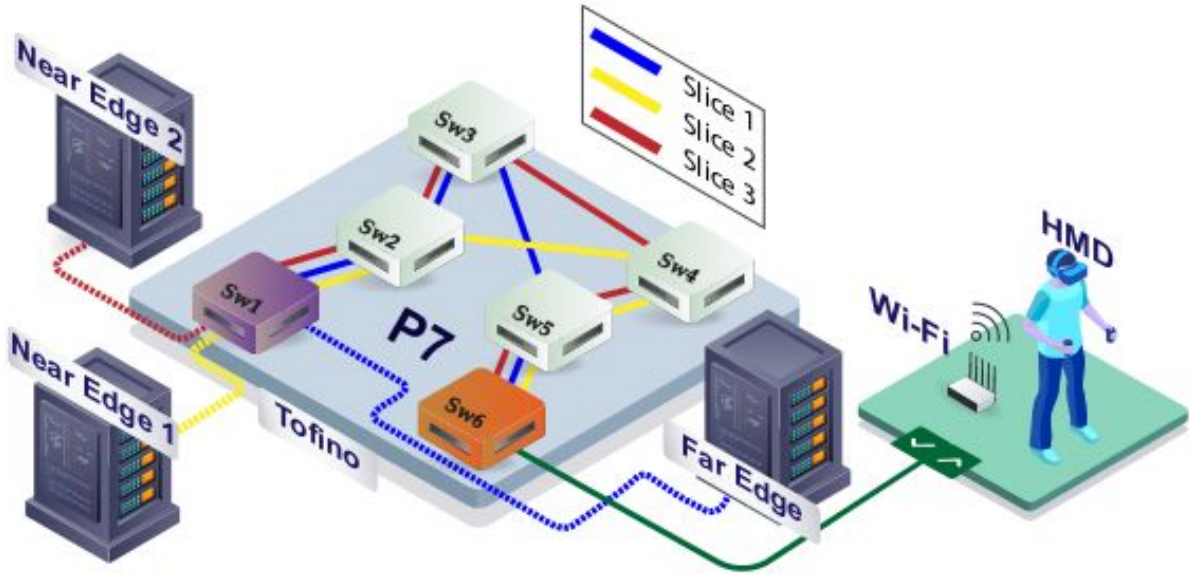


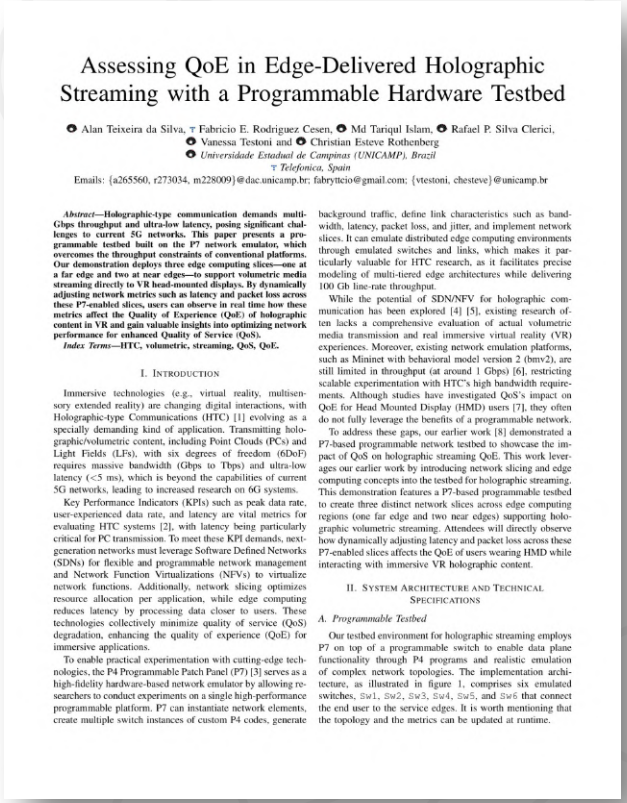
Scenario	PL (%) DL(ms)	QoE Rating
Ideal	0 0	Q1
Minimal PL	0.1 0	Q1
Moderate DL	0 4	Q2
High DL	0 10	Q3
Low PL, High DL	0.1 10	Q3
High PL	2.0 0	Q2

A. Teixeira et al. **Assessing QoE in Edge-Delivered Holographic Streaming with a Programmable Hardware Testbed.** In IEEE NetSoft'25 demos.

Key points

- Holographic video streaming QoE testbed
- Support for Point Cloud directly in the HMD
- Emulated topologies with P7





Assessing QoE in Edge-Delivered Holographic Streaming with a Programmable Hardware Testbed

Alan Teixeira da Silva, Fabricio E. Rodriguez Cesen, Md Tariqul Islam, Rafael P. Silva Clerici, Vanessa Testoni and Christian Esteve Rothenberg
Universidade Estadual de Campinas (UNICAMP), Brazil
Telefonica, Spain
Emails: {a265560, r273034, m228009}@dac.unicamp.br; fabryttcio@gmail.com; {vtestoni, chesteve}@unicamp.br

Abstract—**Holographic-type communication demands multi-Gbps throughput and ultra-low latency, posing significant challenges to current 5G networks. This paper presents a programmable testbed built on the P7 network emulator, which overcomes the throughput constraints of conventional platforms. Our demonstration deploys three edge computing slices—one at a far edge and two at near edges—to support volumetric media streaming directly to VR head-mounted displays. By dynamically adjusting network metrics such as latency and packet loss across these P7-enabled slices, users can observe in real time how these metrics affect the Quality of Experience (QoE) of holographic content in VR and gain valuable insights into optimizing network performance for enhanced Quality of Service (QoS).**

Index Terms—**HTC, volumetric, streaming, QoS, QoE.**

I. Introduction

Immersive technologies (e.g., virtual reality, multisensory extended reality) are changing digital interactions, with Holographic-type Communications (HTC) [1] evolving as a specially demanding kind of application. Transmitting holographic/volumetric content, including Point Clouds (PCs) and Light Fields (LFs), with six degrees of freedom (6DoF) requires massive bandwidth (Gbps to Tbps) and ultra-low latency (<5 ms), which is beyond the capabilities of current 5G networks, leading to increased research on 6G systems.

Key Performance Indicators (KPIs) such as peak data rate, user-experienced data rate, and latency are vital metrics for evaluating HTC systems [2], with latency being particularly critical for PC transmission. To meet these KPI demands, next-generation networks must leverage Software Defined Networks (SDNs) for flexible and programmable network management and Network Function Virtualizations (NFVs) to virtualize network functions. Additionally, network slicing optimizes resource allocation per application, while edge computing reduces latency by processing data closer to users. These technologies collectively minimize quality of service (QoS) degradation, enhancing the quality of experience (QoE) for immersive applications.

To enable practical experimentation with cutting-edge technologies, the P4 Programmable Patch Panel (P7) [3] serves as a high-fidelity hardware-based network emulator by allowing researchers to conduct experiments on a single high-performance programmable platform. P7 can instantiate network elements, create multiple switch instances of custom P4 codes, generate background traffic, define link characteristics such as bandwidth, latency, packet loss, and jitter, and implement network slices. It can emulate distributed edge computing environments through emulated switches and links, which makes it particularly valuable for HTC research, as it facilitates precise modeling of multi-tiered edge architectures while delivering 100 Gb line-rate throughput.

While the potential of SDN/NFV for holographic communication has been explored [4] [5], existing research often lacks a comprehensive evaluation of actual volumetric media transmission and real immersive virtual reality (VR) experiences. Moreover, existing network emulation platforms, such as Mininet with behavioral model version 2 (bmv2), are still limited in throughput (at around 1 Gbps) [6], restricting scalable experimentation with HTC's high bandwidth requirements. Although studies have investigated QoS's impact on QoE for Head Mounted Display (HMD) users [7], they often do not fully leverage the benefits of a programmable network.

To address these gaps, our earlier work [8] demonstrated a P7-based programmable network testbed to showcase the impact of QoS on holographic streaming QoE. This work leverages our earlier work by introducing network slicing and edge computing concepts into the testbed for holographic streaming. This demonstration features a P7-based programmable testbed to create three distinct network slices across edge computing regions (one far edge and two near edges) supporting holographic volumetric streaming. Attendees will directly observe how dynamically adjusting latency and packet loss across these P7-enabled slices affects the QoE of users wearing HMD while interacting with immersive VR holographic content.

II. System Architecture and Technical Specifications

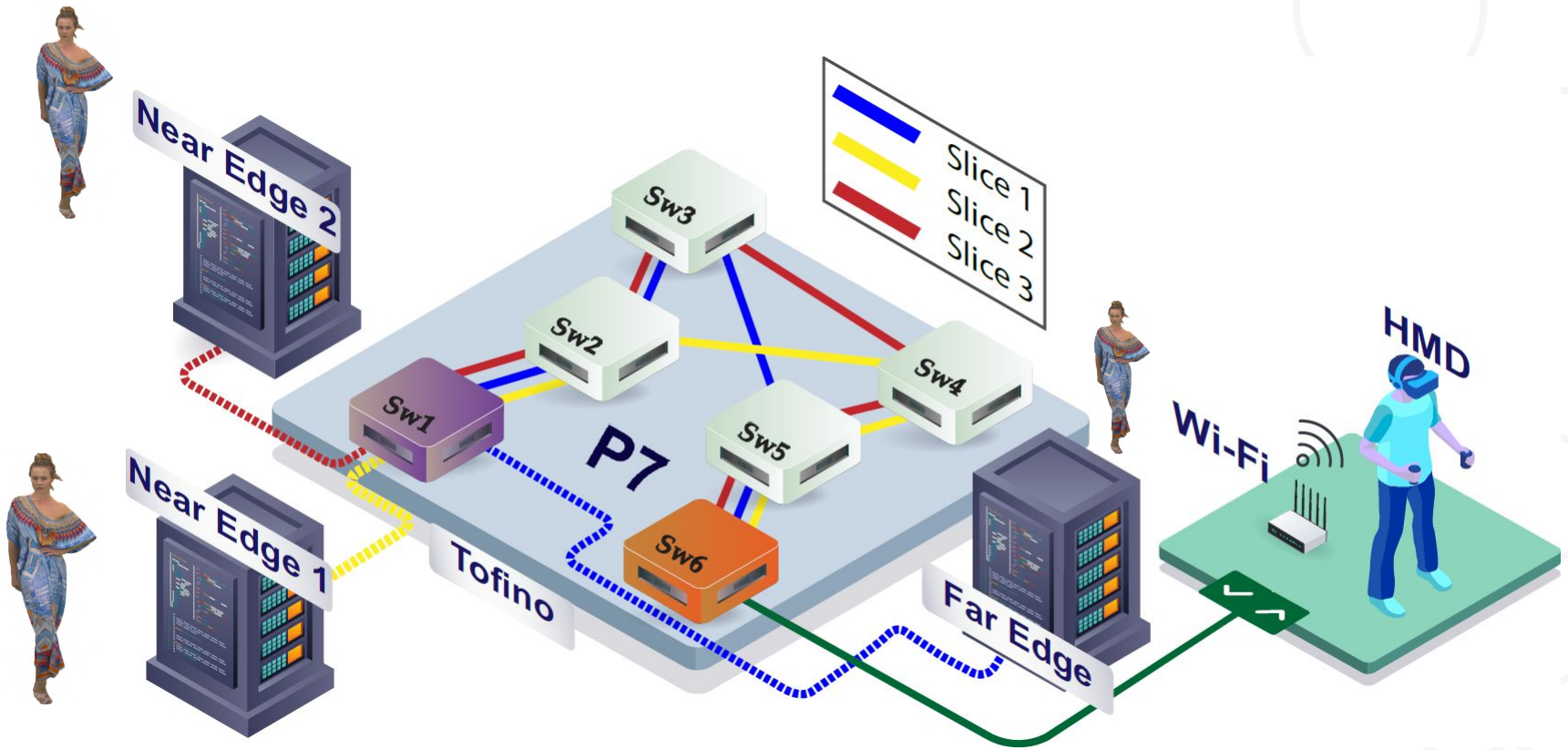
A. Programmable Testbed

Our testbed environment for holographic streaming employs P7 on top of a programmable switch to enable data plane functionality through P4 programs and realistic emulation of complex network topologies. The implementation architecture, as illustrated in figure 1, comprises six emulated switches, `Sw1`, `Sw2`, `Sw3`, `Sw4`, `Sw5`, and `Sw6` that connect the end user to the service edges. It is worth mentioning that the topology and the metrics can be updated at runtime.

Customer Premises — **BS MEC - Access Cloud - PoP DC** **Edge** — **Cloud DCs Core**

Volumetric media streaming QoS / QoE

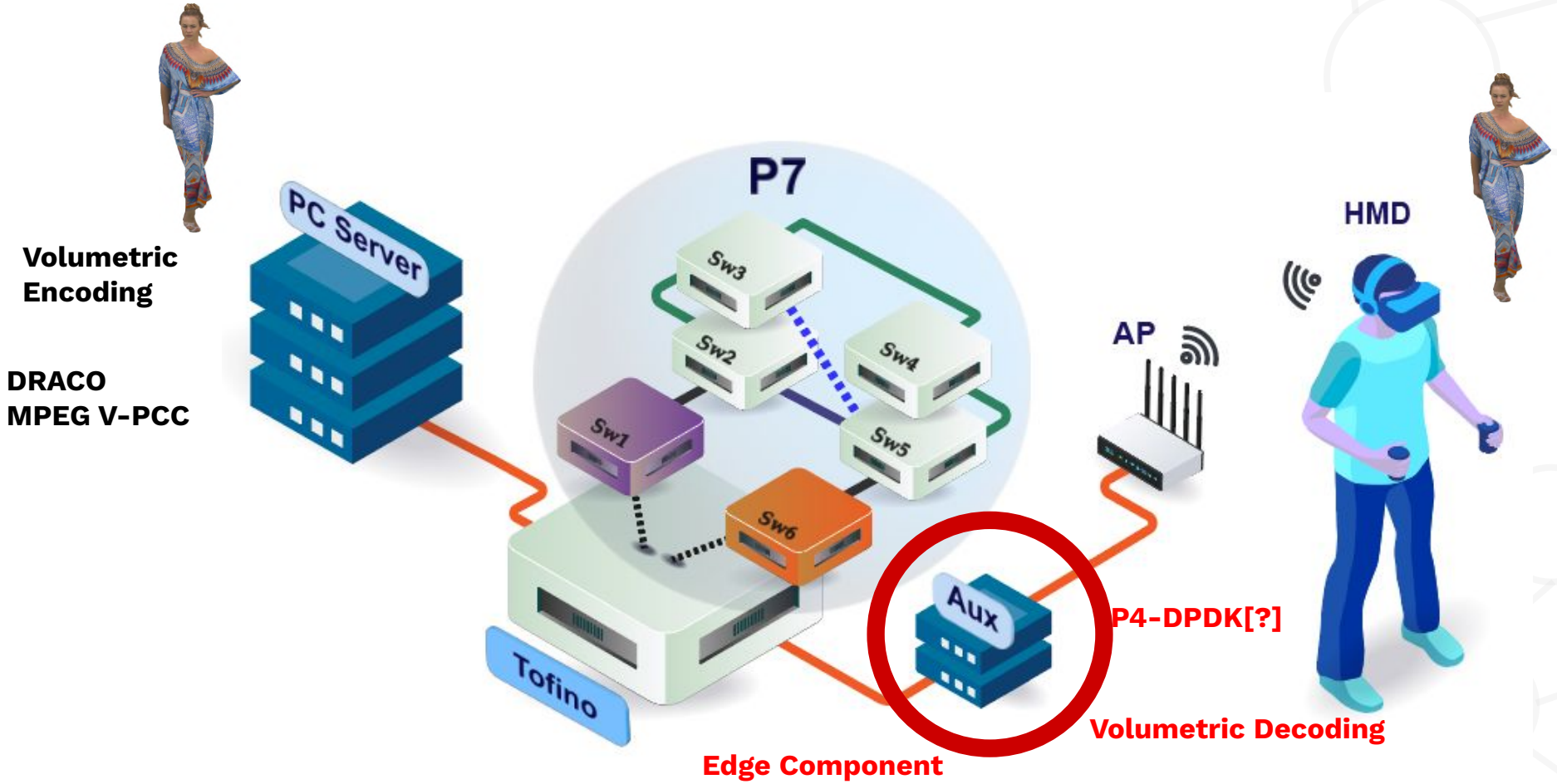
Immersive Virtual Reality in HMD



Assessing QoE in Edge-Delivered Holographic Streaming with a Programmable Hardware Testbed. In IEEE **NetSoft'25 demos.**

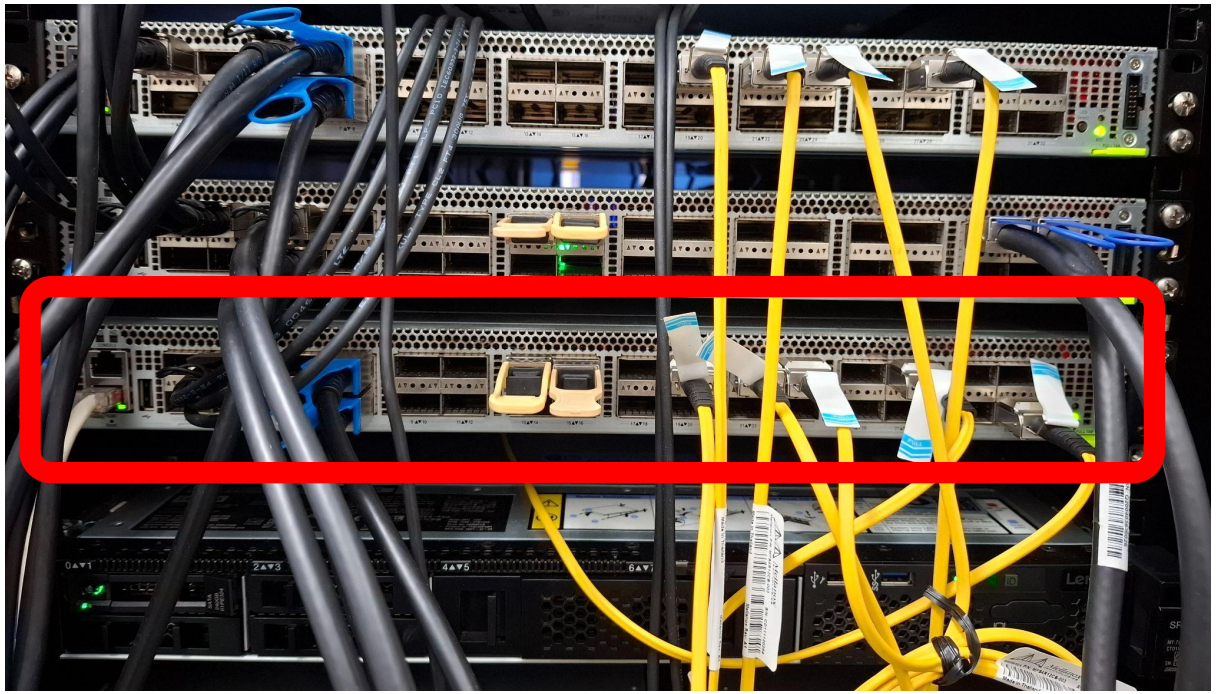
Volumetric media streaming QoS / QoE

Immersive Virtual Reality in HMD



Volumetric media streaming QoS / QoE

HW setup



TOFINO + P7

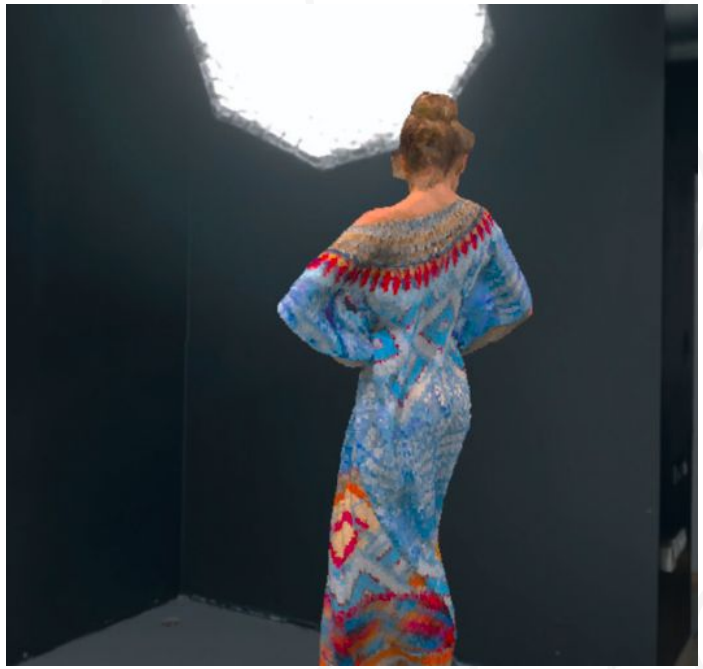
Meta Quest 3

Volumetric media streaming QoS / QoE

Immersive Virtual Reality in HMD

Front

Side



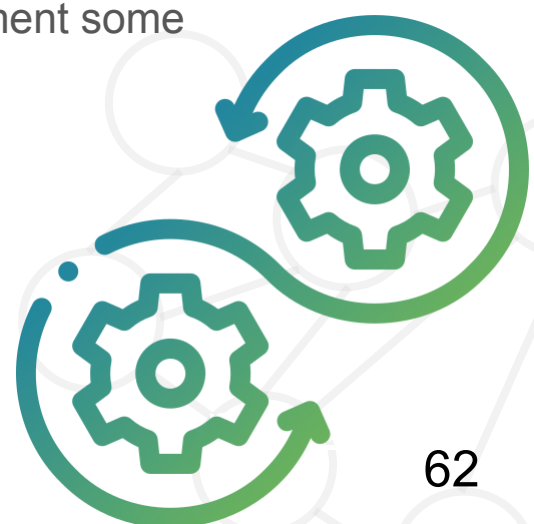
Posterior

Assessing QoE in Edge-Delivered Holographic Streaming with a Programmable Hardware Testbed. In IEEE **NetSoft'25 demos.**

Future Directions

Work-in-progress

- **Improve performance:**
  - We are currently limited by the capacity of the Tofino recirculation port (used to emulate latency, jitter and pipeline switching). We are studying the use of loopback ports and jumper cables to reduce the use of recirculation ports.
- **Precise measurements and traditional INT support:**
  - Currently, the P7 egress pipeline is not used for any function. Therefore, we studied the development of traffic monitoring strategies in this block, improving the measurements offered to the user, and also implementing INT monitoring natively.
- **Packet Loss and Jitter models:**
  - Instead of defining a fixed % of packet losses or a fixed jitter variation, we are looking to implement some packet loss and jitter models, as supported in Mininet.
- **P7 for SmartNICs:**
  - We are implementing support for network topology emulation also on SmartNICs (Bluefield 2). Next, we plan to integrate both solutions for a more complete emulation environment.
- **Open RAN integration:**
  - Plans for application testing in Open RAN Brasil islands, other RNP testbeds?
- **And more (incl. community / collaboration, etc):**
  - Distributed AI workloads, Emulation of LEO, satellite / aerial networking conditions



Community & Impact

Beyond supporting research needs, these tools can democratize 6G training/education.

- **Educational Labs:** Students use P7 to build and break networks without expensive HW.
- **Open Ecosystem:** All tools are open-source, fostering community contributions.
- **Standardization:** Allows validation of IETF/3GPP drafts before they mature.
  - Early validation. Stress-testing NFs, AI-native Functions, APIs, etc.
- **Reproducible results**

Key Takeaways

Uncertainty

- We cannot predict 6G protocols, so we must build experimental platforms that can adapt to anything.

6G Research

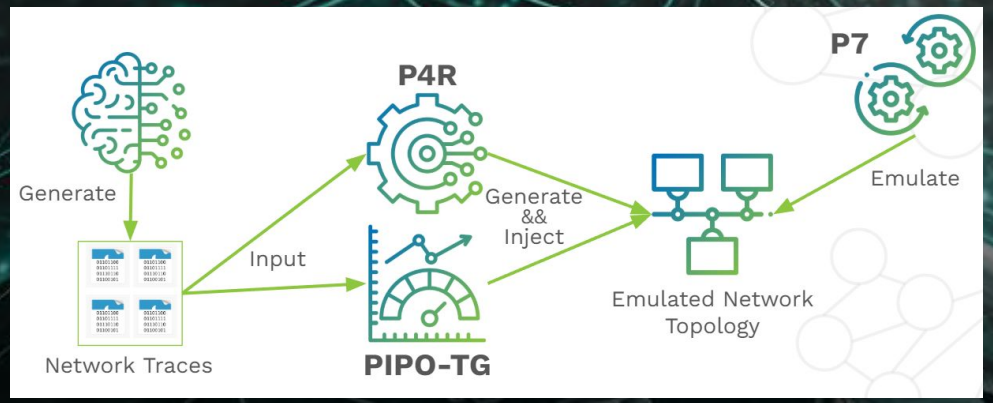
- Needs: openness, flexibility, agility (time-to-test), fine granular time-sensitiviy @ 10s Gbps,

Programmability

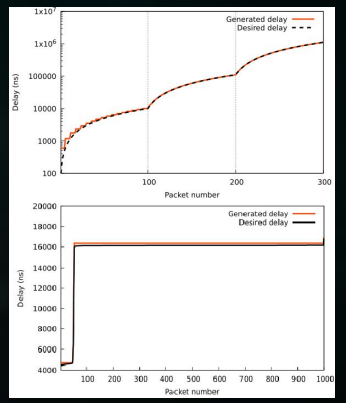
- P4 & Tofino + SmartNICs are key, accessible building blocks for hardware-accelerated, flexible, realistic experimentation.

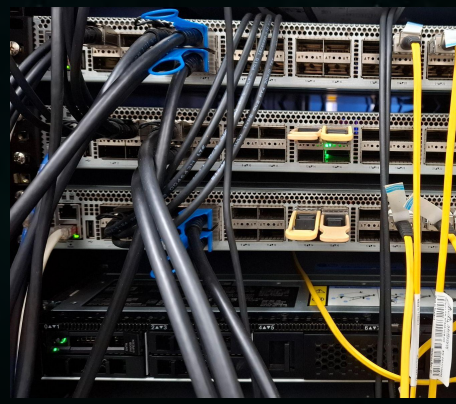
Our Toolbox

- Ongoing open-source efforts (P7, PIPO-TG, P4R) provide the necessary fidelity to validate the "Unknown" of 6G starting with the lesser-known of 5G (e.g. uplink performance) and demanding apps (e.g. volumetric media).



Front





Organized by:

xGMobile · Inatel · FAPEMIG · EMBRAPII · GOVERNO DE MINAS AQUI O TREM PROSPERA. · MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO · GOVERNO DO BRASIL DO LADO DO POVO BRASILEIRO

Thanks for your attention!

GitHub Repositories:

github.com/intrig-unicamp
github.com/smartness2030

Contact:

Christian Rothenberg
chesteve@unicamp.br
https://smartness2030.tech/

Obrigado!

Christian Esteve Rothenberg

SMARTNESS / University of Campinas (UNICAMP)

Organized by:

xGMobile Inatel FAPEMIG EMBRAPII GOVERNO DE MINAS AQUI O TREM PROSPERA. MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO GOVERNO DO BRASIL DO LADO DO POVO BRASILEIRO

Questions?

ACKNOWLEDGMENTS & DISCLAIMER

This work has been performed within the framework of the FAPESP Engineering Research Center (ERC) Program under FAPESP grant agreement #2021/00199-8 (SMARTNESS).

What is SMARTNESS 2030?

CPE: FAPESP Engineering Research Center (ERC)

- Co-Financed by the São Paulo Research Foundation (**FAPESP**).
- FAPESP is a **solid and stable** foundation, with budget of 1% of all state taxes collection (3.5 Billion SEK in 2021).
- ERC is FAPESP's **top program** for collaborative **research with Industries.**
- ERC premise: the execution of internationally competitive research in accordance with **global excellence** benchmarks.
- There are currently more than 15 ERC in different technological areas, e.g., oil and gas, biotechnology, agribusiness, energy, artificial intelligence, etc.
- **SMARTNESS** is the **first ERC** in the **Telecom area**

https://fapesp.br/cpe

SMARTNESS 2030

A networking-centric Engineering Research Center

Mission

Cutting-edge research in communication networks and advanced digital application services.

Towards 6G.

Founders

Ericsson, UNICAMP, USP and UFSCar.

Hub center at UNICAMP.

Long-term investment

10 years.

56 MBRL (~120 MSEK)
1:1:2 – Ericsson: FAPESP: UNICAMP.

50+ associated researchers
15+ university partners
120+ scholarships

History/ Status

2018-20 – Work on the Proposal

Feb/ 2021 – Prop. submission

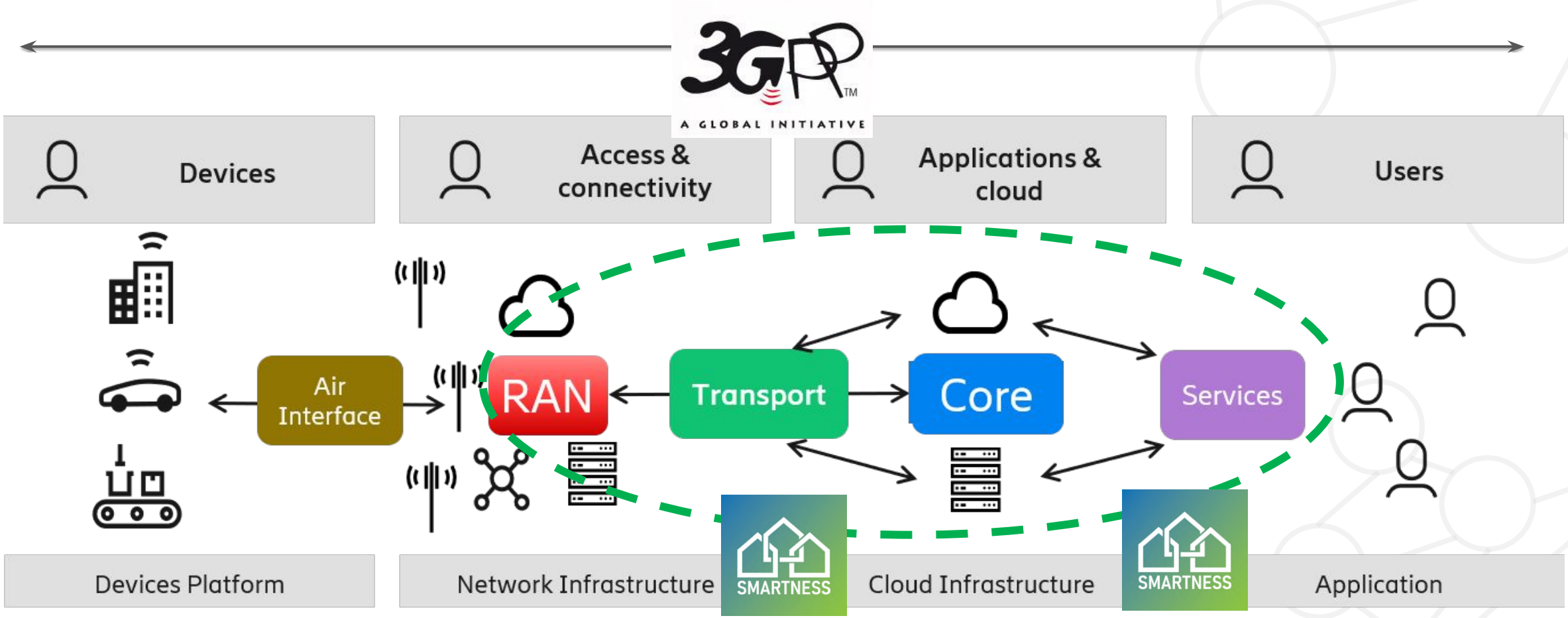
May / 2022 – FAPESP approval

Dec / 2022 – Kick-off ceremony

April 2023 - Official start

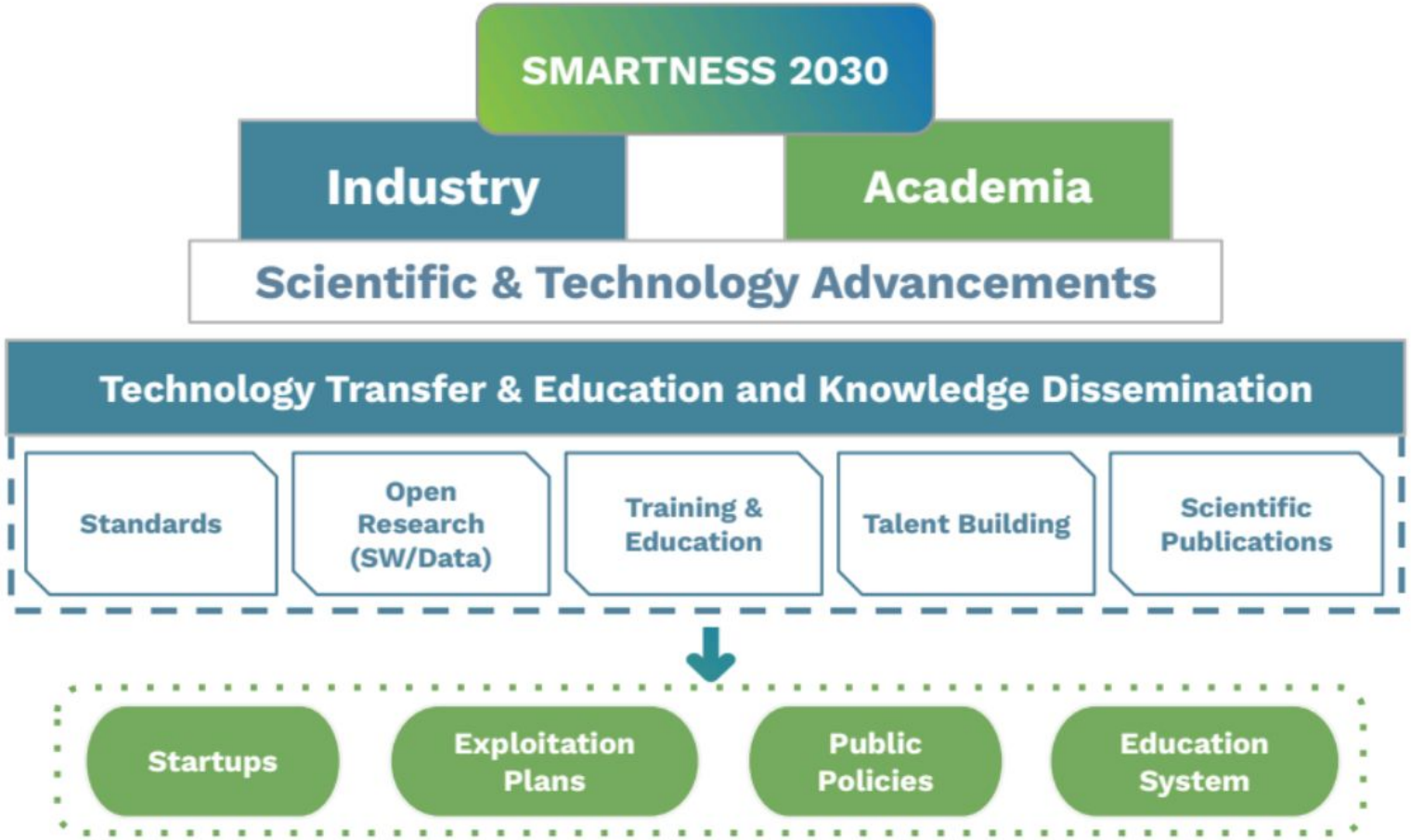
Where is Smartness in the 6G E2E architecture?

Main scope



About SMARTNESS (2023-2033)

Impact areas of the FAPESP Engineering Research Center (ERC)



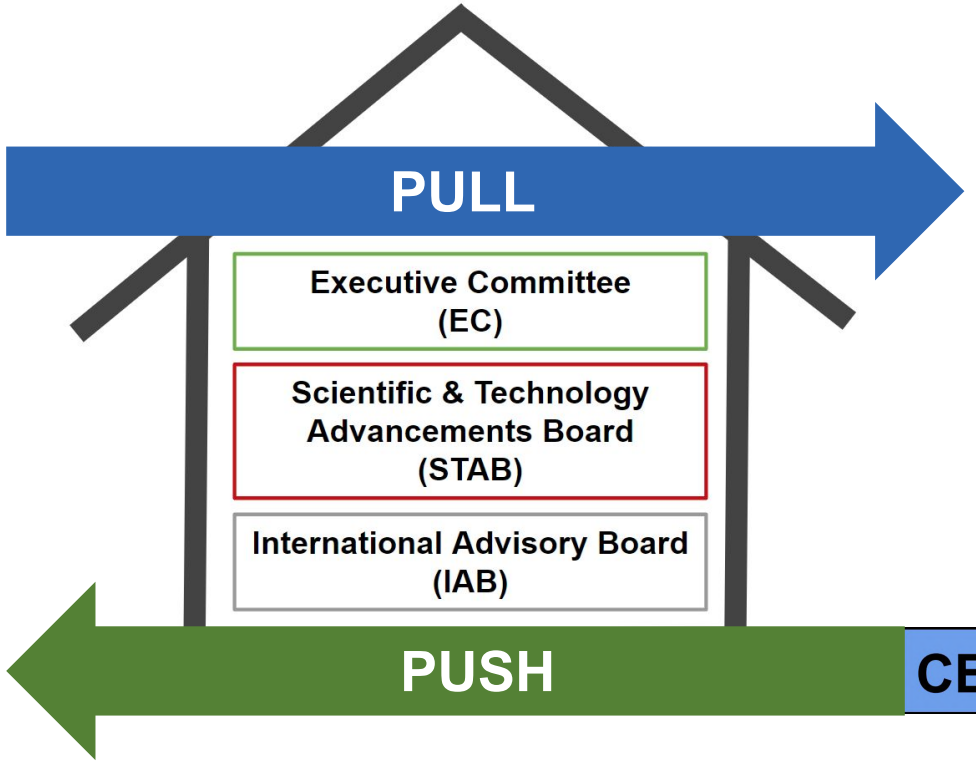
PUSH/PULL modes of operation

"Technology Push & Market Pull" like workflows

- **Academic Research Push:** New research findings, ideas, trends, etc. from SMARTNESS pushed to Ericsson Research
- **Ericsson Research Pull:** New research contribution demands/opportunities from projects / standards brought to SMARTNESS 2030 to shape ongoing Research Strands and/or create new ones.

ERICSSON

- Technology Journeys
- Future Network Programs
- EU SNS JU Projects
- Standardization
- Open Source
- Etc.



SMARTNESS

Beyond State-of-the-Art

Management

Organization, Roles & Responsibilities

International Advisory Board (IAB)

- Industry
- Telecom Operators
- National Ecosystem
- Academia

ERICSSON [Brazil & International]

Business Areas

Ericsson Research

- **Cloud** — Dr. Mattias Wildeman
- **AI/ML** — Dr. Jean Martins
- **Network Control** — Dr. Peter Öhlén
- **Network Data** — Gergely Pongracz
- **Applications** — Dr. Gyanesh Patra

Executive Committee (EC)

- **Deputy Center Director** — Dr. Maria V. Marquezini (Ericsson)
- **Director / PI - Coordinator** — Prof. Christian Rothenberg (UNICAMP)
- **Technology Transfer Deputy (TTD)** — Dr. Mateus Santos (Ericsson)
- **Technology Transfer Coordinator (TTC)** — Prof. Jó Ueyama (USP)
- **Education and Dissemination Deputy (EDP)** — Dr. Klaus Raizer (Ericsson)
- **Education and Dissemination Coordinator (EDC)** — Prof. L. Bittencourt (UNICAMP)
- **Partnership Coordinator (PPC)** — Prof. H. Figueroa (UNICAMP)

Scientific & Technology Advancements Board (STAB)

- **Fluid Control & Data Planes** — Prof. Christian Rothenberg, Prof. Fábio Verdi
- **Cognitive Architectures** — Prof. Ricardo Gudwin
- **Customized Edge Computing** — Prof. Edmundo Madeira, Prof. Luiz Bittencourt
- **Sustainability** — Prof. T. C. M. B. Carvalho, Prof. Daniel Batista
- **Machine Intelligence** — Prof. F. Von Zuben
- **Trustworthiness** — Prof. Marco A. A. Henriques, Prof. Jó Ueyama

Management

- **Executive Manager** — Dr. Bárbara Ilze (UNICAMP)
- **Technology Transfer Manager** — Dr. Eduardo Sartori (UNICAMP)
- **Education and Dissemination of Knowledge Manager** — Dr. Roberta Steganha (UNICAMP)
- **Communication & Marketing**

Scientific & Technology Advancements | Areas

STAs

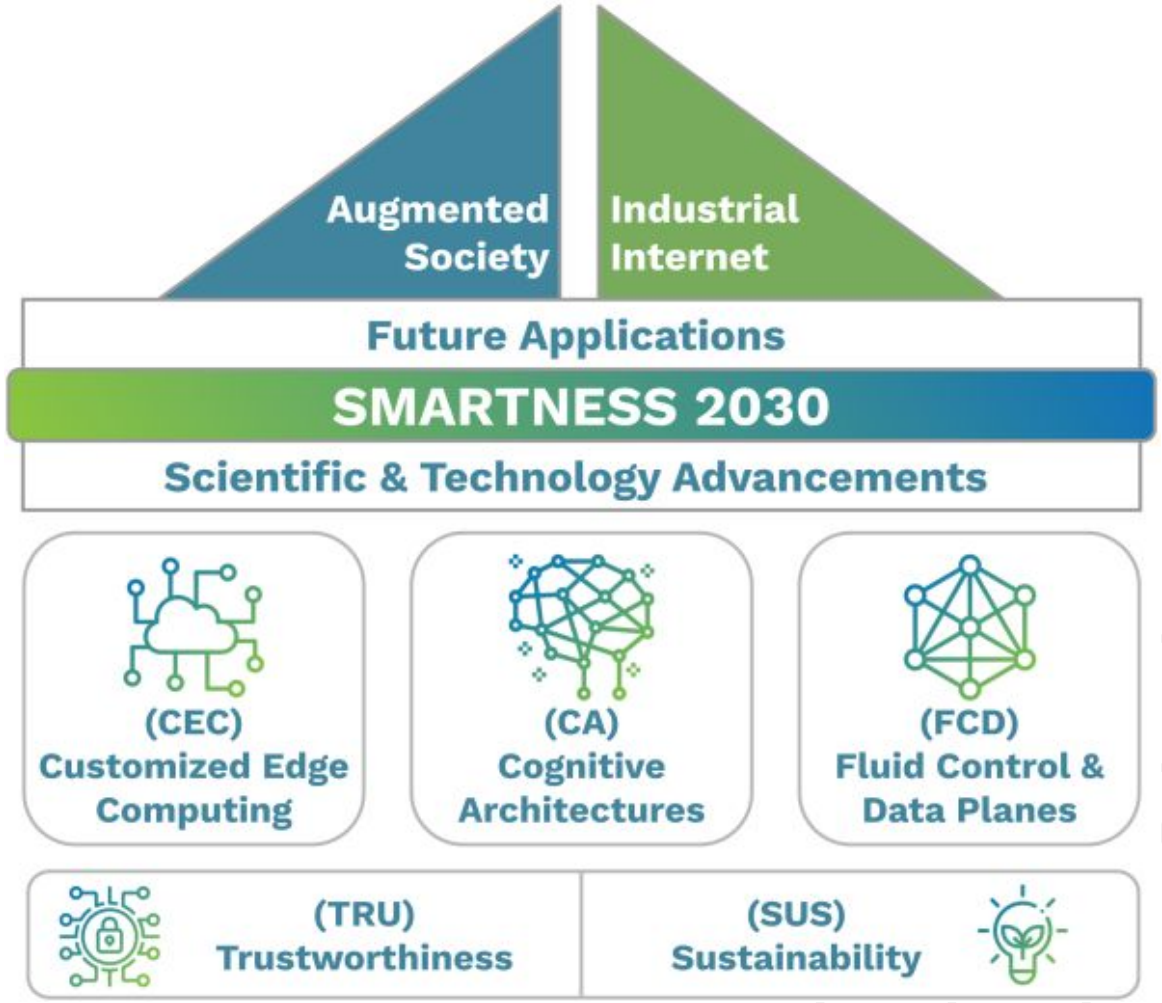
CEC: Customized Edge Computing

CA: Cognitive Architectures & Machine Intelligence

FCD: Fluid Control & Data planes

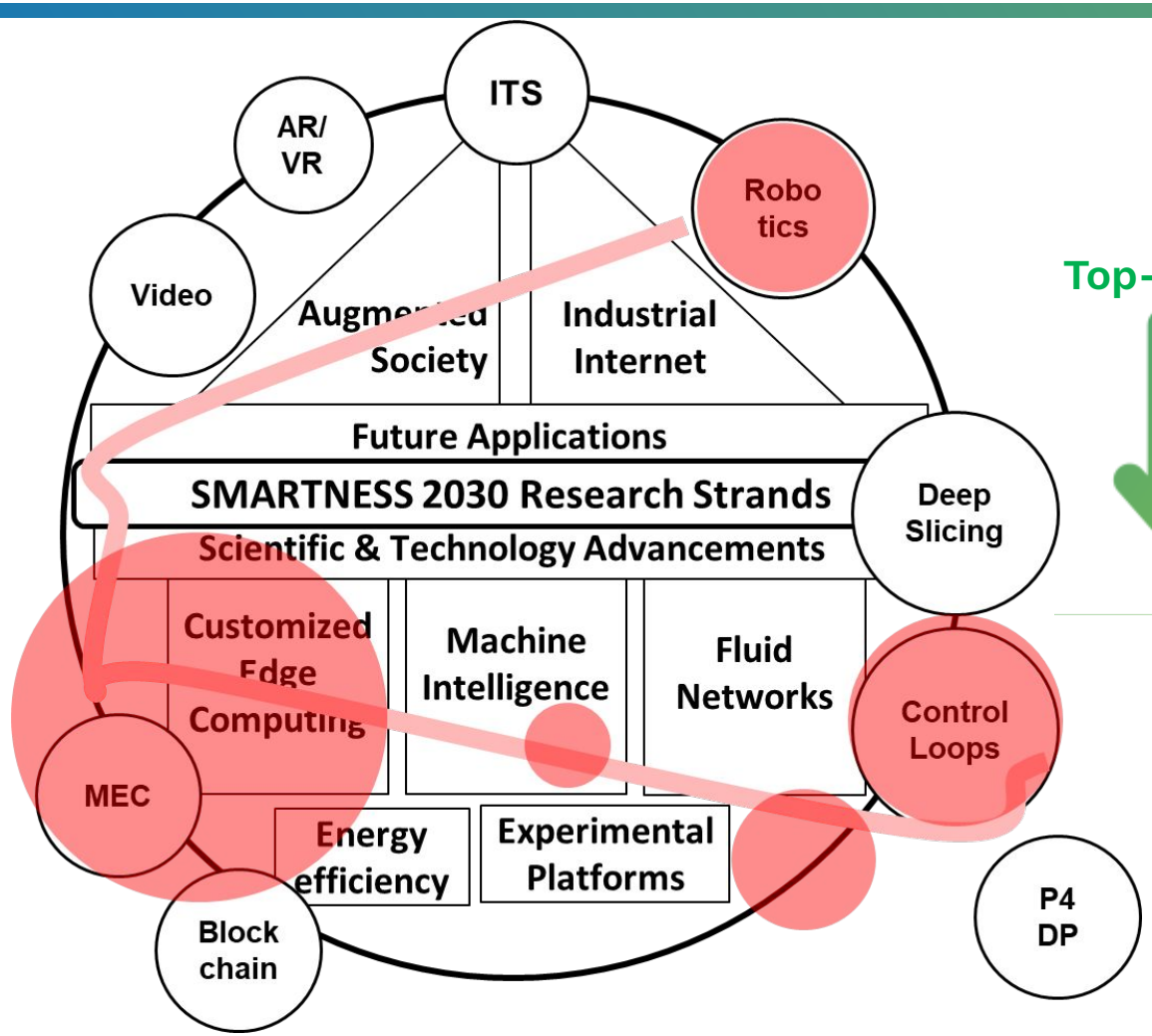
TRU: Trustworthiness

SUS: Sustainability

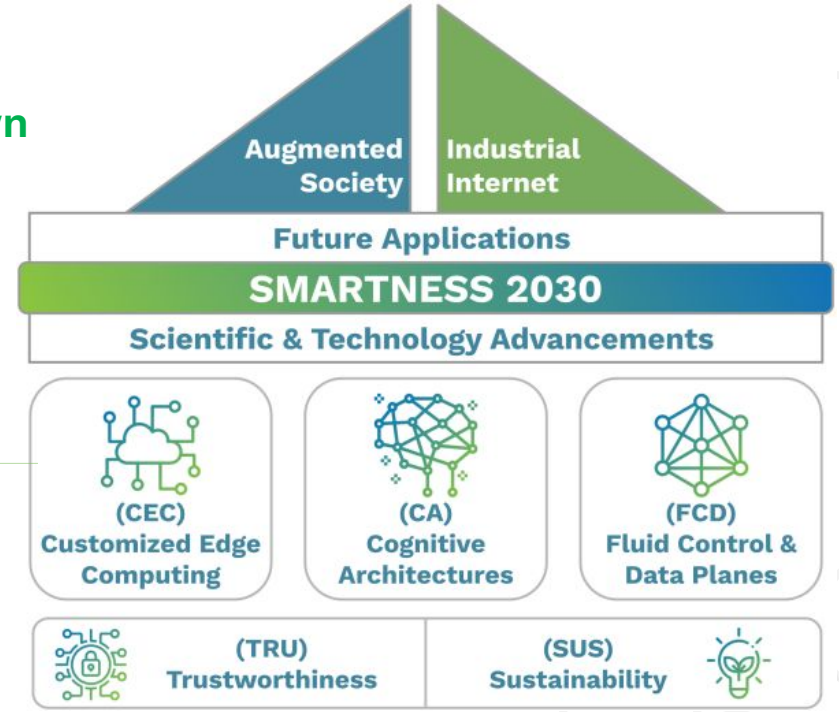


Research Strands (RS)

From Apps/Services to STAs (and vice versa)



Top-down



Bottom-up

Research Strands (RS) - Day 1

RS as organizational thrusts hosting WPs

NEWTON - In Edge Network Industrial automation

- Lead PI: Christian Rothenberg (Unicamp)
- ER Receiver: Gergely Pongracz and Gyanesh P.

C3LA - Cognitive Closed Control Loops Architecture for Edge IoV

- Lead PI: Christian Rothenberg (Unicamp)
- ER Receiver: Gyanesh P. and Szilveszter N

DEMIST - Distributed Edge Computing Swarm Intelligence

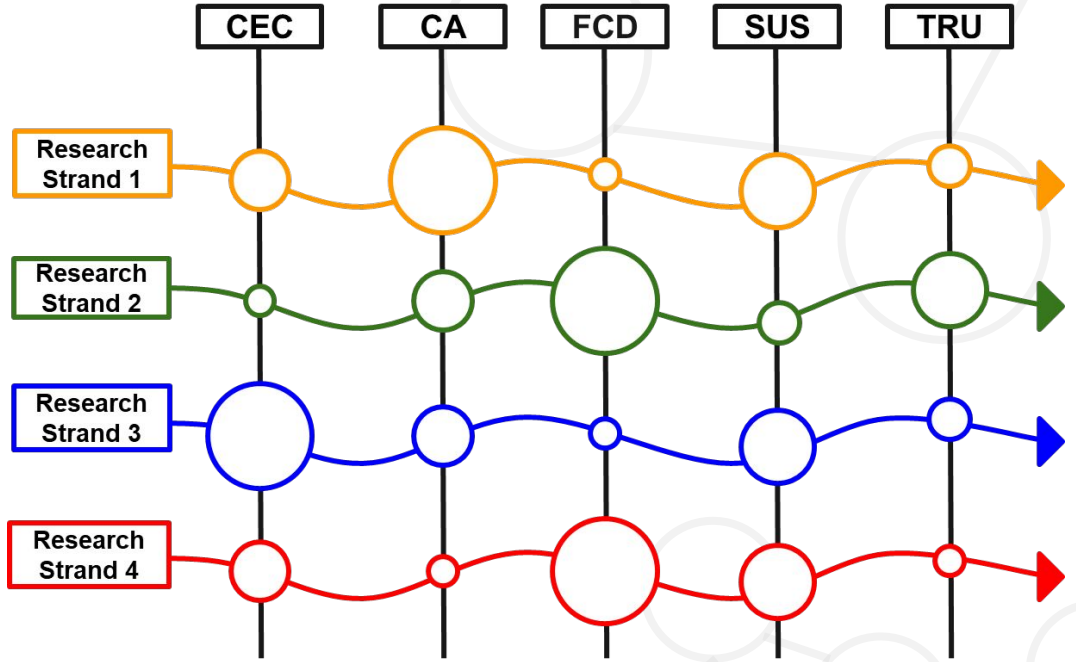
- Lead PI: Luiz Bittencourt (Unicamp)
- ER Receiver: Mattias Wildeman and Björn Skubic

DisCoNet - Distributed Computer Networking

- Lead PI: Fabio Verdi (UFSCar)
- ER Receiver: Andrew Williams and Vinay Yadhav

ADVENTURE - Adaptive and Secure Net. Applications over the Industrial Internet

- Lead PI: Daniel Batista (USP)
- ER Receiver: TBD



Scientific & Technology Advancements

WPs ~18-24 month duration

Research Strands (RS) - Day 2

RS as organizational thrusts hosting WPs