CHALMERS
UNIVERSITY OF TECHNOLOGY

P5: Event-driven Policy Framework for P4-based Traffic Engineering

Citation for the original published paper (version of record):
Famelis, P., Katsikas, G., Katopodis, V. et al (2023). P5: Event-driven Policy Framework for P4-based Traffic Engineering. IEEE International Conference on High Performance Switching and Routing, HPSR, 2023-June. http://dx.doi.org/10.1109/HPSR57248.2023.10148012

(article starts on next page)

P5: Event-driven Policy Framework for P4-based Traffic Engineering

Panagiotis Famelis*, Georgios P. Katsikas*, Vasilios Katopodis*, Carlos Natalino†, Lluis Gifre Renom‡, Ricardo Martinez‡, Ricard Vilalta‡, Dimitrios Klonidis*, Paolo Monti†, Daniel King¶, Adrian Farrel¶
* *UBITECH*, Athens, Greece {pfamelis, gkatsikas, vkatopodis, dklonidis}@ubitech.eu
† *Chalmers University of Technology*, Gothenburg, Sweden {carlos.natalino, mpaolo}@chalmers.se
‡ *Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA)*, Castelldefels (Barcelona), Spain {lluis.gifre, ricardo.martinez, ricard.vilalta}@cttc.es
¶ *Old Dog Consulting*, Llangollen, United Kingdom {daniel, adrian}@olddog.co.uk

***Abstract*—We present P5; an event-driven policy framework that allows network operators to realize end-to-end policies on top of P4-based data planes in an intuitive and effective manner. We demonstrate how P5 adheres to a service-level agreement (SLA) by applying P4-based traffic engineering with latency constraints.**

***Index Terms*—P5, SDN, P4, SLA, traffic engineering.**

I. Introduction

Software-Defined Networking (SDN) is poised to greatly simplify network management through network-wide visibility and direct control over the underlying switches from a logically-centralized control plane. Since 2014, P4 stands on the cutting edge of SDN as the primary language for data plane programming [1], catalyzing innovation in numerous areas of networking, such as packet scheduling [2], traffic engineering [3], network telemetry [4], security [5], etc.

Open Networking Foundation (ONF) is driving the evolution of programmable networking through state of the art P4 controllers, such as ONOS [6], and broader P4 programming frameworks, such as SD-Fabric [7]. These platforms are inline with the P4 ecosystem and community, offering elegant abstractions that bundle together (i) a schema of the P4 pipeline automatically derived from the P4 compiler, (ii) target-specific binaries to deploy a pipeline onto a P4 device, and (iii) driver behaviours that allow to map pipeline-agnostic flow objectives to pipeline-specific rules.

The problem. Despite the huge popularity and constant evolution of the P4 language as well as the community efforts to advance the P4 control plane, expressing intuitive end-to-end network policies atop P4 data planes is still extremely hard. Such policies express how the network's forwarding behavior should change in response to changing conditions. Realizing these policies in a P4-based network requires not only domain-level expertise, but also a systematic ability to (i) collect network state across multiple switches, (ii) reason about this state in an end-to-end fashion, and (iii) compile a correct and compelling traffic engineering strategy that will adapt the data plane according to the policy requirements.

This work is partially funded by the EC through the 5GPPP TeraFlow project with grant agreement number 101015857 and the HORIZON-JU-SNS-2022 ACROSS project with grant agreement number 101097122.

Our work. We propose P5; an event-driven policy framework for P4-based traffic engineering. Our solution introduces two important abstractions atop P4 devices and their drivers. First, a service layer that oversees and controls multiple P4 devices allowing network operators to establish end-to-end connectivity between any two endpoints in the network in the form of intents. Secondly, we introduce a policy framework based on the event-condition-action policy model [8], which intuitively allows network operators to associate an end-to-end service with a service-level agreement (SLA) when a certain (set of) condition(s) is met. We implemented P5 as part of the ETSI TeraFlowSDN (TFS) open-source controller [9]–[11], a novel disaggregated SDN control plane comprised of stateless and stateful Kubernetes micro-services that interact with each other to fulfill network management tasks.

Our demo. We demonstrate P5 on a Mininet-based topology of bmv2 [12] software P4 switches and hosts, where P5 implements a policy to ensure that the end-to-end latency of a service is strictly bounded. We introduce artificial delay to several links between switches to evaluate the efficiency of our solution to (i) detect SLA violations, (ii) trigger path re-computation, and (iii) establish new service paths as a result of dynamic P4 flow rule modifications across multiple switches. In stark contrast to existing solutions, we highlight the simplicity of P5 in expressing end-to-end network policies, which make it appealing for developers and researchers to perform quick prototyping in programmable data planes. We encourage the open-source community to embrace ETSI TFS and further contribute to its microservices in order to support additional use cases.

II. P5 Design

P5 is designed as an ensemble of seven microservices, as shown in Figure 1. Network administrators interact with P5 through a web-based user interface (UI), which allows to (i) declare the devices and links (i.e., topology) to manage, (ii) create overlay services within their network(s), and (iii) associate specific network policies to these services to satisfy customer requirements. All this information is persisted into a logically-centralized, yet physically distributed and scalable, database provided by the *Context* microservice.

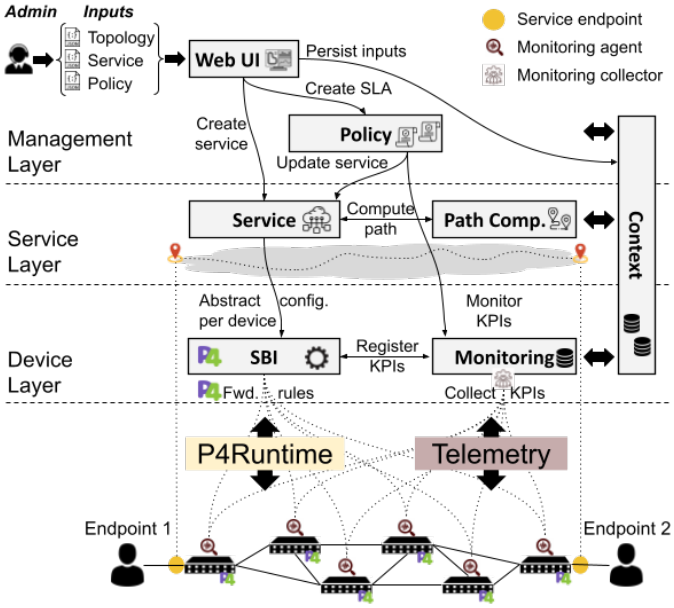

Fig. 1. P5 design based on the ETSI TeraFlowSDN open-source controller.

Concept. P5 spans across three layers as shown in Figure 1. The device layer leverages a South-Bound Interface (SBI) microservice to interact with the underlying network devices through the P4Runtime API and a *Monitoring* microservice for collecting network telemetry. Managing a network at the level of individual devices requires network administrators to master highly-complex and technology specific device configurations. For this reason, P5 introduces additional microservices atop the device layer as shown in Figure 1. This tiered segregation offers two powerful abstractions:

- A service layer that allows network administrators to describe end-to-end connectivity between any two endpoints through high-level intents.
- A management layer that allows network administrators to associate end-to-end connectivity services with run-time network policies.

Abstraction #1: Abstract end-to-end services translated into P4 configuration. Upon a service request by a network administrator (see Figure 1), the Service microservice receives a request to provision connectivity between two remote endpoints from the Web UI. First, the Service microservice requests a path between these endpoints - highlighted in yellow in Figure 1 - from the Path Computation microservice. Then, the Service microservice configures the underlying network devices along the path between the endpoints through the SBI. To keep the Service layer agnostic from technology-specific details, Service leverages a minimal service definition (see Listing 1) that allows users to express *what* they want to connect, while letting the underlying system decide *how* to realize the connection. The Service microservice translates this minimal service definition into an abstract device configuration model that is in turn automatically translated into P4 rules from the P4 device driver of the SBI microservice.

```
{
  "service_type":"P4",
  "service_endpoint_ids": [
    {// endpoint A
      "device_uuid": "SWA",// device ID
      "endpoint_uuid": "X" // port number
    },
    {// endpoint B
      "device_uuid": "SWB",// device ID
      "endpoint_uuid": "Y" // port number
    }
  ]
}
```

Listing 1: Example end-to-end service definition in P5, for P4-based connectivity between two endpoints.

Abstraction #2: Real-time policies atop end-to-end services. Managing the run-time of an end-to-end service is of paramount importance for network administrators as modern systems become more and more complex. P5 offers another powerful abstraction, which allows network administrators to associate monitoring metrics stored in the Monitoring database with conditions according to the event-condition-action policy model [8]. When these conditions are met, the Monitoring microservice raises an alarm that is consumed by the Policy microservice to trigger specific service or device-level actions. The example policy in Listing 2 invokes path re-computation for a given service as an action to bound the end-to-end latency of a service below 4 ms. Additional actions can be requested for other use cases, such as adding specific service constraints or configuration, in which case the network administrator shall specify an action configuration This is how the Policy microservice adds another "P" on top of P4 (i.e., P5), offering event-driven SLAs for end-to-end connectivity services through P4 pipelines.

```
{
  "service_uuid": "d5261206-1047-00345",
  "policy_rule": {
    "priority": 0,
    "condition_list": [
      {
        "kpi_id": "E2E_LATENCY",
        "operator": "GREATER_THAN",
        "kpi_value": 4000 // in us
      }
    ],
    "action_list": [
      {
        "action": "RECOMPUTE_SVC_PATH",
        "action_config":[]
      }
    ]
  }
}
```

Listing 2: Example P5 network policy for bounding the end-to-end latency of a service below 4ms. The action simply triggers path re-computation, without additional configuration needed.

III. Demo Storyline

Demo setup. To verify P5, we use a Mininet-based topology of P4 switches based on the bmv2 [12] software switch as shown in Figure 2. On top of this topology, we deploy P5 as part of the open-source ETSI TFS controller [9] using the seven microservices depicted in Figure 1. The network administrator inputs a list of devices and links in JSON format, which P5 parses and establishes connections with all eight (8) P4 switches through the SBI component. Once device handshaking is completed, the topology is stored into the Context microservice, thus the network administrator can proceed with service instantiation as per Listing 1. In this demonstration example, the service endpoints are "SW1-port4" and "SW8-port4", thus we formulate the JSON accordingly.

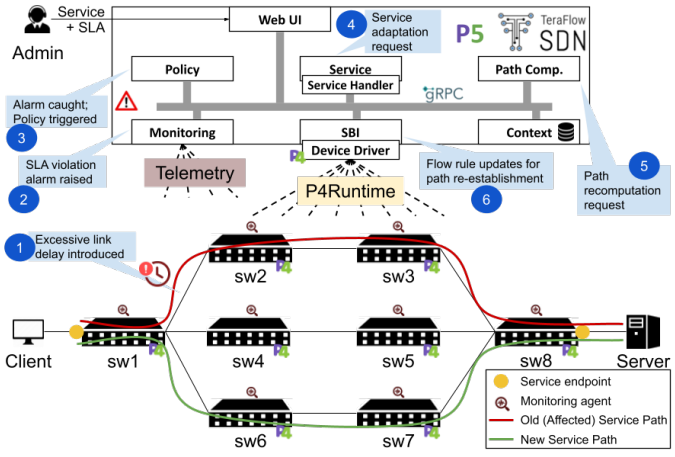

Fig. 2. P5 demonstration scenario.

Workflow. We demonstrate a policy-based service restoration workflow that is comprised of various steps, highlighted in blue in Figure 2. An input policy - similar to the example shown in Listing 2 - bounds the end-to-end latency of the deployed service between client and server below a certain threshold. A probe is deployed with the purpose of monitoring the end-to-end latency and reporting it to the Monitoring microservice. Initially, we assume that the service is established according to the red path shown in Figure 2. To validate the policy, we explicitly introduce excessive link latency (using Linux traffic control) along the service path as shown by step 1 in Figure 2. The P5 Monitoring microservice captures this state change in step 2 and raises an alarm for potential policy violation, which is caught by the Policy microservice in step 3. This event causes the execution of a policy action, which jointly involves a service update (step 4) through path re-computation (step 5). When a new service path is returned by the Path Computation microservice (e.g., the green path in Figure 2), Service compiles a list of device configuration commands for establishing the new path followed by another list of commands for decommissioning the old path. These commands are translated into actual P4 flow rules by the SBI, before being enforced to the data plane via the P4Runtime API (step 6). In the scenario in Figure 2, P5 can pick any path between client and server, thus the red and green paths in the figure are illustrative.

Audience interaction. During the live demonstration, attendees will be able to select which link to introduce latency to and how much latency to introduce, and we will be able to observe the behavior of P5 through the steps just described. The P5 Web UI will be visualizing the available devices & links, the deployed service and its parameters, as well as the real-time state of the provisioned policy.

IV. Conclusion and Future Work

We introduce P5; a modern SDN framework for managing P4 data planes using two powerful abstractions atop low-level P4 device configuration: (i) an end-to-end service layer that allows network administrators to establish intent-based end-to-end connectivity between endpoints and (ii) an overlay management layer that facilitates policy definition, allowing network administrators to associate deployed services with conditions on system-level KPIs and corresponding remedy actions in an intuitive fashion. We demonstrate an end-to-end P5 service with an adaptive latency-based policy as part of the open-source ETSI TFS controller atop software-based P4 topologies in Mininet.

Future work. Towards the next release of the ETSI TFS controller [10], we consider two orthogonal development activities as future work for P5. First, the support for in-band network telemetry [4] for capturing the network state in a P4-native manner. Secondly the evaluation of P5 in inter-domain scenarios with heterogeneous device types and larger topologies, focusing on scalability and performance aspects.

References

[1] P. Bosshart *et al.*, "P4: Programming Protocol-independent Packet Processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, Jul. 2014.

[2] A. Sivaraman *et al.*, "Programmable packet scheduling at line rate," in *Proceedings of the 2016 ACM SIGCOMM Conference*, ser. SIGCOMM '16, New York, NY, USA, 2016, p. 44–57.

[3] T. Holterbach *et al.*, "Blink: Fast Connectivity Recovery Entirely in the Data Plane," in *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, Boston, MA, Feb. 2019, pp. 161–176.

[4] The P4.org Applications Working Group., "In-band Network Telemetry (INT) Dataplane Specification Version 2.1," Nov. 2020.

[5] A. G. Alcoz *et al.*, "Aggregate-Based Congestion Control for Pulse-Wave DDoS Defense," in *ACM SIGCOMM 2022 Conference*, ser. SIGCOMM '22, New York, NY, USA, 2022, p. 693–706.

[6] Open Networking Foundation (ONF), "Open Network Operating System (ONOS) SDN controller," Mar. 2023.

[7] ——, "SD-Fabric™," Mar. 2023.

[8] M. Boucadair, Q. Wu, Z. Wang, D. King, and C. Xie, "Framework for Use of ECA (Event Condition Action) in Network Self Management," IETF, Internet-Draft, Nov. 2019, work in Progress.

[9] "ETSI Open Source Group for TeraFlowSDN," Feb. 2023.

[10] "ETSI TeraFlowSDN GitLab," Mar. 2023.

[11] R. Vilalta *et al.*, "Teraflow: Secured Autonomic Traffic Management for a Tera of SDN Flows," in *2021 Joint European Conf. on Netw. and Commun. & 6G Summit (EuCNC)*. IEEE, 2021, pp. 377–382.

[12] P4 Language, "Behavioural model (bmv2) reference P4 software switch," Mar. 2023.