Offloading Robotic and UAV applications to the network using programmable data planes

Fabricio E Rodriguez Cesen
(PhD Candidate)
University of Campinas (UNICAMP)

Christian Esteve Rothenberg
(Supervisor)
University of Campinas (UNICAMP)

***Abstract*—Next-generation 5G networks are rapidly expanding to support the growing demand for efficient connectivity in Internet of Things (IoT) and Machine-to-Machine (M2M) applications across various sectors (e.g., agriculture, automotive, healthcare, smart cities, and manufacturing). Industrial Internet of Things (IIoT) promises to transform manufacturing through Digital Twins while Industry 4.0 advances digitalization with Cyber-Physical Systems (CPS), machine learning, big data, and cloud computing. Hence, achieving Ultra-low latency (ULL) is crucial for applications like robotic control. Although 5G powered by Software-Defined Networking (SDN) and Network Function Virtualization (NFV) have improved the network capacity and reduced the ULL constraints, challenges persist due to wireless signal unpredictability. To address these issues, this research proposes leveraging in-network applications to the network edge to implement ULL solutions for industrial and Unmanned aerial vehicles (UAVs) applications. Furthermore, we propose the hardware-based P7 emulation environment to evaluate data plane applications' performance, feasibility, effectiveness, and impact.**

I. Introduction & Motivation

With next-generation 5G networks rapidly growing, new opportunities and challenges are emerging. Applications in different trends (e.g. industry 4.0, IoT) are gaining new envisions.

With the IoT and M2M, networks need to support connected a diverse range of devices. It is expected to have more than $50 billion of devices connected during the next years [1]. Having high amounts of transmitted data and various network services, the network's capabilities need to grow together [2].

In parallel, industry 4.0 enhances the digitalization of manufacturing through CPS (humans/robots working together), sharing and analyzing information. This is supported by ML, big data, and cloud computing across the life cycle.

Nowadays, industrial robots are more intelligent than their predecessors. Advanced control and ML techniques have enabled robots to be faster and more accurate than human workers.Not reacting in time (e.g., emergency stop) can entail a severe impact, including collision or, human injuries.

ULL applications with extremely low loss and delay variation are required in new-generation networks [3]. However, horizontal (e.g., distance, nodes processing) and vertical (e.g., NIC, OS, Hypervisor, Application) delays affect the performance of end-to-end communication.

This work was supported by the Innovation Center, Ericsson S.A., and by the Sao Paulo Research Foundation (FAPESP), grant 2021/00199-8, CPE SMARTNESS. This study was partially funded by CAPES, Brazil - Finance Code 001.

The evolution of SDN and NFV has already revolutionized many industries. 5G improved network capacity and ULL constraints. However, wireless signals can be undeterministic, disabling 5G alone to be feasible for latency-critical applications such as robotic control. With the benefits of SDN, it is possible to enable some tasks to be resolved in the data plane (inherently, closer to the requester). Without practically reaching the controller, we can significantly reduce latency and reaction time.

Traditional network environments can support a wide set of applications (e.g., vehicles, UAVs, robots, sensors), and the remote control is usually made from Mobile Edge Computing (MEC) or an external cloud. Our initial insights suggest that moving the control logic to the network edge is essential to reduce processing times. Also, these reduce the amount of traffic in the aggregation or core of the networks. We can achieve an edge solution using programmable data planes. Hence, it is possible to potentiate industrial networks and achieve low latency communication.With Programming Protocol-Independent Packet Processors (P4) [4] gaining envision in data plane solutions, we can use P4-capable switchesto implement an edge solution as a control strategy.

Network emulation is a valuable tool for testing and evaluating network configurations and protocols in a safe and controlled environment. With an ever-increasing demand for complex network environments being developed by both the industry and academia, software-based environments struggle to deliver high-fidelity experiments for real scenario instances. Advances in network programmability, such as P4, have primarily fueled this surge in demand. Along the same lines, the hardware-based emulation of networks [5], [6] represents a potential approach to leverage high-fidelity, high-performance testing capabilities while offering flexibility and customizability of software-based solutions.

Considering the above challenges and issues, this paper aims to investigate the use of P4 switches to implement edge solutions as a control strategy for ULL applications in industrial networks and with UAVs. The research questions to be addressed are related to the feasibility and effectiveness of using P4-capable switches for edge solutions, the impact on network performance, and the scalability and practicality of the proposed solution. In parallel, we aim to validate the P4-based in-network solution in a hardware-based environment to achieve high performance, low latency, and a realistic scenario.

II. Related work & Open Challenges

In this section, we briefly present the closest related work on in-network applications in programmable data plane.

A. *Industrial Robot Control*

Recent literature research aims to provide in-network action developments. Some authors have driven the development of offloading cloud actions to local networks. In [7], authors offload from the cloud to a local network box, tasks of control, and communication deploying in P4. The work in [8] implements a sophisticated security logic on the data plane.

In [9], authors bring computer vision tasks to the network by deploying a P4-based system capable of identifying patterns in images and performing actions accordingly using convolution filters. Similar in [10], the authors implement a complex event processing to process streams of basic events. The authors in [11], evaluated an environment with robot arms and human collaboration using a VR/AR application. They implemented a solution using cloud-native programs and CPU core allocation, demonstrating the impact of delays during deployment.

A similar approach is found in the MEC architectures that bring cloud-computing capabilities to the network edge [12]. In [13], an approach based on MEC is proposed, where authors aim that if robot actions are programmed in the P4 switch, the processing of sending information to the MEC can be reduced, and a direct response from the edge device can be obtained.

In [14], parts of the industrial controller logic were outsourced to the data plane. The results obtained show a significant reduction in the sensor's delay. A similar approach is explored in [15], where authors investigate performance trade-offs between different in-network platforms and aboard different opportunities to add critical tasks from the industrial scenario. The insights were validated with an in-network coordinate transformation task.

From the wide set of emerged applications, industrial, more specifically robots (e.g., UAVs, robot arms, digital twins), has become a trend in innovation and development of new solutions. UAVs, especially Quadcopters, are becoming an important application powered by the new network characteristics [16]. However, a significant challenge in UAV navigation is to ensure safe operation, avoiding obstacles or other devices. Bringing the programmable data plane benefits and the flexibility of P4, implementing a Collision avoidance algorithm in the edge device can fill the gap for low latency applications over the network.

With Industry 4.0 and new-generation networks such as 5G, new UAV applications are attracting growing interest. Driven by the rising demand in commercial and civil applications, UAVs are primarily designed for navigating from a starting position to a destination way-point. However, a significant challenge in navigation is to ensure safe operation without collisions with obstacles or other devices.

Different strategies have been developed and proposed for collision avoidance. Therefore, the approaches developed for collision avoidance are based on defining a signal to determine a trajectory free of obstacles.

Existing works related to collision avoidance control algorithms fit into six categories [17] (i.e., Path planning, Conflict resolution, Model predictive control, Potential field, Geometric guidance, and Motion planning of teams of multi-UAVs).

Collision cone and Prioritized planning require less computational power and are simpler to implement. Considering the principal characteristics of the algorithms, such as using velocity parameters, supporting static & moving objects, being designed for 3D environments, being suitable for large teams, algorithm focus on distributed or centralized scenarios, the threshold used to detect a collision, it is possible to define the complexity to simplify the algorithm.

With this classification of complexity, we can identify the algorithms that are suitable for In-network implementation. A collision cone defines a secure area and creates a safe path to avoid collisions based on the position and velocity of the devices. In Prioritized planning, each UAV is given priority based on its task, and a centralized algorithm can find a solution for each UAV to avoid collisions.

B. *Network Emulation*

Alternatives to deploy and validate P4-based solutions in a realistic network scenario are frequently limited to small-scale environments (e.g., speeds, number of devices, complexity), emulation/virtualization environments or simulation-based approaches. These challenges compromise different aspects, such as realism, flexibility, scalability, and customizability of the experiments, among others.

Mininet [18] has core limitations in providing high-fidelity network experimentation since it is limited by the characteristics of the host.

Koponen et al. [19] proposes NSX, a network virtualization platform developed by VMware to manage multi-tenant domains in data center environments. Similarly, CrystalNet [20] consists of three main components to emulate large-scale production networks: (i) the Topology Generator leverages statistical models and real-world network data; (ii) the Traffic Modeler leverages ML techniques; (iii) the Network Emulator leverages VM for the testing of different network configurations and policies. However, the work is limited to virtual switches and routers.

More recently, BNV [21] leverages hardware virtualization technologies (e.g., Intel VT-d, SR-IOV) to enable the virtualization of network devices (e.g., Open vSwitches). SimBricks [22] is designed to enable end-to-end simulation of host networking stacks. It simulates components of the host networking stack, such as the NIC, driver, kernel, and application, to provide a more comprehensive, flexible, and cost-effective approach.

TurboNet [23] is the first to leverage the programmability of modern switches to emulate network behavior accurately. Specifically, it uses P4 to implement switch behavior in hardware. Nevertheless, not all link characteristics and the support of a custom P4 code are present.

C. *Open Challenges*

With the characteristics of new-generation networks, the rapid increase of network applications and connected devices, and the fact that resources can be limited, the demand over the network is reaching a critical point. As if it were not enough, ULL applications demand rigorous standards to achieve the proposed objectives, being critical in some scenarios (e.g., detecting objects and defining actions). These challenges directly affect the evolution and the addition of new applications.

The advances in SDN, most specifically, in programmable data planes, in turn, can enable the possibility of in-network applications. In-network solutions have emerged with the advent of programmable and stateful networks. These solutions allow certain control functions to be offloaded to the network itself, enabling tasks to be executed entirely in the data plane, which is much closer to or at the network edge. This approach significantly reduces latency and reaction time, as it eliminates the need to traverse the centralized controller, which may encounter congestion and delays in the core of the network.

Network programmability has opened up many applications and opportunities, and various approaches have been explored to use in-network applications to enhance traditional cloud scenarios. In this context, several related works focusing mainly on P4 in-network solutions have been investigated.

1) In-network Applications: Considering the success of P4 in network applications and given the research scope, there are several potential areas to explore P4 in non-networking scenarios. Since P4 unlocks multiple capabilities, for instance, (i) to enable network programmability; (ii) packets processed in hardware, enabling the creation of custom packet and pipeline distribution; (iii) to define own packet formats, parser, and processing logic; (iv) customize the behavior of network devices and implement new features; (v) offers a high throughput and low latency, for a high-speed network environment. However, applying P4 to a non-networking scenario requires innovative development and design choices for P4. If we perform an in-network action directly with P4, how do we adapt a control algorithm in a programmable device to perform an in-network action directly with P4? What if the communication is not synchronized between end-to-end devices? How can we process and respond to messages, apply algorithms, and optimize resources from a network device? Finally, how can we integrate in-network solutions to control other devices, such as robots and UAVs? These questions remain open, presenting research opportunities to extend the benefits of network programmability to non-networking or non-traditional scenarios.

2) Collision avoidance algorithms: A collision avoidance algorithm is necessary to prevent collisions between multiple UAVs or objects (e.g., buildings) in the airspace. A UAVs can collide without a collision avoidance algorithm, causing damage or even accidents. Therefore, developing effective UAVs collision avoidance algorithms is critical for safe and reliable operations. In a programmable network, offloading the computational and communication burden to the network infrastructure is possible. Furthermore, deploying the algorithm inside the network makes it possible to integrate it with other network functions, such as traffic management, congestion control, and routing. This can enable more coordinated and optimized use of the airspace, improving the overall efficiency and safety of UAVs operations. In addition, implementing the algorithm inside the network can provide a more scalable and flexible solution, as it can be updated and modified centrally without requiring changes to individual UAVs. This can also make deploying and managing the collision avoidance system more accessible, as the network infrastructure can be leveraged. In this spirit, it is essential to have an open-source in-network Collision Avoidance algorithm towards data plane flexibility, extending traditional network functions, and exploring the programmability, performance, and scalability in a new era of network applications.

3) Hardware-based emulation: Traditional network experiment solutions, such as virtual and emulation-based environments, suffer from performance fidelity, trade-offs, and scalability constraints. Therefore, there is a need for a realistic experimental platform that can provide high-fidelity performance, scalability, and flexibility. In this spirit, it is critical to leverage the power of P4-based hardware (Tofino) to provide a realistic experimental platform with high-fidelity performance, scalability, flexibility, and support for data plane programmability using a hardware-based environment. An emulation tool that offers high-fidelity 100G traffic network emulation, including various link characteristics such as latency, jitter, packet loss, bandwidth, and the option to customize network topologies, has the potential to become an important tool for P4-based network emulation.

III. Approaches & Contributions

Our work and contributions consist of a set of approaches aiming at addressing the above-mentioned gaps.

A. *Low Latency Robot Control in Programmable Data Planes*

Recently emerged programmable and stateful networks have given rise to in-network solutions, enabling simple calculations to be offloaded to the network itself. By enabling some tasks (i.e., requests) to be resolved entirely in the data plane (inherently, closer to the requester) without practically reaching the centralized controller, we can significantly reduce latency and reaction time without the need to overcome possible congestion and delays in the core of the network.

In this paper, we take the first steps towards this direction and investigate whether industrial robot controllers can benefit from the revolutionary network paradigm shift by overcoming the possible pitfalls via an in-network solution. In particular, we present the first in-network robotic control application (i.e., offloaded to the programmable data planes) capable of reacting (i.e., responding to the robot) to sudden changes in the robotic cells. To reach this end, we leverage the key capabilities of network programmability, and we design and implement custom functions to parse and analyze the TCP communication between the robot and the controller and

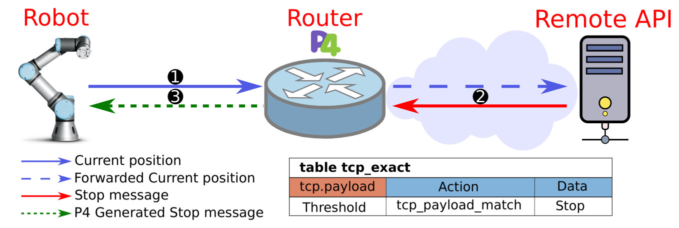


table tcp_exact

tcp.payload	Action	Data
Threshold	tcp_payload_match	Stop

Fig. 1: In-network P4-based implementation.

perform basic calculations (e.g., distance, average filtering) in-network [24]. Particularly, when we detect a position threshold violation in the data plane (by intercepting status messages sent by the robots to the controller), a custom reply packet is crafted at the first hop in the network to deliver an emergency stop command to the robot arm within no time.

The main goal of our in-network robot arm control is to investigate the potential of a programmable data plane (e.g., P4-based device) deployed close to the target robot components (e.g., arm, sensor) in a centralized network with undeterministic connections. Particularly, we explore how to reduce the latency of critical actions (e.g., emergency stop, alarm notification, synchronization) when needed.

Accordingly, the system being investigated encloses a robot and a controller (see Fig. 1), where the robot arm is programmed to do well-structured repetitive tasks. In contrast, the controller role encloses verification, failure response, synchronization process activities, etc. One challenge of in-network solutions arises from messing up the TCP session. If we manipulate or create any message by the P4 device, we need to sync all the ACK and SEQ numbers adequately to keep all TCP parameters valid for both endpoints.

When the robot sends a message with its position (Fig. 1) ➊, and it matches a defined threshold, the P4 router generates an automatic response with a calculated ACK and SEQ numbers to the robot ➌, without any interaction of the controller.

Altogether, our experimental findings suggest that an in-network approach can play a critical role in ULL applications. The proof of concept evaluation using a P4 router to send rapid actions to the arm demonstrates that programmable devices could unlock new ways to offload robot controller actions to the network. We also confirm that with P4, we can effectively match payload information from the robot and use it to craft valid in-network packets acting as fast controller actions.

B. In-network Centralized Collision Avoidance Algorithm

UAVs, are becoming an important application powered by the new network characteristics. Having external sensors (e.g., camera, GPS, proximity sensor), UAVs can observe sudden changes in the operational area, such as unexpected obstacles or even other UAVs. In such critical circumstances, not reacting in time (e.g., adjusting movements) can entail a severe impact, including collision. To resolve this, each robot is directly connected to a powerful Remote API for continuous monitoring and swift and precise interventions.

Distributed and centralized environments have their strengths and weaknesses. In this context, figure 2 presents our approach, a centralized controller enhanced with P4.

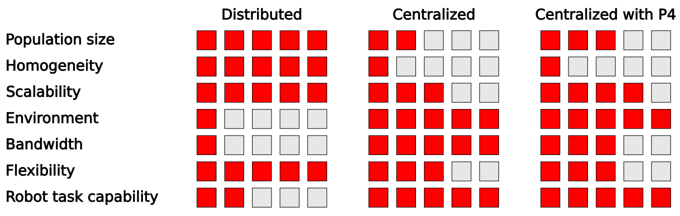


Fig. 2: Comparison of Distributed and Centralized scenarios.

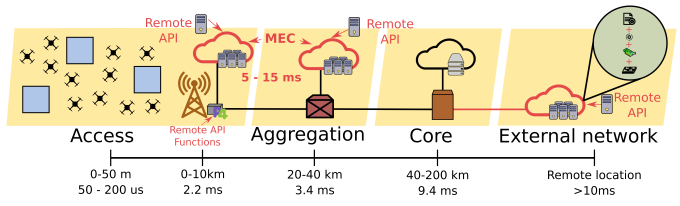


Fig. 3: Quadcopter Scenario.

Nowadays, enterprises increasingly offload their business-critical workloads to the cloud to benefit from low infrastructure costs, high availability, and flexible resource management. However, they have to face the unreliable (i.e., lossy and congested links, latency) nature of today's network. We bring cloud-computing capabilities to the network edge in a MEC architecture. However, we fancy additional processing delays (e.g., switch, NIC, OS, Hypervisor, Application). If some control functions are offloaded to the network edge device (e.g., P4 switch), it is possible to reduce this processing time while directly responding from the network.

We propose an in-network Centralized Collision Avoidance Algorithm in the Data Planes, bringing the benefits of a P4 device. Adding the remote API functions in the edge device, we avoid the latency of the MEC $(5-15ms)$ or even of an external network (>10 ms) [25] [26] Despite the possible limitations (e.g., complex operations, memory), we argue the possibility of implementing an approach based on a Real-Time Three-Dimensional Obstacle Avoidance Using an Octomap (3DVFH+) [27] algorithm. Having a complete view of the environment and adding the capability of avoiding collisions.

If we deploy the in-network solution in a programmable device located at the edge of the network, we can avoid the processing of the remote API located in the MEC and the 3DVFH+ algorithm software processing. The total time of processing is presented in equation 1. As an example, we can assume horizontal delay values from the literature.

$$T_t = \overbrace{0.05ms}^{\text{Access}} + \overbrace{2.2ms}^{\text{Edge}} + \overbrace{\cancel{5ms}}^{\text{MEC}} + \overbrace{\cancel{0.3ms}}^{\text{3DVFH+}} + \mathbf{P4} \tag{1}$$

Considering the benefits of implementing in P4 the collision avoidance algorithm, the proposed solution overview of the logic applied in the P4 code to detect a collision using the 3DVFH+ is presented in Algorithm 1.

In our approach, we use a histogram to maintain the position of the objects in the environment. This information is stored using registers. Considering the 32 bits per register limitation in Tofino Native Architecture (TNA), depending on the size of the histogram, the total number of registers can vary.

Algorithm 1 P4 algorithm flow

```
Require: Parse Packet Headers
    type ← Message Type
    histogram ← Register Possition
    if type is Command Message or Message From API then
        Forward
    else
        Update Histogram Cost
    end if
    if histogram not Obstacle then
        Forward
    else
        Get Directions; Modify Packet; Forward
    end if
```

To simplify the allocation of information, it is possible to divide into groups. Each of these groups will allocate 30 bits of the histogram. To compare the information of the histogram, it is possible to use different approaches. The resource of recirculation or sequential register calls can be employed.

In a histogram, each object can have different weights. Different values represent the presence or absence of an object and the priority of the cost of devices. The representation in the histogram is made by setting a number 1 where an object is located and a 0 where it is empty. This information is also used to detect a possible collision during the avoidance algorithm.

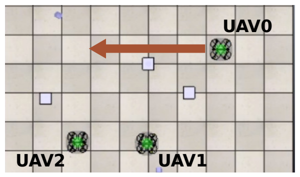
We validate our collision avoidance algorithm, following the testbed presented in Fig. 3. Drones are implemented in CoppeliaSim together with flight mechanisms and hardware representations. The P4 code with the collision avoidance algorithm runs in a Tofino switch. Finally, the remote Application Programming Interface (API) is a Python script connected to CoppeliaSim over TCP. This script controls the drone's destinations, formations, and actions.

To validate the avoidance control performed by the P4 code, we add different link metrics (i.e., latency) to the link between the Tofino switch and the Remote API. Then, we verify that the drones avoid the collision.

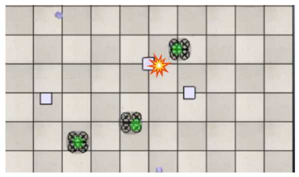
The scenario in Fig. 4 shows UAV collision detection with a static object, where *UAV0* has a destination point at the other side of the grid (see Fig. 4a), and during its path, it has to cross over a static object. When *UAV0* reaches nearby (defined secure zone) the object (see Fig. 4b), the P4 switch calculates and sends to *UAV0* the information of the avoidance path to avoid a collision (Fig. 4c). After the avoidance movement, *UAV0* returns on his way to the destination.

C. P4 Programmable Patch Panel (P7)

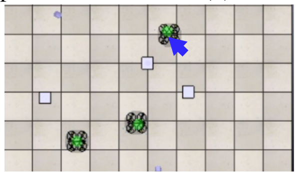
Virtual and software-based environments, such as Mininet, have become popular for network experimentation. However, these platforms often have limitations, including low transmission speeds and trade-offs between scalability and performance fidelity. Advances in P4 programmability and new P4 hardware that supports TNA have enabled the possibility of emulating various network link characteristics and creating network topologies for running line-rate traffic in a single P4 switch (i.e., Tofino). P7 emulator [28]–[30] allows the configuration of network scenarios with different link characteristics, including 100G traffic capacities, using a single P4 switch.



(a) Initial position



(b) Collision detected



(c) Avoidance movement

Fig. 4: Collision avoidance use case UAV and object.

With increasingly powerful and complex networking environments (e.g., robots and UAVs) being worked out by industry and academia research, notably fueled lately by network programmability advances and efforts, 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 broadly impact the quantity and quality of networking devices. In this scenario, preparing and running experiments are frequently limited to small-scale environments (e.g., speeds, number of devices, complexity), emulation/virtualization environments , or simulation-based approaches. In the end, well-known trade-offs of networking experimentation hit the research roadmap and compromised different aspects such as realism, flexibility, scalability, and customizability of the experiments, among others.

P7is 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 with 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) from a traffic or trace generator to the emulated networking scenario.

In this context, it is crucial to have a solution capable of accurately representing and modeling complex network scenarios. We can model the behavior between our applications (e.g., UAVs) and their controller, representing different metrics and variations that can affect end-to-end communication.

With P7, we leverage the possibility of setting a custom pipe distribution model. We propose a solution where a dedicated pipe runs the P7 P4 code, and a separate pipe runs the user-defined P4 code (e.g., collision avoidance).

IV. Final Remarks

In industry 4.0 applications, having more than one device working in synchronization is common, and they must share their information. We believe that apart from looking for use cases with one robot, future research should look for a scenario where two or more robots work nearby in a shared space. If one of the robots is in a collision course with others (including objects or humans detected by collaborative sensors), it is necessary to act rapidly. Programmable network devices may react to such events and send a message of a new course or even a stop action. These activities are required within a short time to prevent possible accidents effectively.

Overall, we presented the design of a P4 implementation of in-network actions to control a robotic arm and a collision avoidance algorithm in edge by offloading some controller functions to a programmable edge network device itself. We described scenarios where a programmable device could make a difference in terms of ultra-low latency response. We showed how to effectively manipulate the content of the messages to craft new replies within an established TCP session. Moreover, we explore the benefits of a P4 device (e.g., High Performance, Reconfigurability, Protocol Independent) in a ULL use case. Our preliminary experimental evaluation demonstrated how the delay affects the connection to an accurate stop position or avoidance message. The kinematic configuration of the robot and drone (i.e., acceleration, step size) also affects the error (stop position difference) to apply an action. Furthermore, we demonstrate that the collision avoidance algorithm in P4 can detect and react fast to avoid possible impacts effectively.

Programmable data planes in delay-critical scenarios open up new opportunities for a different range of applications (e.g., sensor monitoring, data filtering, threshold matching). All in all, our P4-based robotic arm control experience suggests intriguing opportunities that are not limited to robots only but apply to a wide range of applications (e.g., industry 4.0, automation, real-time control, synchronization) where low and deterministic latency is paramount.

In parallel, we show how P7 contributes to the ecosystem of affordable 100G experimental platforms with a user-friendly, cost-effective 100G network emulator in support of traditional networking, advanced programmable networking research, and teaching purposes. Also, supporting scenarios where the variation of metrics is required for the tests. Being a high-fidelity testbed, P7 facilitates repeatable and reproducible research by sharing P7 topology files to be compiled and deployed, resulting in the same output anywhere (along with the specific Tofino target capabilities permit). Furthermore, P7, with its multiple pipelines and line-rate support, will leverage the possibility of validating our in-network solutions by defining different scenarios with custom metrics and behaviors.

Acknowledgment

This work was supported by the Innovation Center, Ericsson S.A., and by the Sao Paulo Research Foundation (FAPESP), grant 2021/00199-8, CPE SMARTNESS. This study was partially funded by CAPES, Brazil - Finance Code 001.

References

[1] J. Bradley *et al.*, "Embracing the Internet of everything to capture your share of $14.4 trillion," *White Paper, Cisco*, vol. 318, 2013.

[2] S. Talwar and R. Vannithamby, *Introduction*. John Wiley & Sons, Ltd, 2016, ch. 1, pp. 1–8.

[3] A. Nasrallah *et al.*, "Ultra-low latency (ULL) networks: A comprehensive survey covering the IEEE TSN standard and related ULL research," *arXiv preprint arXiv:1803.07673*, 2018.

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

[5] A. A. Alli and M. M. Alam, "The fog cloud of things: A survey on concepts, architecture, standards, tools, and applications," *Internet of Things*, vol. 9, p. 100177, 2020.

[6] K. Nisar *et al.*, "A survey on the architecture, application, and security of software defined networking: Challenges and open issues," *Internet of Things*, vol. 12, p. 100289, 2020.

[7] J. Rüth *et al.*, "Towards in-network industrial feedback control," in *NetCompute*. New York, NY, USA: ACM, 2018, pp. 14–19.

[8] A. C. Lapolli *et al.*, "Offloading Real-time DDoS Attack Detection to Programmable Data Planes," in *IFIP*. USA: IEEE, 2019, pp. 19–27.

[9] R. Glebke *et al.*, "Towards executing computer vision functionality on programmable network devices," in *ACM CoNEXT ENCP*, 2019.

[10] T. Kohler *et al.*, "P4CEP: Towards In-Network Complex Event Processing," in *NetCompute*. NY, USA: ACM, 2018, p. 33–38.

[11] B. G. Nagy *et al.*, "Towards Human-Robot Collaboration: An Industry 4.0 VR Platform with Clouds Under the Hood," in *IEEE ICNP*, 2019.

[12] Y. Hu *et al.*, "Mobile edge computing A key technology towards 5G," *ETSI*, vol. 11, pp. 1–16, 2015.

[13] W. Wang, "5G Mobile Platform with P4-enabled Network Slicing and MEC," September 2019.

[14] J. Vestin *et al.*, "FastReact: In-network control and caching for industrial control networks using programmable data planes," in *IEEE ETFA*, vol. 1, 2018, pp. 219–226.

[15] I. Kunze *et al.*, "Investigating the Applicability of In-Network Computing to Industrial Scenarios," in *IEEE ICPS*, 2021, pp. 334–340.

[16] S. K. Khan *et al.*, "The role of unmanned aerial vehicles and mmWave in 5G: Recent advances and challenges," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 7, p. e4241, 2021.

[17] S. Huang *et al.*, "Collision avoidance of multi unmanned aerial vehicles: A review," *Annual Reviews in Control*, vol. 48, pp. 147–164, 2019.

[18] B. Lantz, B. Heller, and N. McKeown, "A Network in a Laptop: Rapid Prototyping for Software-Defined Networks," in *SIGCOMM*, ser. Hotnets-IX. New York, NY, USA: ACM, 2010.

[19] T. Koponen *et al.*, "Network Virtualization in Multi-tenant Datacenters," in *USENIX NSDI 14*. Seattle, WA: USENIX, Apr. 2014, pp. 203–216.

[20] H. H. Liu *et al.*, "CrystalNet: Faithfully Emulating Large Production Networks," in *26th SOSP*. NY, USA: ACM, 2017.

[21] P. G. Kannan *et al.*, "BNV: Enabling Scalable Network Experimentation through Bare-Metal Network Virtualization," in *USENIX CSET*, ser. CSET'18. USA: USENIX, 2018, p. 6.

[22] H. Li *et al.*, "Enabling End-to-End Simulation for Host Networking Evaluation using SimBricks," 2020.

[23] J. Cao *et al.*, "TurboNet: Faithfully Emulating Networks with Programmable Switches," in *IEEE ICNP*, Madrid, Spain, 2020, pp. 1–11.

[24] F. Rodriguez *et al.*, "In-Network P4-Based Low Latency Robot Arm Control," in *Proceedings CoNEXT*. NY, USA: ACM, 2019, p. 59–61.

[25] S. D. A. Shah, M. A. Gregory, S. Li, and R. D. R. Fontes, "SDN Enhanced Multi-Access Edge Computing (MEC) for E2E Mobility and QoS Management," *IEEE Access*, vol. 8, pp. 77 459–77 469, 2020.

[26] S. Das and M. Ruffini, "PON Virtualisation with EAST-WEST Communications for Low-Latency Converged Multi-Access Edge Computing (MEC)," in *2020 OFC*, 2020, pp. 1–3.

[27] S. Vanneste *et al.*, "3DVFH+: Real-Time Three-Dimensional Obstacle Avoidance Using an Octomap," in *MORSE 2014*, 2014, pp. 89–100.

[28] F. Rodriguez *et al.*, "P4 Programmable Patch Panel (P7): An Instant 100G Emulated Network on Your Tofino-based Pizza Box," in *ACM SIGCOMM'22 Demo and Poster Session*, 2022.

[29] ——, "Network Emulation with P7: A P4 Programmable Patch Panel on Tofino-based Hardware," in *SBRC*, may 2023.

[30] ——, "Towards multiple pipelines network emulation with P7," in *NetSoft*, Madrid, Spain, Jun. 2023.