Achieving Seamless IoT Interoperability Through Data Plane Programmability

# Achieving Seamless IoT Interoperability Through Data Plane Programmability

Alexandre Heideker*, Dener Silva†, Carlos Kamienski†, Angelo Trotta*,
* *Department of Computer Science and Engineering, University of Bologna, Italy*
† *Federal University of the ABC, CMCC, Santo André, Brazil*
Emails: {angelo.trotta5, alexandre.heideker}@unibo.it, {dener.silva, carlos.kamienski
@ufabc.edu.br

***Abstract*—The rapid proliferation and vast diversity of Internet of Things (IoT) devices present significant challenges to achieving seamless interoperability within modern networks. Many legacy IoT systems still rely on outdated protocols, rendering them incompatible with newer devices and creating communication bottlenecks. Additionally, the need for real-time traffic adaptation and modification further complicates the integration of these heterogeneous systems. This paper explores the potential of data plane programmability as a solution to address these challenges, enabling dynamic control over network behavior and improving compatibility between diverse IoT devices. By allowing real-time adjustments to traffic flows and adapting protocols on the fly, programmable data planes offer a flexible and scalable approach to ensuring interoperability across both modern and legacy IoT systems. Experimental results demonstrate the feasibility and effectiveness of this approach, highlighting its potential to enhance IoT network performance and longevity.**

***Index Terms*—IoT, P4, Interoperability**

## I. Introduction

The rapid proliferation of IoT devices has created a highly diverse ecosystem, with billions of devices deployed across sectors such as healthcare, smart cities, and industrial automation. This diversity spans device types, communication protocols, and computational capabilities, which presents substantial challenges in achieving seamless interoperability within modern networks. Many IoT devices utilize different communication standards, such as Wi-Fi, Bluetooth, Zigbee, or proprietary protocols, making it difficult for them to communicate effectively [1]. This lack of standardized protocols causes fragmentation, limiting the ability of IoT applications to fully integrate and function efficiently across different platforms. Addressing this fragmentation is crucial for realizing the full benefits of IoT, especially in environments where multiple devices and systems must coexist seamlessly.

Furthermore, this heterogeneity complicates network management, especially as many IoT devices operate within legacy systems that are not easily compatible with modern technologies. Legacy systems often rely on outdated IPv4 standards, while newer IoT devices utilize IPv6, creating additional barriers to interoperability [2]. The absence of a unified communication framework exacerbates this issue, requiring innovative solutions to bridge these gaps. As the IoT ecosystem continues to expand, ensuring device and network interoperability will be crucial to unlocking the full capabilities of IoT technologies.

On the other hand, the need for a real-time environment is widely claimed in traceability solutions [3], medical monitoring [4], and Industry 4.0 [5] - some solutions are unfeasible without a real-time response, such as autonomous vehicles. The construction of this environment involves the development of devices with adequate computational capacity, high-performance software platforms, and a network infrastructure capable of minimizing information delivery delays. Also, the modern network architecture infrastructure can play a crucial role in the security and interoperability needs of software-defined networks (SDN) [6].

In the last decades, a growing move to take control of the network behavior has taken place in Telecom providers and large private networks - the Software-defined Networks (SDN) [7]. The first approach was to remove the decision power and management from forwarding devices, i.e., the control plane, providing centralized access, control, and programmability without proprietary solutions. Although the OpenFlow protocol for the control plane significantly improves network control and management, SDN-enabled devices provide fixed support for a currently used set of protocols—hardware providers must implement new protocols and low-level packet processing behaviors. The next step is the data-plane programmability through Programming Protocol-Independent Packet Processors (P4) language [8]. The P4 can support new protocols before the OpenFlow by building specific flux tables and extending the hardware support. Also, data plane programmability supports changes in packet fields and payload based on the algorithm implemented by the P4 [9].

In this work, we explore the data plane programmability to steer and modify IoT traffic, an essential tool for interoperability. In our experiments, we modify the destination of a supposedly legacy traffic, its distribution between different destinations, and the application protocol intervention directly in the P4-enabled device. Also, we perform a novel technique to convert TCP traffic into a UDP datagram directly in a P4 switch, providing low-level network protocol interoperability.

The performance evaluation shows that P4-based solutions can provide a real-time solution to promote IoT interoperability in low-level network protocols as well as application-level protocols, such as MQTT.

Key contributions of this paper include:

- Exploration of the capability of data-plane programmabil-

ity to provide seamless interoperability focuses on how programmable data planes enable real-time adjustments to network protocols and traffic steering in heterogeneous IoT device ecosystems.
- Presenting the testbed environment used to evaluate the feasibility of the proposed solution involves detailing the setup that simulates real-world IoT scenarios, allowing for the practical testing of data-plane programmability.
- A novel technique to convert TCP segments into a UDP datagram promoting interoperability and extending the life cycle of legacy IoT devices.
- The proposed scenario's performance evaluation, which considers the trade-offs and overhead introduced by P4-based programmability, highlights how P4's flexibility in managing IoT traffic comes at the cost of added computational complexity and potential performance bottlenecks.

The rest of the paper is structured as follows. Section II presents the related work. Section III explores the real-time challenge and presents the data plane programmability as a tool for interoperability. Section IV details the testbed used to evaluate the proposed scenarios Section V discusses the experimental results and their value face real-time constraints. Section VI concludes the work.

## II. Related Work

The ability to change the packet content, aggregate, or discard it in the forwarding time inside the switch enables P4 to provide a solution tool for IoT real-time traffic demands. An IoT traffic specificity is a huge number of small packets - i.g. the payload of LoRa [10] radio protocol typically has 50 bytes. In Wang et al. [11], authors implement an aggregation method of small packets using P4, as well as its disaggregation restores the original traffic profile. The proposed solution was evaluated in a switch that uses the Barefoot Tofino chip - the state-of-the-art P4 hardware - achieving 100 Gbps throughput.

P4 is a specific-purpose language that considers restrictions on time-to-forward packets, avoiding loops and more complex code structures. However, sophisticated mathematical methods can be implemented, as in Wu et al. [11], in which the authors implement the Chinese remainder theorem in P4 to make an arithmetic aggregation encode and decoder of packets.

In a high-level intervention, Banno and Osawa [12] use P4 to accelerate the MQTT-SN, a new IoT protocol evolved from MQTT. Authors use the P4-enabled switch to implement part of the broker functionality. They conducted a performance evaluation in the Mininet simulator, showing latency reduction between publisher and subscriber. Arslan et al. [13] use P4 to change the behavior of the congestion control mechanism to improve bandwidth usage in larger bandwidth-delay appliances. The proposed mechanism, a.k. Bolt, reducing the latency to 80% and keeping til 400Gbps of line-rate utilization. In a P4-enabled network environment, this proposal can be put into production after all tests and evaluation without hardware replacement, reducing the CAPEX and e-wast production.

Further advancements in using P4 for IoT traffic management are explored by Madureira et al. [14], who introduce IoTP, a protocol that embeds data aggregation algorithms within hardware switches. By leveraging P4's programmability, the solution improves network efficiency, showing how P4 can be used to reduce redundant traffic and optimize bandwidth in resource-constrained IoT environments. Similarly, Uddin et al. [15] propose the Muppet architecture, which integrates P4 and SDN to create a multi-protocol edge switching system for large-scale IoT deployments. The Muppet switch allows low-latency communication between IoT devices using different protocols, such as BLE and Zigbee, without requiring modifications to the devices themselves. This cross-protocol interoperability demonstrates P4's ability to enhance communication between heterogeneous IoT devices. Finally, Laki et al. [16] present P4Pi, a platform designed for networking education that uses Raspberry Pi to teach P4 programming and practical networking concepts. While the focus of P4Pi is educational, the platform showcases how P4 can be implemented in low-cost hardware for real-world IoT applications, making it accessible for use in practical deployments.

## III. Real-time interoperability

It is a chaotic scenario—this is an intuitive conclusion about the growing adoption of smart devices and their impact on network traffic using traditional network technologies. Real-time response is mandatory in scenarios like Industry 4.0, smart urban transportation, self-driven cars, health care, and so on.

This demand can be translated in numbers, like in Chauhan et al. [17] showing hundreds of hours by year spent looking for parking lots in big cities. Even the transition to the sixth-generation wireless networks demands a solution to providing a reliable network service in rural areas or damaged infrastructures [18], sustaining real-time services. The effort to mitigate latency is a recurrent research subject by proposing new infrastructure technologies and distributing the computation through the network Edge-Fog-Cloud continuum [19], [20]. The step forward in using idle computational power at the edge of the network, a.k.a. fog, is using network devices themselves. These devices' notorious performance requirements, along with programmatic skills, can potentially contribute to mitigating delays. On the other hand, scalability and interoperability solutions can be improved in a software-defined network architecture using control and data plane programmability—prioritization, as well as preprocessing, can be made directly in the switches.

### A. Data-plane Programmability

Data plane programmability demands a specific-purpose programming language based on a different hardware architecture regarding memory organization, I/O operations, and CPU functionalities. Performance constraints impose a high-quality code level and prohibit using certain structures like loops and high-complexity mathematical operators [8]. The time spent processing a packet must be adequate to the nominal switch bandwidth, avoiding delays and buffering.

Defined by Open Network Foundation, the V1Model[1] architecture in Fig.1 defines the main structures of a P4 switch dataflow: the Parser is a state machine that decodes and identifies each incoming packet; the Ingress and Egress Match-Action defines flux tables and interacts with the SDN controller through OpenFlow; the Traffic Manager applies rules and algorithms to the packet; and finally, the Deparser assembly the outgoing packet.

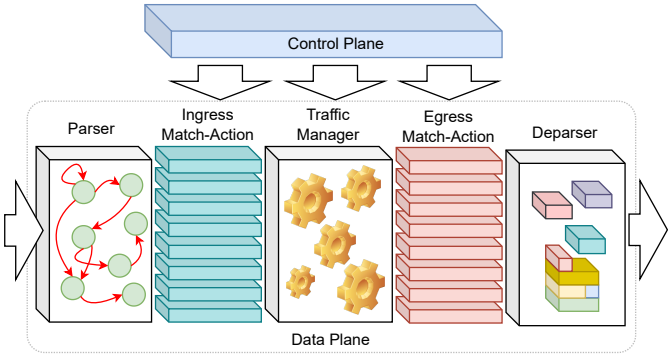


Fig. 1: V1Model Switch architecture.

Although SDN is widely adopted by cloud and telecom operators nowadays, P4-enabled appliances and soft switches can improve their skills, expand OpenFlow support to new versions, and perform real-time operations. Except for the scenario of converting TCP to UDP protocols, all the proposed experiments can be performed using OpenFlow and its Controller actuation. Still, the performance and scalability constraints are the reason for the claim of P4 adoption.

On another front, efforts to move the computation to the network edge, integrating Cloud and Fog infrastructure, can improve the performance by reducing delays and bandwidth usage [21]. Initiatives like SmartNICs and P4Pi [16] can be used in this scenario, moving P4 algorithms to the edge. SmartNIC is a network adapter that implements a P4 switch using an FPGA chip, and P4Pi is a P4-enabled soft switch implemented using the Raspberry Pi SBC. Fig.2 shows a typical scenario where a P4-enabled switch is between the traffic generation and its destination. This is a generalization, considering the switch can be placed at the edge or in the core telecom infrastructure—the reason why the proposal is founded on scalability constraints.

## IV. Methodology

P4 is a specific purpose programming language, and its learning requires a shift in skill sets for network engineers and developers. These specialized skills face limited hardware support constraints, security concerns, and the complexity of custom packet processing [22]. It is essential to mitigate these issues and promote the dissemination of the P4 simulation and emulation tools, considering the high cost of specialized hardware, such as FPGA or Intel Tofino-based switches. In between, better available choices are Mininet and P4Docker,

[1]https://p4.org/

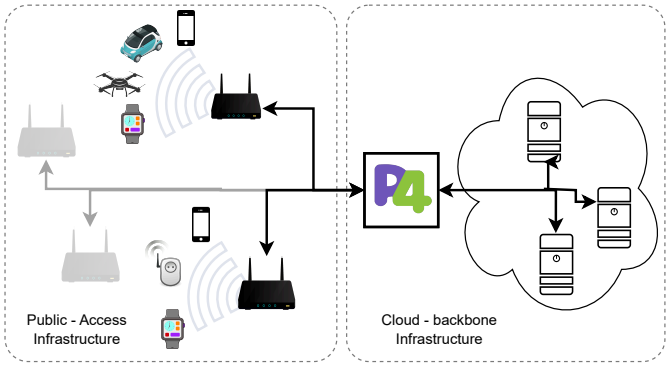


Fig. 2: This is a general scenario using P4 as an interoperability solution. The P4-enabled appliance is placed in telecom infrastructure to act in IoT traffic.

all based on the BMV2 soft-switch. Mininet is a well-known network simulation environment that runs on a single machine. The P4Docker is a new tool based on Docker, which provides flexibility and scalability in the experimentation of more complex scenarios [23]. The experimental scenarios were executed on a Ubuntu 22 virtual machine configured with 1 vCPU running at 3.2 GHz and 2 GB of RAM. The processing time between packet creation and its arrival at the destination was calculated for all experiments. To achieve this, we used the P4FlowForge tool[2], a traffic generator capable of producing UDP, TCP, and MQTT packets, among others. This tool can embed the packet creation timestamp in nanoseconds within the payload, allowing the destination to calculate the interval between the creation and arrival of each packet with high precision.

We tested workloads with 100, 200, and 300 messages per second. These rates were chosen based on preliminary experiments, avoiding overload in the hosts. This suggests that they provide a balanced understanding of the switch behavior under different workloads and allow us to observe variations in processing times. This range of workloads enables us to assess how the system performs as traffic intensity increases, offering insight into its scalability and responsiveness under real-time constraints. We applied a 99% confidence interval for all scenarios to ensure statistical significance in 30 repetitions.

## V. Evaluation and Results

In the first scenario, the traffic comes from a sensor or a group of sensors that needs to be redirected to another destination host. It is a typical network management task and an alternative solution to changing the target of legacy devices without support for manually changing their destination. The ingress packet is parsed, and the destination field of the IP header is rewritten. Fig. 3 depicts this scenario, and Fig. 4 shows the packet delay of the experiment. The baseline is the delay without any packet modification. The results show an expected increase in delay, but the overlapping of confidence intervals denotes an insignificant overhead.

[2]https://github.com/dnredson/p4flowforge

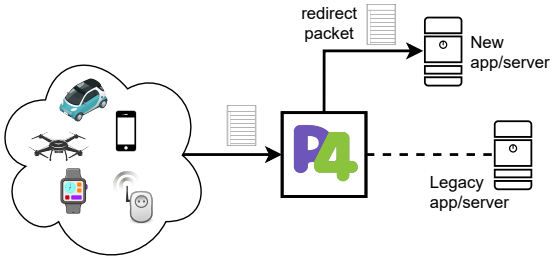


Fig. 3: Steering traffic using P4. The switch identifies the IoT traffic, and its destination address field is rewritten.

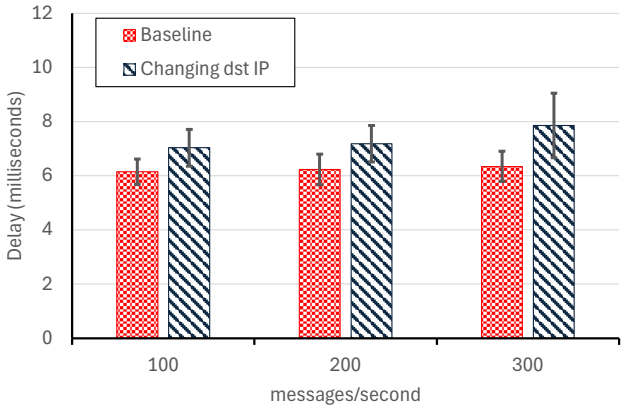


Fig. 4: Delay between the traffic generator and the destination host. The baseline shows the delay between the traffic generator and the original destination host.

The second scenario explores P4's ability to change the payload content. Even with the same host destination, sometimes the application target must be changed, as seen in Fig. 5. A typical task is changing the MQTT topic, and considering the same situation described in the first experiment, the data plane can make this modification. The implementation of this technique includes the definition of a new kind of protocol header, including the TCP fields and part of the payload as a personalized field. It is a high-cost operation, as shown

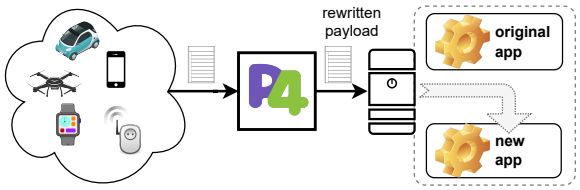


Fig. 5: Change in the high-level application protocol. In this scenario, the payload is changed to redirect the traffic to another application.

in Fig. 6, although this is a characteristic of the testbed software. The switch architecture is quite different from the PC architecture used in the experiments, and the translation of the P4 code by Target-independent Compiler for Protocol-independent Packet Processors (T4P4S) [24]. Even with this performance overhead, the time added to change the topic is about 30 milliseconds in the worst case.

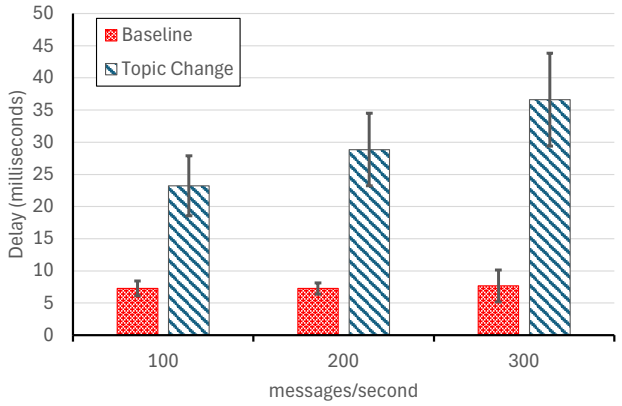


Fig. 6: Delay in changing the destination topic in MQTT traffic compared with the original traffic as a baseline.

The third scenario assesses the switch's capability to clone packets, shown in Fig. 7 - some legacy switches have this skill natively, but using P4 can improve this capability by choosing the protocol, field value, or even the payload content to trigger the clone operation.

Even though this kind of intervention increases network usage, the action must be moved close to the core infrastructure, reducing traffic at the network's edge. A similar solution is to subscribe the second host to the first one in a publish-subscriber architecture, but the delay imposed by this cascade operation may be prohibitive, depending on the application constraints.

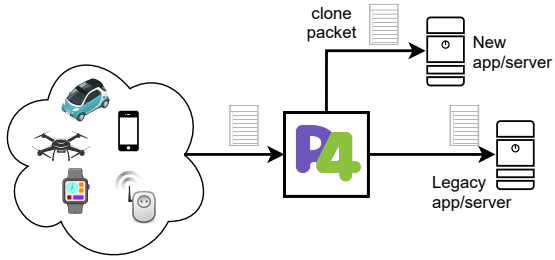


Fig. 7: Clone traffic scenario. In this scenario, the IoT traffic is cloned to another destination.

Fig. 8 shows the delay between the traffic generator and the original target host as a baseline and between the traffic generator and the two hosts when the packet is cloned. The smaller delay in the cloned packet, even inside the confidence interval, is due to the cloned packet being processed before the original one.

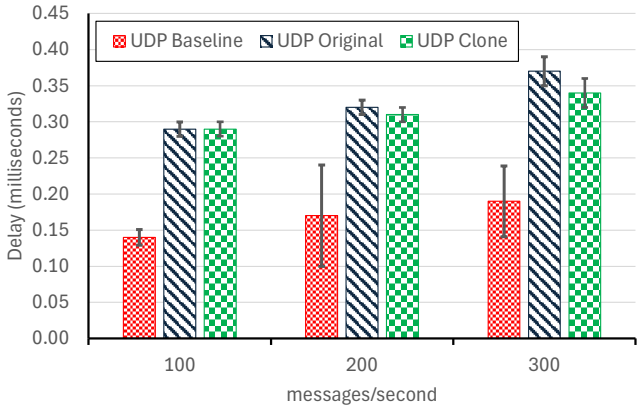


Fig. 8: The delay between the traffic generator and the target hosts in the cloning scenarios. The baseline is the delay without cloning between the traffic generator and the original host.

The last scenario, shown in Fig. 9, increases the complexity of the switch intervention in the cloning operation, converting a TCP packet into a UDP clone. Interoperability scenarios include operating in multiple protocols simultaneously, i.e., an application using a TCP MQTT client and cloning its traffic into an MQTT-SN using UDP to another broker.

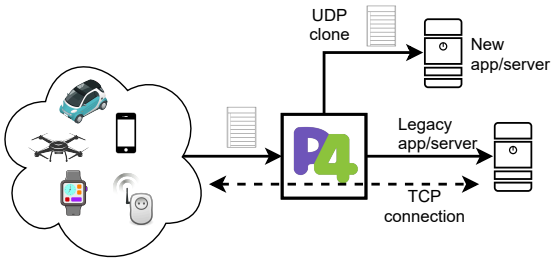


Fig. 9: Change in the network low-level protocol. In this scenario, a UDP clone of the TCP segment is created and sent to another application/server.

An ingress TCP packet with chosen characteristics triggers the cloning process. The packet is flagged as a clone, and the ingress match-action structure captures it. The new UDP header is composed using part of the original TCP header. Fig. 10 shows the delay between the traffic generator and the original target using the TCP protocol, increasing by about 4 milliseconds in the worst case. Fig. 11 shows the delay to the new target of cloned packets in the UDP protocol, increasing by about 0.3 milliseconds in the worst case.

### A. *Discussions*

Although the P4Docker testbed can implement large and highly complex scenarios, one of its cornerstones is the BMV2 soft switch, which is not recommended for production purposes. The performance of VMV2 can't be compared with

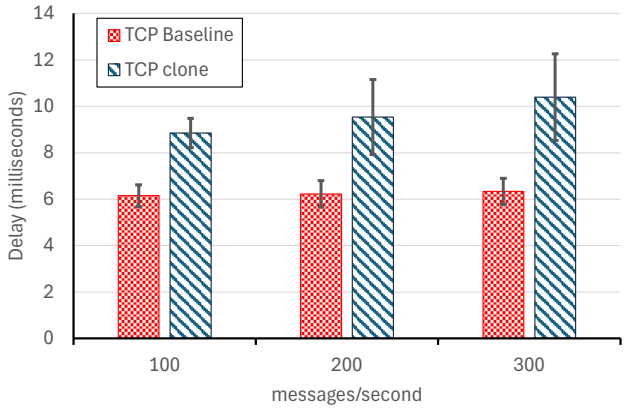


Fig. 10: The delay between the traffic generator and the TCP original target.

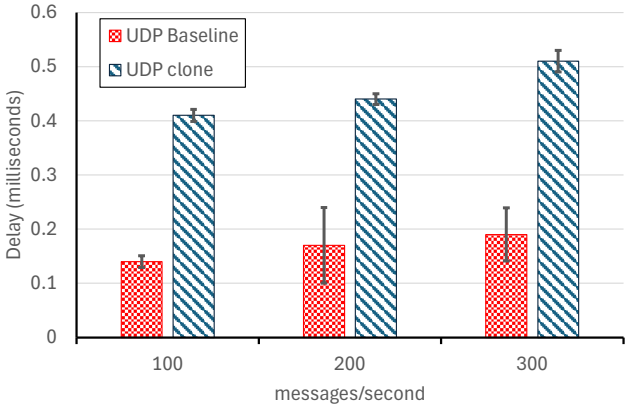


Fig. 11: The UDP cloned packet delay between the traffic generator and the new target.

P4 hardware solutions, but it can validate the behavior of new ideas, as this paper shows.

Another performance issue is using the T4P4S framework, which translates the P4 code into a C language code that can run in an architecture other than P4 hardware. Although the C code grants high performance due to its low-level proximity, T4P4S must translate calls to the specialized hardware structure in a switch architecture into an alternative code.

Even without expecting outstanding quantitative performance from the testbed, the results can be used as a comparative parameter. In industrial real-time applications, the admitted delay in a switch is 10 milliseconds [25]. Except by Topic's Changing scenario - in Fig. 5 - all experimental results impose a delay below this reference.

The best available solution admits a maximum of 30 microseconds, and this performance level can be reached using P4 hardware. The performance of FPGA or Tofino-based solutions operates with a typical delay of 30 nanoseconds, and the worst case in a Tofino switch is 100 microseconds [26]. Our algorithms implemented in P4 hardware can easily meet real-time requirements.

Finally, converting TCP segments into a UDP datagram directly in the switch is a novel solution. It does not require computational infrastructure and avoids privacy constraints since the packet is processed directly in the network.

## VI. Conclusions

This paper evaluated data plane programmability as a tool to acquire interoperability in real-time scenarios, a seamless solution in performance and scalability, for a high-demand IoT environment. We presented a performance evaluation in four different scenarios using switch-based solutions in P4, where computational solutions are traditionally used. We also presented the payload intervention and TPC segment into UDP datagram conversion directly into the data plane, a novel serverless technique. Our results show the feasibility of the proposed solutions in the P4-enabled devices, ensuring an inexpressive increase in delay in real-time scenarios.

In future work, we hope to implement new algorithms with higher levels of complexity as well as performance evaluation on P4-enabled hardware, such as Tofino or FPGA-based switches.

## Acknowledgment

The work has been supported by the PRIN 2022 project RAIN4C - "Reliable Aerial and satellIte Networks: joint Communication, Computation, Caching for Critical scenarios" (Project id: 20227N3SPN, CUP: J53D23007020001).

This work was supported by the São Paulo Research Foundation (FAPESP), Brazil, under Grant 20/05152-7.

## References

[1] M. Noura, M. Atiquzzaman, and M. Gaedke, "Interoperability in internet of things: Taxonomies and open challenges," *Mobile networks and applications*, vol. 24, pp. 796–809, 2019.

[2] E. Saavedra *et al.*, "Leveraging iot harmonization: An efficacious nb-iot relay for integrating 6lowpan devices into legacy ipv4 networks," *Applied Sciences*, vol. 14, no. 8, p. 3411, 2024. [Online]. Available: https://www.mdpi.com/

[3] M. N. M. Bhutta and M. Ahmad, "Secure identification, traceability and real-time tracking of agricultural food supply during transportation using internet of things," *IEEE Access*, vol. 9, pp. 65 660–65 675, 2021.

[4] H. Mukhtar, S. Rubaiee, M. Krichen, and R. Alroobaea, "An iot framework for screening of covid-19 using real-time data from wearable sensors," *International journal of environmental research and public health*, vol. 18, no. 8, p. 4022, 2021.

[5] I. Behnke and H. Austad, "Real-time performance of industrial iot communication technologies: A review," *IEEE Internet of Things Journal*, 2023.

[6] S. S. A. Gilani, A. Qayyum, R. N. B. Rais, and M. Bano, "Sdnmesh: An sdn based routing architecture for wireless mesh networks," *IEEE Access*, vol. 8, pp. 136 769–136 781, 2020.

[7] N. Foster, N. McKeown, J. Rexford, G. Parulkar, L. Peterson, and O. Sunay, "Using deep programmability to put network owners in control," pp. 82–88, 2020.

[8] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[9] B. Goswami, M. Kulkarni, and J. Paulose, "A survey on p4 challenges in software defined networks: P4 programming," *IEEE Access*, vol. 11, pp. 54 373–54 387, 2023.

[10] S.-Y. Wang, C.-M. Wu, Y.-B. Lin, and C.-C. Huang, "High-speed data-plane packet aggregation and disaggregation by p4 switches," *Journal of Network and Computer Applications*, vol. 142, pp. 98–110, 2019.

[11] X. Wu, Z. Jin, W.-K. Jia, and X. Shi, "Aggregating multiple small-data frames using arithmetic encoding in p4 switches," in *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2021, pp. 1–6.

[12] R. Banno and K. Osawa, "Acceleration of mqtt-sn protocol using p4," in *2022 IEEE 11th International Conference on Cloud Networking (CloudNet)*. IEEE, 2022, pp. 16–21.

[13] S. Arslan, Y. Li, G. Kumar, and N. Dukkipati, "Bolt: Sub-RTT congestion control for Ultra-Low latency," in *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. Boston, MA: USENIX Association, Apr. 2023, pp. 219–236. [Online]. Available: https://www.usenix.org/conference/nsdi23/presentation/arslan

[14] A. L. R. Madureira, F. R. C. Araújo, and L. N. Sampaio, "On supporting iot data aggregation through programmable data planes," *Computer Networks*, vol. 177, p. 107330, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128620301195

[15] M. Uddin, S. Mukherjee, H. Chang, and T. V. Lakshman, "Sdn-based multi-protocol edge switching for iot service automation," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 12, pp. 2775–2786, 2018.

[16] S. Laki, R. Stoyanov, D. Kis, R. Soulé, P. Vörös, and N. Zilberman, "P4pi: P4 on raspberry pi for networking education," *ACM SIGCOMM Computer Communication Review*, vol. 51, no. 3, pp. 17–21, 2021.

[17] V. Chauhan, M. Patel, S. Tanwar, S. Tyagi, and N. Kumar, "Iot enabled real-time urban transport management system," *Computers & Electrical Engineering*, vol. 86, p. 106746, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790620306017

[18] A. Trotta, A. Heideker, I. Zyrianoff, G. Interdonato, and S. Pizzi, "(poster) advancing non-terrestrial networks for critical scenarios with the rain4c framework," in *2024 20th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*, 2024, pp. 780–782.

[19] F. B. Oliveira, M. Di Felice, and C. Kamienski, "Iotdeploy: Deployment of iot smart applications over the computing continuum," *Internet of Things*, vol. 28, p. 101348, 2024.

[20] C. Kamienski, I. Zyrianoff, L. F. Bittencourt, and M. Di Felice, "Iotinuum: The iot computing continuum," in *2024 20th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*. IEEE, 2024, pp. 732–739.

[21] L. Gigli, I. Zyrianoff, F. Zonzini, D. Bogomolov, N. Testoni, M. D. Felice, L. De Marchi, G. Augugliaro, C. Mennuti, and A. Marzani, "Next generation edge-cloud continuum architecture for structural health monitoring," *IEEE Transactions on Industrial Informatics*, vol. 20, no. 4, pp. 5874–5887, 2024.

[22] E. F. Kfoury, J. Crichigno, and E. Bou-Harb, "An exhaustive survey on p4 programmable data plane switches: Taxonomy, applications, challenges, and future trends," *IEEE Access*, vol. 9, pp. 87 094–87 155, 2021.

[23] D. Silva, A. Heideker, L. Trombeta, B. Carvalho, J. Kleinschmidt, and C. Kamienski, "P4docker: Enabling efficient p4 switch testbeds with docker integration," in *Extended Proceedings of the XLII Brazilian Symposium on Computer Networks and Distributed Systems*. Porto Alegre, RS, Brasil: SBC, 2024, pp. 1–8.

[24] P. Vörös, D. Horpácsi, R. Kitlei, D. Leskó, M. Tejfel, and S. Laki, "T4p4s: A target-independent compiler for protocol-independent packet processors," in *2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR)*, 2018, pp. 1–8.

[25] T. Leyrer, P. Varis, W. Wallace, P. Gangadar, M. Mandhana, P. Jayarajan, and S. Karaiyan, "Analysis and implementation of multi-protocol gigabit ethernet switch for real-time control systems," in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, 2021, pp. 1–6.

[26] D. Franco, E. O. Zaballa, M. Zang, A. Atutxa, J. Sasiain, A. Pruski, E. Rojas, M. Higuero, and E. Jacob, "A comprehensive latency profiling study of the tofino p4 programmable asic-based hardware," *Computer Communications*, vol. 218, pp. 14–30, 2024.