Enabling Cloud-native IoT Device Management

Anastassios Nanos
ananos@nubis-pc.eu
Nubis PC

Ioannis Plakas
iplakas@nubis-pc.eu
Nubis PC

Georgios Ntoutsos
gntouts@nubis-pc.eu
Nubis PC

Charalampos Mainas
cmainas@nubis-pc.eu
Nubis PC

ABSTRACT

As the Internet of Things (IoT) continues to proliferate, the integration and management of IoT devices within higher-level orchestration frameworks have become paramount. The diverse nature of IoT devices, along with their resource constraints, necessitate a holistic approach to orchestration for seamless integration into cloud-native environments. While K8s has emerged as the de-facto standard for container orchestration, its inherent complexity presents a significant barrier to the management of such devices.

We look into the implications of bringing IoT devices closer to the cloud-native ecosystem, highlighting the need for orchestration simplification in the context of edge computing. Furthermore, we advocate for a unified approach to application deployment across the Cloud-Edge-IoT continuum, highlighting deployment methodologies for diverse and distributed applications. This work aims to contribute to the implementation of a more cohesive and efficient continuum for the deployment and management of applications in IoT environments. In particular we introduce simple enhancements to a popular IoT management framework (Akri) to reduce expected resource utilization for Edge gateways and take the first step towards a fully unified infrastructure management solution, based on cloud-native concepts.

KEYWORDS

IoT, cloud-native, k8s, containers

ACM Reference Format:
Anastassios Nanos, Ioannis Plakas, Georgios Ntoutsos, and Charalampos Mainas. 2024. Enabling Cloud-native IoT Device Management. In *Proceedings of International Workshop on MetaOS for the Cloud-Edge-IoT Continuum (MECC'24).* ACM, New York, NY, USA, 6 pages.

MECC'24, April 22nd 2024, Athens, Greece

1 INTRODUCTION

The advent of the Internet of Things (IoT) has ushered in an era of unprecedented connectivity, transforming the way we interact with and perceive the world [6]. As billions of IoT devices spread throughout our environments [3], the need for secure and robust device management with seamless integration into higher-level orchestration frameworks, becomes increasingly pressing [7]. In this work, we address the pivotal role of IoT device management in the broader context of orchestration, highlighting the challenges posed by resource scarcity at the edge. While Kubernetes (K8s) has emerged as the standard for container orchestration, its complexity appears to be a bottleneck, particularly when orchestrating applications on resource-constrained edge devices. We aspire to contribute to the development of more accessible and efficient orchestration solutions that cater to the diverse and resource-limited nature of IoT environments.

Specifically, we investigate the complexities of managing IoT devices within the context of higher-level orchestration frameworks and assess the challenges posed by resource constraints at the edge [4], while examining their impact on orchestration using K8s [2]. Additionally, we propose strategies for simplifying popular IoT management frameworks (KubeEdge and Akri) to enhance their usability for managing IoT devices.

Building on common concepts introduced by the research community [5], we introduce a unified approach to application deployment across the whole continuum (Cloud, Edge, IoT), fostering seamless operation in diverse multi-architecture environments. Moreover, we contribute insights and recommendations for the development and deployment process of applications, bridging the gap between IoT device management and higher-level orchestration.

The rest of this paper is organized as follows: Section 2 presents the challenges related to IoT device management. In Section 3, we discuss the device management options

available, mainly related to cloud-native solutions, namely KubeEdge and Akri. Section 4 provides the design of our approach and the changes we introduce to Akri, while Section 5 concludes, presenting our plan and next steps.

2 MOTIVATION

The management of IoT devices presents several unique challenges due to the diverse nature of IoT ecosystems, the scale of deployments, and the necessity for secure and efficient operations. We group the identified challenges into three major categories: (i) Firmware management, (ii) Orchestration, (iii) Security and Onboarding.

Addressing these challenges requires a holistic approach to IoT device management that involves robust security practices, scalable architectures, interoperable standards, and efficient lifecycle management strategies. In this work, we lay the ground for tackling some of these challenges using well-known approaches from the Cloud ecosystem. Although our work is far from complete, introducing mechanisms that facilitate the integration of IoT devices as first-class citizens to the rest of the infrastructure becomes a necessity. As IoT continues to evolve, ongoing efforts are crucial to overcoming these challenges and ensuring the reliable and secure operation of IoT devices in diverse environments.

2.1 IoT Device Management Challenges

In what follows, we try to identify the key challenges related to IoT device management.

2.1.1 Firmware management. Device heterogeneity is by far the most critical challenge to address, as the diversity of hardware is vast and there is no common standard or protocol used by IoT device vendors. Ensuring seamless communication and interoperability among devices from different manufacturers can be challenging. Over-the-air (OTA) *Firmware updates* must be conducted efficiently and securely to address vulnerabilities, introduce new features, and fix bugs. Implementing effective rollback strategies in case an update fails or causes issues is not only challenging, but critical as well. The absence of universal standards for IoT devices can hinder *interoperability* and create challenges in integrating devices from different vendors. Devices using different communication protocols may struggle to communicate seamlessly within an IoT ecosystem.

2.1.2 Orchestration. Managing a vast number of IoT devices in large-scale deployments requires robust and scalable orchestration and device management solutions. The sheer volume of data generated by numerous devices can strain network bandwidth and storage capabilities, so an efficient mechanism to store, filter, or even perform analytics on such data is crucial.

Additionally, IoT devices may operate in environments with intermittent connectivity, leading to challenges in real-time monitoring and management. Ensuring secure and reliable communication channels and protecting devices from network-based attacks are ongoing concerns.

2.1.3 Security and Onboarding. Implementing strong authentication mechanisms and proper authorization controls is crucial for preventing unauthorized access. Ensuring end-to-end encryption of data to protect sensitive information from interception is also a constant challenge, as well as the security handling throughout the entire device lifecycle, from manufacturing to decommissioning.

Another key challenge is how to onboard a new hardware device to administrative domains: streamlining the provisioning and onboarding process for new devices, including initial configuration and setup, is a major point of discussion. The same stands for Zero-touch provisioning, in order to minimize manual intervention during deployment.

Finally, challenges related to resource allocation, end-of-life handling and regulatory compliance are also crucial, but out of scope of this work.

2.2 Cloud-native ecosystem

The cloud-native concept represents a paradigm shift in the way applications are designed, developed, deployed, and managed. This would not have been possible without the use of containers, an application packaging technology that has revolutionized application deployment in diverse environments.

Cloud-native refers to an approach in software development and deployment that leverages cloud computing and embraces key principles such as scalability, resilience, automation, and agility. Particular emphasis is given in the use of containerization and micro-services as modern practices to optimize the development and operational processes.

Essentially, containers are lightweight, portable, and isolated units that encapsulate an application along with its dependencies, libraries, and runtime. The key features of containers consist of rapid deployment, efficient resource utilization, and consistency among diverse environments. The inherent interoperability aspect of containers gave them a huge advantage when all others solution would require considerable amount of engineering effort to package, diversify and deploy. For instance, binary package management systems suffer from dependency resolving issues and pollute the surrounding software stack.

Containers provide flexibility in deploying applications across diverse environments, including on-premises data centers, public clouds, and hybrid cloud setups. This flexibility aligns with the evolving needs of organizations leveraging

multiple cloud providers or maintaining a hybrid infrastructure.

Deploying containers on leaf devices poses several challenges due to the resource constraints and unique characteristics of these devices. IoT devices typically have low-powered processors, with limited support for generic operating systems, making it hard or even impossible to run containerized applications. Additionally, IoT devices often have limited RAM, and containers, along with their management stack, can consume a significant portion of available memory. This constraint may lead to performance issues and potential system instability. Finally, storage space on IoT devices is often severely limited. Containers and their associated images can consume a considerable amount of storage.

As a result, the community has proposed alternative solutions for initial integration of IoT devices to the cloud-native ecosystem. Management frameworks (dashboards, key-value stores, message queue systems) are running as cloud-native micro-service deployments in Edge devices with close proximity to the IoT infrastructure. To communicate with the leaf devices, specialized micro-services are deployed at these Edge devices, which essentially act as proxies/gateways. In the next section we describe two of the most popular cloud-native management frameworks for Edge/IoT devices: *KubeEdge* and *Akri*.

3 IOT DEVICE MANAGEMENT

In this section we briefly present available solutions for IoT device management in K8s, namely *KubeEdge* and *Akri*. Additionally, we introduce the cloud-native concept, discuss how these frameworks integrate with the cloud-native ecosystem and elaborate on their merits and limitations.

3.1 Cloud-native concepts

Kubernetes (k8s) [1], stands as the preeminent open-source container orchestration platform that plays a pivotal role in the seamless deployment and management of containerized applications. At its core, K8s automates the deployment, scaling, and operation of application containers, providing a robust infrastructure for building, running, and orchestrating distributed systems. In the context of IoT, K8s serves as a foundational layer for external service orchestration, efficiently managing the life-cycle of micro-services that facilitate IoT device functionality. K8s abstracts away the complexities of underlying infrastructure, allowing developers to focus on building and deploying applications without concerning themselves with the intricacies of service management and execution.

Contrary to Cloud and Edge deployments, IoT deployments present a particular challenge: hardware devices cannot match the resource characteristics of general purpose devices. IoT devices are usually microcontrollers with minimal compute capabilities, limited support for general purpose operating systems and specialized hardware to provide network connectivity (BLE, Zigbee, etc.). As a result, the cloud-native axiom does not hold as neither the hardware, nor the software can support running containers, the basic block of the cloud-native deployment paradigm. In the following sections we briefly analyse the chalanges related to device management and in particular firmware load and updating, and describe two of the most popular cloud-native compatible IoT management frameworks.

3.2 Firmware management

Firmware updates play a critical role across diverse sectors of IoT. In automotive applications, they are essential for fortifying vehicle security, integrating new functionalities, and rectifying performance bottlenecks to ensure safe and efficient driving experiences. Transitioning to agriculture, firmware updates enable precision farming methodologies, optimize resource allocation, and integrate seamlessly with IoT sensors to monitor crop health and environmental conditions in real-time.

Furthermore, firmware updates are integral to optimizing supply chain management processes in the retail sector. Devices such as smart shelves, tracking sensors, and inventory management systems rely on these updates to enhance real-time visibility, streamline logistics operations, and bolster overall supply chain efficiency.

With up-to-date firmware releases, retailers can orchestrate seamless coordination across their supply chains, mitigate disruptions, and effectively meet evolving customer demands.

3.3 OTA updates

Over-the-air (OTA) updates are crucial in IoT deployments as they enable the management and maintenance of devices without requiring physical access. In the realm of IoT, where devices are often distributed across diverse and sometimes inaccessible locations, the ability to remotely update firmware becomes paramount. OTA updates allow manufacturers to address bugs, introduce new functionalities, and enhance security measures without the need for physical access to each device. This capability not only streamlines the maintenance process but also minimizes downtime and operational disruptions for end-users.

Additionally, in the dynamic landscape of IoT, where security threats continually evolve, the prompt deployment of OTA updates becomes a fundamental strategy to fortify devices against emerging vulnerabilities. Ultimately, the concept of OTA firmware updates in the realm of IoT ensures

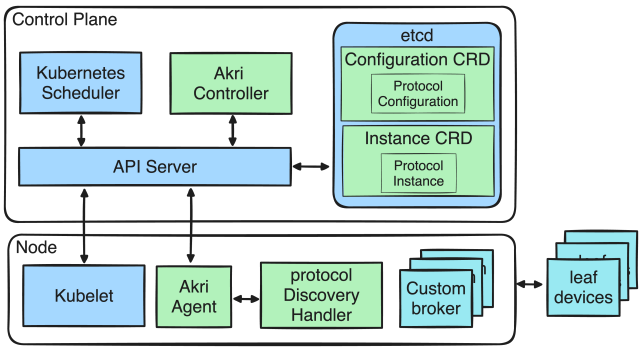

Figure 1: Akri high-level architecture

that connected devices remain resilient, adaptable, and capable of delivering optimal performance throughout their operational life cycles.

3.4 KubeEdge and Akri

KubeEdge[1], an open-source solution, expands the capabilities of native containerized application orchestration and device management to Edge hosts. Leveraging K8s, KubeEdge offers essential infrastructure services for networking, application deployment, and metadata synchronization between the IoT devices and K8s.

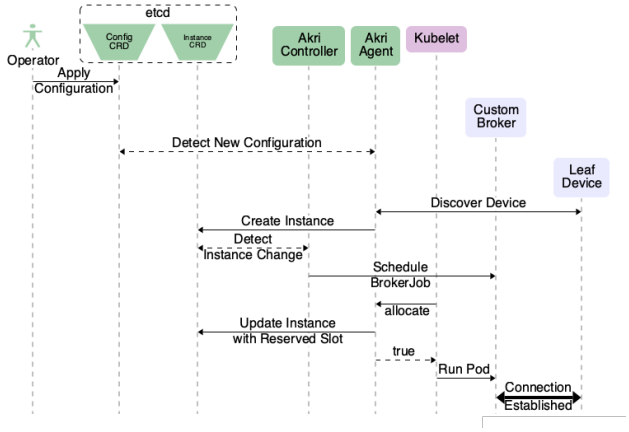

Figure 2: Device Instantiation in Akri

Akri (Agent for Kubernetes Resource and Infrastructure)[2] is a framework designed to extend K8s for managing edge devices and their resources in IoT environments. Akri focuses on enabling K8s to dynamically discover, expose, and consume edge devices as resources within the cluster.

The core of Akri architecture consists of two main components: the Agent and a custom Controller. To facilitate configuration and device discovery, Akri comes with custom resource definitions (CRDs) and Discovery Handlers.

The first custom resource, the Akri Configuration, is where devices are defined. This allows Akri to understand what kind of device it should look for. Akri's Discovery Handlers look for the device and inform the Agent of discovered devices. The Agent then creates Akri's second custom resource, the Akri Instance, to track the availability and usage of the device.

The Akri Controller enables the use of the device across the cluster. It sees each Akri Instance (which represents a leaf device) and deploys a ("broker") Pod that is tailored to each IoT device and knows how to connect and use it.

Figure 2 visualizes the workflow of the instantiation of a simple IoT device in Akri, along with the accompanying *BrokerJob* Pod.

3.5 Current limitations

While both KubeEdge and Akri provide a notion of cloud-native integration, supporting the whole life-cycle of IoT devices remains an important challenge. Both frameworks focus on the device output, and data manipulation, rather than the actual device management (firmware load and update, systems telemetry, etc.). Supporting OTA updates requires external tools, which may provide the needed functionality but need to be adapted to the cloud-native concept, bringing extra complexity and management burden. In this work, we address the issue of firmware updates by building on Akri and its mechanisms, facilitating the deployment of diverse applications on IoT devices. In the following section, we present two enhancements to the Akri architecture to address this limitation and extend its functionality to handle leaf devices at scale.

4 CLOUD-NATIVE IOT

In this section we elaborate on the changes we introduce to the Akri architecture to accommodate the integration of firmware update management of leaf devices in the cloud-native concept. We first describe the prototype device we use, along with the firmware we run and we go through the changes we have introduced, focusing on points that could further facilitate IoT device management on K8s.

4.1 IoT device firmware

To facilitate the development of our framework, we assume a simple IoT device, running an open-source-based firmware with OTA updates. We use a Rapsberry Pi Pico-W[3], with picowota[4].

[1]https://kubeedge.io/
[2]https://docs.akri.sh/
[3]https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html
[4]https://github.com/usedbytes/picowota

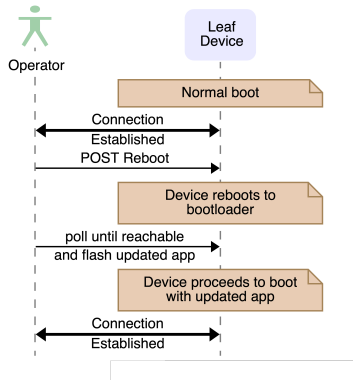

Figure 3: Example device OTA firmware update workflow

The application we run is a simple *blink* app, provided as part of the picowota examples, with additional FreeRTOS support for handling application threads and exposing HTTP endpoints for metadata querying and rebooting. The device essentially supports the following operations:

- Initial boot: The device connects to a predefined SSID, requesting an IP address via DHCP. Once connected, a thread starts to blink the LED with a specified frequency. At the same time, a separate thread is being spawned, exposing two HTTP endpoints, one for metadata querying and the other one for rebooting the device.
- Metadata endpoint: Once a GET request is received, the device responds with firmware version (embedded in the bootloader & application binary).
- Reboot endpoint: Once a POST request is received, the device reboots to the bootloader stage, waiting for an updated application binary. The device uses the existing credentials to connect to a predefined SSID, asking for an IP address via DHCP.
- Firmware Flash: Upon the receipt of an updated binary, the device proceeds to boot normally, using the updated application. The application is spawned as a separate thread, and does not affect the exposed HTTP endpoints.

Figure 3 visualizes the above steps in a sequence diagram. In the following sections we describe the changes we introduce to Akri, to support OTA updates on the example device described above.

4.2 Discovery handlers

Akri has implemented IoT device discovery via a number of protocols with diverse brokers and applications to demonstrate usage. A Discovery Handler is anything that implements the *DiscoveryHandler* service and *Registration* client defined in the Akri's discovery gRPC protobuf file. These Discovery handlers run as their own Pods and are expected to register with the *Agent*, which hosts the Registration service defined in the gRPC interface.

Our custom implementation, in addition to performing the expected operation (discovering the device and registering with the Akri Agent), populates extra metadata from the device, by querying specific HTTP device endpoints. The information gathered refers to the *firmware type*, *firmware version*, and an extra flag to allow updating or not (*lock*). This flag prevents accidental device re-purposing.

Our enhanced Discovery handler, populates the Akri Instance with this additional metadata, triggering a specific event when the *lock* flag is down and the configuration YAML presents an updated / different firmware type or version.

4.3 Device Lifecycle management

The common lifecycle of an IoT device in Akri is shown in Figure 1. Essentially, Akri assumes that the device is already functional and responsive, and manages application functions that need to accompany the fleet of IoT devices available. For instance, the common example Akri uses to describe its architecture is a set of OVNF cameras, reachable by their IP address.

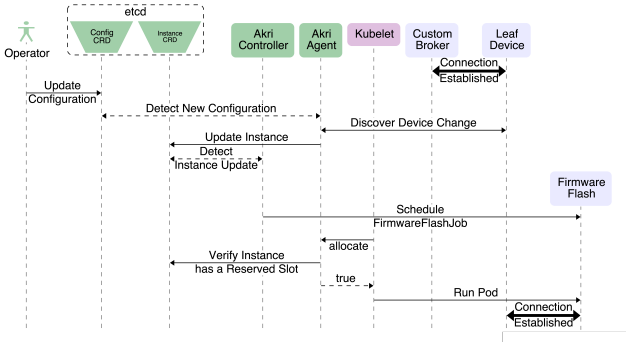

Figure 4: Configuration Update and FirmwareFlash Job trigger

In this example, it is assumed that any firmware updates, or upgrades are handled by external tools, that may or may not be deployed in the k8s cluster. Usually, such architectures assume internet connectivity provided to the devices, and their firmware polls a specific endpoint about updates. The device itself then downloads the update and performs the upgrade according to the recipe available by the vendor.

This approach presents a number of limitations: (a) first, it is assumed that the devices have public connectivity. There are numerous cases where this is not advised, as security and privacy are crucial. (b) Second, there is no unified control of the updates for these devices. If all devices are homogeneous

and perform the same task, then this is not a major issue. However, if there are diverse devices that perform different functions, then managing the firmware is a completely orthogonal task than the actual function the devices perform. Decoupling the firmware management from the actual application brings additional management complexity, which leads to significant managerial overhead.

The enhancement we introduce to Akri is the *firmware update Job*. Essentially, it is a *BrokerJob*, triggered by an event generated by an update in the configuration with the help of the *Discovery handler*. The workflow of triggering such an event is shown in Figure 4.

4.4 Firmware updates

Upon the trigger of a *Firmware update* event, the Controller schedules a Firmware update Job, a terminating Kubernetes Pod that handles the firmware device update operation. In our initial implementation, the job issues a request to reboot the device to Firmware update mode; waits for the device to be discoverable again; flashes the updated firmware OTA; and finally lets the device boot to the updated state. The pod, upon successful completion, exits, and the Akri Controller garbage-collects the completed job. This process is in par with the example IoT device operation, described in Section 4.1.

5 CONCLUSIONS

As we move towards the era of extreme heterogeneous computing, the intricate relationships between IoT device management, orchestration complexities, and the unification of application deployment in evolving IoT environments become increasingly crucial. This work aims to advance our understanding of these relationships and contribute to the implementation of a more cohesive and efficient continuum for deploying and managing applications in IoT environments.

We have described the challenges of managing low-resource IoT devices within high-level orchestration frameworks like Kubernetes, emphasizing the need for simplification in orchestration, especially in edge computing scenarios. By advocating for a unified approach to application deployment across the Cloud-Edge-IoT continuum, we aim to ensure seamless operation of diverse applications in multi-architecture environments.

Through the exploration of novel strategies for integrating IoT firmware and application management, along with harmonizing orchestration across diverse IoT devices, this work seeks to enhance the usability and efficiency of popular IoT management frameworks. The proposed enhancements to Akri are the first step towards a more streamlined and accessible management solution based on cloud-native concepts.

Addressing the challenges posed by resource constraints at the edge, we aspire to offer insights and recommendations for bridging the gap between IoT device management and higher-level orchestration. We plan to further pursue this endeavor, upstream our changes to Akri and enhance device support for this framework, while exploring the feasibility of adding a customized container runtime, able to update IoT devices directly.

ACKNOWLEDGMENTS

This work has been funded in part by the Horizon Europe research and innovation programme of the European Union, under grant agreements no 101092912 (MLSysOps) and 101136024 (EMPYREAN).

REFERENCES

[1] Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, and John Wilkes. 2016. Borg, Omega, and Kubernetes. *Commun. ACM* 59, 5 (apr 2016), 50–57. https://doi.org/10.1145/2890784

[2] Juan-Manuel Fernandez, Ivan Vidal, and Francisco Valera. 2019. Enabling the Orchestration of IoT Slices through Edge and Cloud Microservice Platforms. *Sensors* 19, 13 (2019). https://doi.org/10.3390/s19132980

[3] Dragi Kimovski, Roland Mathá, Josef Hammer, Narges Mehran, Hermann Hellwagner, and Radu Prodan. 2021. Cloud, Fog, or Edge: Where to Compute? *IEEE Internet Computing* 25, 4 (July 2021), 30–36. https://doi.org/10.1109/MIC.2021.3050613

[4] Henna Kokkonen, Lauri Lovén, Naser Hossein Motlagh, Abhishek Kumar, Juha Partala, Tri Nguyen, Víctor Casamayor Pujol, Panos Kostakos, Teemu Leppänen, Alfonso González-Gil, Ester Sola, Iñigo Angulo, Madhusanka Liyanage, Mehdi Bennis, Sasu Tarkoma, Schahram Dustdar, Susanna Pirttikangas, and Jukka Riekki. 2023. Autonomy and Intelligence in the Computing Continuum: Challenges, Enablers, and Future Directions for Orchestration. arXiv:cs.MA/2205.01423

[5] D. Spatharakis, I. Dimolitsas, G. Genovese, I. Tzanettis, N. Filinis, E. Fotopoulou, C. Vassilakis, A. Zafeiropoulos, A. Iera, A. Molinaro, and S. Papavassiliou. 2023. A Lightweight Software Stack for IoT Interoperability within the Computing Continuum. In *2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*. IEEE Computer Society, Los Alamitos, CA, USA, 715–722. https://doi.org/10.1109/DCOSS-IoT58021.2023.00112

[6] Shreshth Tuli, Fatemeh Mirhakimi, Samodha Pallewatta, Syed Zawad, Giuliano Casale, Bahman Javadi, Feng Yan, Rajkumar Buyya, and Nicholas R. Jennings. 2023. AI augmented Edge and Fog computing: Trends and challenges. *J. Netw. Comput. Appl.* 216, C (jul 2023), 28. https://doi.org/10.1016/j.jnca.2023.103648

[7] A. Zafeiropoulos, E. Fotopoulou, C. Vassilakis, I. Tzanettis, C. Lombardo, A. Carrega, and R. Bruschi. 2023. Intent-Driven Distributed Applications Management Over Compute and Network Resources in the Computing Continuum. In *2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*. IEEE Computer Society, Los Alamitos, CA, USA, 429–436. https://doi.org/10.1109/DCOSS-IoT58021.2023.00074