Cyber-Resilient Edge Computing: A Holistic Approach with Multi-Level MAPE-K Loops

Marco Stadler, Johannes Sametinger
LIT Secure and Correct Systems Lab
Dept. of Business Informatics – Software Engineering
Johannes Kepler University Linz, Austria
{first name}.{last name}@jku.at

Michael Riegler
IT Platforms and Operations
ENGEL AUSTRIA GmbH
Schwertberg, Austria
michael.riegler@engel.at

2024 IEEE 21st International Conference on Software Architecture Companion (ICSA-C) | 979-8-3503-6625-9/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICSA-C63560.2024.00020

***Abstract*—As edge computing continues to play a pivotal role in modern computing architectures, ensuring robust cybersecurity becomes imperative. This paper introduces our emerging results on a comprehensive approach to bolster the cyber-resilience of edge computing systems by incorporating the MAPE-K (Monitor, Analyze, Plan, Execute, and Knowledge) loop. The proposed methodology involves the application of the MAPE-K loop at different levels of an edge computing architecture, aiming to create a holistic defense mechanism against cyber attacks. We present a prototype of our framework and assess its viability and efficacy by leveraging real-world edge devices deployed in an industrial production setting. Initial evidence from our results suggests that this novel approach leads us to reconsider how we construct more resilient edge computing architectures.**

***Index Terms*—Edge Computing, MAPE-K Loop, Cyber Attack Resilience**

I. Introduction

Edge Computing Systems (ECSs) are more popular than ever before. Media portray ECSs as one of the most significant phenomena of our time. Recent industry forecasts predict that market values will reach 317 billion U.S. dollars by 2026 [18], and scientific research communities have demonstrated a growing interest in ECS research in recent years. For example, a search for "edge computing" in the ACM Digital Library (16.10.2023) returns 675,121 results, of which 176,725 (or more than 26%) were published only in the last five years [1].

The concept of edge computing entails the placement of substantial computing and storage resources at the outer edges of the Internet, i.e., in close proximity to the sensors [17]. For this reason, ECSs provide numerous benefits. ECSs satisfy ultra-low latency requirements, shift sensitive data processing from the cloud closer to the device, and combat the ever-increasing energy demand [3]. ECSs are employed in a wide range of application scenarios. For example, in the context of smart manufacturing [14] or cyber-physical systems [4].

While the ECS's concept is widely used and has yielded promising results thus far, these systems' unique design characteristics make them especially susceptible to malicious attacks [21]: (1) Most access control models only offer **Coarse-Grained Access Control**, which is unsatisfactory in edge computing due to the system's complexity. (2) Due to the **Operating System and Protocol Heterogeneity** utilized by ECSs, it is challenging for security engineers to design a unified protection mechanism for edge computing. (3) Due to their rudimentary design, edge devices provide limited user feedback, resulting in increased **Attack Unawareness** among users. (4) ECSs are supplied with **Weaker Computational Power** compared to cloud servers. Therefore, these systems employ weaker defense systems and are consequently more susceptible to existing attacks.

A prominent example of attacks leveraging these attack surfaces are Denial of Service (DoS) attacks, and the even more pervasive and severe Distributed Denial of Service (DDoS) attacks [19]. DoS attacks are defined as the deliberate endeavor of an attacker to obstruct the access of authorized users to the targeted resources of a service. This can be accomplished, for example, by flooding the (edge) device with requests, thereby obstructing legitimate traffic [10].

Current architectural solutions to secure ECSs against these types of attacks lack the consideration of security-by-design (i), are not able to migrate security frameworks (ii), provide only fragmented coarse-grained access control (iii), or only provide isolated and passive defense mechanisms (iv) [21]. While all of these issues must be tackled in order to ensure the security of systems, the primary emphasis of this paper will be on the latter (iv).

In summary, the threat posed by DoS attacks remains high and indicates that current countermeasures cannot prevent such attacks. New strategies will be required in the future to secure ECSs. Hence, we report in this paper our ongoing research, emerging results, and initial efforts toward securing ECS architectures by leveraging a self-adaptive holistic system architecture.

The paper commences with an illustrative example from the real world (Section II). Following this, background information is presented in (Section III), accompanied by a description of the proposed approach in Section IV. The results are examined in the discussion (Section V), with Section VI reviewing related work. Future research directions are proposed in Section VII. Section VIII provides a comprehensive summary of the contributions made in the paper.

II. Motivating Example

Azure, the cloud platform developed by Microsoft, presently holds the position of one of the most prominent cloud provider.

2768-4288/24/$31.00 ©2024 IEEE
DOI 10.1109/ICSA-C63560.2024.00020

"Azure Stack Edge" is the part of the *Azure* platform used to conduct edge computing [11].

Despite Microsoft's extensive infrastructure resources and years of experience, it remains vulnerable to DDoS attacks, as evidenced by the incident that occurred in June 2023. A massive DDoS attack caused disruptions in some of Microsoft's services, including *Azure* and its services [20]. The attackers used several types of DDoS attack traffic, including an *HTTP(S) flood attack*. In this scenario, the attacker sent a large number (millions) of HTTP(S) requests from various source IPs all over the world. As a result, the application backend's computing resources (CPU and memory) were depleted [12].

The present case demonstrates that attacks on ECSs are not only more prevalent than ever before but also capable of targeting anyone, even those with the most recent, cutting-edge security features. Hence, a question emerges as to how one might enhance the resilience of their infrastructure against such forms of attacks. We advocate for the implementation of defense mechanisms that approach architecture in a more holistic manner. In this paper, one step toward accomplishing this objective is outlined: Implementing multi-level MAPE-K loops to increase the resilience of ECSs under attack.

III. Background

Two concepts underpin our efforts to develop more resilient ECSs: the "native" edge computing architecture and the MAPE-K loop. Thus, a brief summary of these principles is presented in this section.

ECSs are frequently implemented using a hierarchical multi-layered architecture. An overview of the layers can be found in Fig. 2.

The architecture consists of multiple layers [9, 15, 22]:

- *Device Layer*: The lowest layer in the architecture, which is accountable for gathering parameter data from various sensors, transmitting it to the *Edge Layer*, and awaiting control instructions from the *Edge Layer*, thereby establishing data and control flow connectivity between the *Device Layer* and *Edge Layer*.
- *Edge Layer*: This layer provides time-sensitive services, including edge data analysis, intelligent computation, process optimization, real-time control, and security and privacy protection, in addition to receiving, processing, and forwarding data flow from the *Device Layer*.
- *Fog Layer*: This layer facilitates the transfer of data, storage, processing, and networking operations from the cloud to end devices through the utilization of network nodes that are in close proximity to the initial layers. As data proceeds from the lower layers to the *Cloud Layer*, it inevitably encounters processes such as computation, storage, networking, decision-making, and data management.
- *Cloud Layer*: The responsibility of extracting potential value from vast amounts of data and ensuring efficient resource distribution at the enterprise, regional, or national level falls within the responsibility of the *Cloud Layer*. Furthermore, by collaborating in the cloud, the *Cloud Layer* is capable of exchanging data, which enables more comprehensive and diverse data value extraction.

Contemporary software engineering systems have evolved beyond their predecessors in terms of complexity and integration. Component interactions of these enormous systems were challenging for architects. In response, self-adaptation was introduced, which enabled software systems to independently adapt to changes occurring either internally or externally in their environment, ensuring high efficiency despite the growing intricacy, unpredictability, and dynamism [7].

A widely acknowledged engineering methodology for achieving self-adaptation involves the utilization of a feedback control loop known as MAPE-K. An overview of the MAPE-K feedback loop is depicted in Fig. 1.

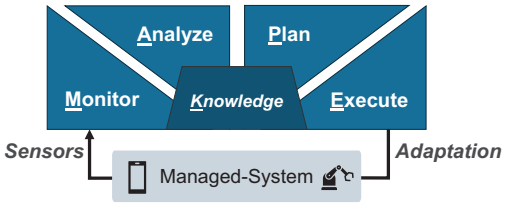

Fig. 1. MAPE-K Feedback Loop

The MAPE-K loop seeks to Monitor, Analyze, Plan, and Execute self-adaptation utilizing a common Knowledge base: A managed system and its environment are initially monitored by sensors (*M*). Then the information is compiled, analyzed, and stored in the knowledge base for future reference (*K*). In addition, patterns in the knowledge base are compared with event data to extract symptoms and patterns (*A*). The planner utilizes the sensor data to formulate a course of action recommendation and propose an adaptation strategy (*P*). The execution of this strategy ultimately leads to an adaptation of the managed system (*E*) [2, 7].

ECSs frequently possess great complexity; thus, the MAPE-K loop is regarded as an effective tool for managing these systems and, more precisely, for addressing the system's complexity.

IV. Approach

A. Multi-Level MAPE-K Loops

ECSs often follow a hierarchical System of Systems (SoS) architecture; respectively we propose to use a MAPE-K loop for a hierarchical control pattern [5] as part of our larger Security Adaptation Manager component, see Fig. 2. The hierarchical control pattern incorporates complete MAPE-K loops at each level of the hierarchy. MAPE-K loops at various levels exchange information to communicate. The MAPE-K loop at a specific level can transmit directives to the level below concerning adaptation strategies that lead after considering all other multiple relevant security intelligence in the hierarchy (multi-level) to corresponding reactions, and conversely. It can also share information about locally planned actions that it has collected, potentially aggregated, or filtered.

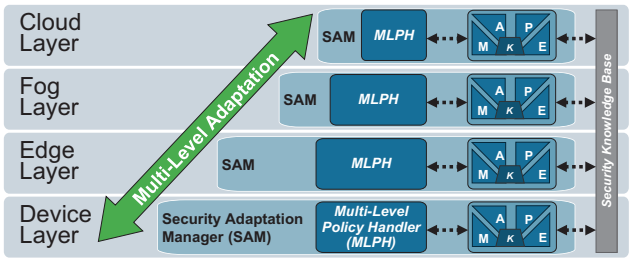

Fig. 2. Multi-Level MAPE-K Loops

We leverage this communication and adaptation flow to transmit security-relevant data (c.f. Security Knowledge Base). During an attack, a layer engages in self-adaptation (i.e., enforces security policies via the Multi-Level Policy Handler) to counter the attack and enhance its resilience. The adaptation is subsequently transmitted to the remaining layers, potentially inducing further adaptation in those layers. Thus, other layers may adapt their behavior preemptively, prior to the attack's arrival.

For example, let's consider a scenario where an attack occurs at the edge layer. The targeted device adapts its behavior (i.e., security level) in response to the attack and communicates the changes to the relevant fog layer node. Based on the level of significance, the fog layer can adjust its behavior or instruct its other edge nodes which have not yet been targeted, to proactively modify their behavior, making the entire system more resilient.

B. Evaluation

To demonstrate the *viability* and *efficacy* of our proposed architecture, we implemented a Proof of Concept (PoC), deployed the PoC to an ECS, and conducted DoS attacks on the ECS.

For the PoC, we use physical hardware, including edge devices typically employed to manage injection molding machines and other computing nodes to construct an ECS. The ECS in our simulation setting consists of one cloud node, two fog nodes, and three edge nodes. At the device layer, we employed measurements at the software level, such as RAM usage, as well as external hardware sensors, like temperature and humidity sensors, to monitor the systems and the surrounding environment. The data collected by the sensors is transmitted via the Message Queue Telemetry Transport (MQTT) protocol. MQTT is a protocol specifically designed for efficient communication in IoT environments. The protocol operates on a server-client system in which the server, referred to as a broker, transmits updates to MQTT clients utilizing the publish-subscribe communication model. Although the protocol is efficient, it is susceptible to DoS attacks and thus serves as an adequate use case to evaluate the proposed architecture [6].

For implementing the self-adaptation and attack states, we use defense readiness condition (DEFCON) levels. The DEFCON levels, their triggers, and the enforced security policies

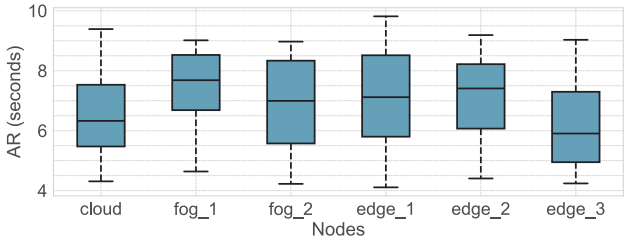

Fig. 3. Adaptability rate (AR) of the nodes in the evaluation

can be found in Table I. The determination of the thresholds was accomplished via iterative testing.

For measuring the *viability*, we randomly seed threshold exceeding data (c.f. Table I) into the system and measure the architecture's adaptability rate (AR). AR refers to the time to adapt. The AR commences when the initial node in the ECS initiates the adaptation process and concludes when all subsequent nodes in the ECS have also adapted.

To evaluate the *efficacy* of our approach, we conducted DoS attacks against the ECS. Since the goal of a DoS is to flood the network and ultimately make the system unavailable, we rely on the cumulative CPU usage of the whole ECS as a metric. This metric facilitates an assessment of both the attack's impact and the system's resilience against such disruptions. In the event of a successful DoS attack, CPU utilization peaks, rendering the device non-functional in our use-case. To contextualize the metric, we compare three scenarios: the cumulative CPU usage in normal operation without a DoS attack (*NO*), the cumulative CPU usage when under a DoS attack without our resilient architecture (*DA*), and finally, the cumulative CPU usage under DoS attack with our resilient architecture (*DAR*).

The repository, alongside our evaluation results, is publicly available on GitHub[1].

DEFCON	Triggers	Security Policies
5	CPU-usage $<$ 30% OR MQTT $<$ 80k msgs/min	*Regular Attack Monitoring*: Normal operating readiness
4	CPU-usage $<$ 40% OR MQTT $<$ 100k msgs/min	*Increased Attack Monitoring*: Promptly detect attacks
3	CPU-usage $<$ 50% OR MQTT $<$ 120k msgs/min	*Rate Limiting*: Limit sensor data forwarding
2	CPU-usage $<$ 60% OR MQTT $<$ 140k msgs/min	*Increase QoS of Critical Data*: Ensure crit. data transmission
1	CPU-usage $>$ 60% OR MQTT $>$ 140k msgs/min	*Localize (shut down MQTT)*: Prevent system outage

TABLE I
DEFCON LEVELS USED IN THE EVALUATION

C. Results

• **Viability:** We conducted three 20-minute simulation runs for every ECS node (6 nodes in total). Randomly every minute, a threshold exceeding data point was seeded for each node (i.e., 20 adaptations per run per node; 60 adaptations per node in total). To minimize the impact of variability and random anomalies, we aggregated the AR into box plots in Fig. 3.

Out of the 360 triggers, all were handled and responded to as expected by the deployed architecture. Every time one of

[1]https://github.com/jku-lit-scsl/ecs-demo

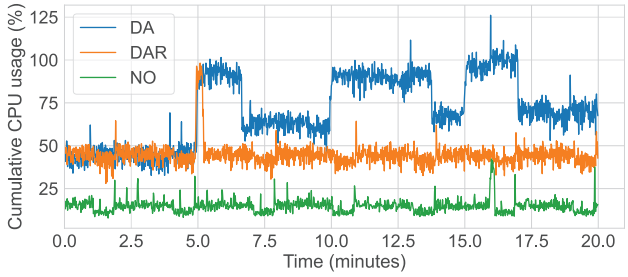

Fig. 4. Results of the efficacy evaluation

the nodes received a trigger, the node adapted the DEFCON level and enforced the respective policy accordingly. The AR of the nodes in Fig. 3 shows, that our architecture is viable to adapt timely in the event of a security incident.

• **Efficacy:** To measure the efficacy, we conducted three simulation runs, one for each scenario (*NO*, *DA*, and *DAR*), and monitored the CPU usage of every node once a second. For the DoS attack runs (*DA* and *DAR*), we started the DoS attack 5 minutes into the experiment and gradually attacked the other MQTT brokers in the ECS after 5 minutes.

Fig. 4 displays the cumulative CPU usage of the ECS over 20 minutes for the different scenarios. In our evaluation three MQTT brokers were installed, leading to an increase in CPU at minutes 5, 10, and 15 for *DA*. In contrast, using our resilient architecture (c.f. *DAR*), the ECS nodes adapted once the first attack was detected, resulting in a lower cumulative CPU usage compared to *DA* throughout the attack.

V. Discussion

The evaluation of our PoC has illuminated a balanced spectrum of advantages and challenges inherent to our proposed architecture. At the forefront of our findings is the system's dynamic adaptability and resilience, particularly showcased through the integration and independent functionality of the MAPE-K loops across different architectural levels. This design not only fosters collaboration between layers but also ensures that the architecture can maintain operational integrity in scenarios of network disruption. Such flexibility is critical, allowing the system to continue functioning via a local MAPE-K loop and seamlessly resume inter-node knowledge sharing once connectivity is restored.

The strategic implementation of DEFCON levels enhances the architecture's adaptability, allowing dynamic adjustment of defenses against current threats and future vulnerabilities. However, this adaptability also introduces potential vulnerabilities, as attackers could exploit the system's behavior by manipulating DEFCON criteria to induce specific responses, potentially compromising functionality or availability.

Compounding this concern is the challenge of defining precise thresholds for DEFCON levels. Establishing these parameters requires extensive testing to ascertain the network's and devices' tolerance to different loads and threat scenarios.

In our architecture, self-adaptation enabled by MAPE-K loops enhances system self-protection by dynamically adjusting to threats, highlighting the importance of self-adaptive processes in bolstering edge computing resilience against cyberattacks. Our proof of concept demonstrates the architecture's potential in cybersecurity adaptability and resilience, while also emphasizing the need for precise calibration and protection against exploitation to optimize defense mechanisms against a wide range of threats.

VI. Related Work

The literature survey conducted by Lara *et al.* [8] focuses on adaptive security systems that are based on MAPE-K. Although the discussed studies pertain to the same subject domain, their emphasis diverges from that of our study. They either prioritize other aspects of security, such as authorization, address predominantly domain-specific issues, such as those related to intelligent vehicles, or struggle to meet the SoS demands associated with ECS.

Moreover, existing literature [13] that addresses ECS tends to broach the subject from a broader perspective. Such discussions, while valuable, stop short of offering tangible solutions or architectural frameworks tailored to the intricate demands of ECS. The explicit treatment of DoS attacks; when these threats are considered, the responses typically lack a multi-layered architectural strategy for system defense. For instance, studies [16] might acknowledge the threat posed by DoS attacks but propose countermeasures that rely on a singular management component for security policy coordination. This approach contrasts with our proposed holistic model, which employs a novel, multi-level architectural strategy to bolster system security against such attacks.

In essence, our work introduces a nuanced methodological approach that leverages the hierarchical nature of ECS to offer a comprehensive, adaptive security architecture. This architecture is designed to effectively counteract a spectrum of cybersecurity threats, including DoS attacks, through a multi-tiered adaptive strategy that goes beyond the conventional methodologies discussed.

VII. Future Plans

To validate the effectiveness of our proposed enhancements, we will conduct real-world testing in diverse environments. This will not only provide insights into the practical challenges of deployment but also help fine-tune our architecture to meet the nuanced needs of different operational contexts.

We plan to develop more sophisticated anomaly detection algorithms to mitigate the risk of attackers exploiting our system's adaptive behavior. These algorithms will leverage machine learning and artificial intelligence to identify and respond to unusual patterns that could indicate manipulation attempts, enhancing the system's ability to distinguish between genuine threats and false triggers.

Looking further ahead, we envision developing a predictive cybersecurity model. This ambitious goal involves harnessing big data and predictive analytics to forecast potential security threats before they materialize, allowing preemptive defensive measures to be enacted and thus redefining the landscape of proactive cybersecurity.

VIII. Conclusion

In conclusion, our study presents a novel advancement in the domain of software architectures by integrating a multi-level MAPE-K loop approach into edge computing systems to bolster cyber-resilience. This study showcases the practical utility and effectiveness of our proposed framework in an industrial setting, highlighting its potential to enhance security in edge computing environments against complex cyber threats. Through initial testing and evaluation, we have shown that our approach enhances edge computing systems' adaptability, robustness, and security, paving the way for more resilient software architectures. Looking forward, we are committed to further refining our framework to address emerging threats and challenges, as well as to establish more robust defenses for edge computing architectures. Our work contributes to the broader discourse on cybersecurity, offering a scalable, dynamic solution that can be adapted across various industries and applications. As we continue to explore this promising field, we invite collaboration and further research to extend the boundaries of what can be achieved in securing edge computing infrastructures.

Acknowledgment

This work has been supported by the LIT Secure and Correct Systems Lab funded by the State of Upper Austria and the Linz Institute of Technology (LIT-2019-7-INC-316).

References

[1] Association for Computing Machinery. *ACM Digital Library*. 2023. URL: https://dl.acm.org/ (visited on 10/16/2023).

[2] Yuriy Brun et al. "Engineering Self-Adaptive Systems through Feedback Loops". In: *Software Engineering for Self-Adaptive Systems*. Vol. 5525. 2009, pp. 48–70.

[3] Keyan Cao et al. "An Overview on Edge Computing Research". In: *IEEE Access* 8 (2020), pp. 85714–85728.

[4] Kun Cao et al. "A Survey on Edge and Edge-Cloud Computing Assisted Cyber-Physical Systems". In: *IEEE Transactions on Industrial Informatics* 17.11 (2021), pp. 7806–7819.

[5] Rogério De Lemos et al. "Software Engineering for Self-Adaptive Systems: A Second Research Roadmap". In: *Software Engineering for Self-Adaptive Systems II*. Vol. 7475. 2013, pp. 1–32.

[6] Dan Dinculeană and Xiaochun Cheng. "Vulnerabilities and Limitations of MQTT Protocol Used between IoT Devices". In: *Applied Sciences* 9.5 (2019), p. 848.

[7] Jeffrey O. Kephart and David M. Chess. "The vision of autonomic computing". In: *Computer* 36.1 (2003), pp. 41–50.

[8] Evangelina Lara et al. "Adaptive Security Based on MAPE-K: A Survey". In: *Applied Decision-Making: Applications in Computer Sciences and Engineering*. 2019, pp. 157–183.

[9] Mohammed Laroui et al. "Edge and fog computing for IoT: A survey on current research activities & future directions". In: *Computer Communications* 180 (2021), pp. 210–231.

[10] Felix Lau et al. "Distributed denial of service attacks". In: *SMC 2000 Conference Proc.* Vol. 3. Oct. 2000, pp. 2275–2280.

[11] Microsoft. *Azure Stack Edge | Microsoft Azure*. 2024. URL: https://azure.microsoft.com/en-us/products/azure-stack/edge (visited on 01/08/2024).

[12] Microsoft Security Response Center. *Microsoft Response to Layer 7 Distributed Denial of Service (DDoS) Attacks | MSRC Blog | Microsoft Security Response Center*. 2023. URL: https://msrc.microsoft.com/blog/2023/06/microsoft-response-to-layer-7-distributed-denial-of-service-ddos-attacks/ (visited on 01/08/2024).

[13] Victor Casamayor Pujol et al. "Edge Intelligence—Research Opportunities for Distributed Computing Continuum Systems". In: *IEEE Internet Computing* 27.4 (2023), pp. 53–74.

[14] Qinglin Qi and Fei Tao. "A Smart Manufacturing Service System Based on Edge Computing, Fog Computing, and Cloud Computing". In: *IEEE Access* 7 (2019), pp. 86769–86777.

[15] Tie Qiu et al. "Edge Computing in Industrial Internet of Things: Architecture, Advances and Challenges". In: *IEEE Communications Surveys & Tutorials* 22.4 (2020), pp. 2462–2488.

[16] Michael Riegler, Johannes Sametinger, and Michael Vierhauser. "A Distributed MAPE-K Framework for Self-Protective IoT Devices". In: *18th Symposium on Software Engineering for Adaptive and Self-Managing Systems*. 2023, pp. 202–208.

[17] Mahadev Satyanarayanan. "The Emergence of Edge Computing". In: *Computer* 50.1 (2017), pp. 30–39.

[18] Statista. *Edge computing market revenue worldwide from 2019 to 2025*. 2023. URL: https://www.statista.com/statistics/1175706/worldwide-edge-computing-market-revenue/ (visited on 10/16/2023).

[19] Ryhan Uddin, Sathish A. P. Kumar, and Vinay Chamola. "Denial of service attacks in edge computing layers: Taxonomy, vulnerabilities, threats and solutions". In: *Ad Hoc Networks* 152 (2024), p. 103322.

[20] Arielle Waldman. *Microsoft: DDoS attacks caused M365, Azure disruptions | TechTarget*. 2023. URL: https://www.techtarget.com/searchsecurity/news/366542236/Microsoft-DDoS-attacks-caused-M365-Azure-disruptions (visited on 01/08/2024).

[21] Yinhao Xiao et al. "Edge Computing Security: State of the Art and Challenges". In: *Proc. of the IEEE* 107.8 (2019), pp. 1608–1631.

[22] Ashkan Yousefpour et al. "All one needs to know about fog computing and related edge computing paradigms: A complete survey". In: *Journal of Systems Architecture* 98 (2019), pp. 289–330.