Adaptation in Edge Computing: A review on design principles and research challenges

FATEMEH GOLPAYEGANI, School of Computer Science, University College Dublin, Ireland
NANXI CHEN*, Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, China and University of Chinese Academy of Sciences, China
NIMA AFRAZ, School of Computer Science, University College Dublin, Ireland
ERIC GYAMFI, School of Computer Science, University College Dublin, Ireland
ABDOLLAH MALEKJAFARIAN, School of Civil Engineering, University College Dublin, Ireland
DOMINIK SCHÄFER, Syntax Systems GmbH & Co., Germany
CHRISTIAN KRUPITZER, Department of Food Informatics and Computational Science Lab, University of Hohenheim, Germany

Abstract Edge Computing places the computational services and resources closer to the user proximity, to reduce latency, and ensure the quality of service and experience. Low latency, context awareness, and mobility support are the major contributors to edge-enabled smart systems. Such systems require handling new situations and change on the fly and ensuring the quality of service while only having access to constrained computation and communication resources and operating in mobile, dynamic, and ever-changing environments. Hence, adaptation and self-organisation are crucial for such systems to maintain their performance, and operability while accommodating new changes in their environment.

This paper reviews the current literature in the field of adaptive Edge Computing systems. We use a widely accepted taxonomy, which describes the important aspects of adaptive behaviour implementation in computing systems. This taxonomy discusses aspects such as adaptation reasons, the various levels an adaptation strategy can be implemented, the time of reaction to a change, categories of adaptation technique, and control of the adaptive behaviour. In this paper, we discuss how these aspects are addressed in the literature, and identify the open research challenges and future direction in adaptive Edge Computing systems.

The results of our analysis show that most of the identified approaches target adaptation at the application level, and only a few focus on middleware, communication infrastructure, and context. Adaptations that are required to address the changes in the context, changes caused by users or in the system itself are also less explored. Furthermore, most of the literature has opted for reactive adaptation, although proactive adaptation is essential to maintain the edge computing systems' performance and interoperability by anticipating the required adaptations on the fly. Additionally, most approaches apply a centralised adaptation control, which does not perfectly fit the mostly decentralised/distributed Edge Computing settings.

*Corresponding author.

Authors' addresses: Fatemeh Golpayegani, fatemeh.golpayegani@ucd.ie, School of Computer Science, University College Dublin, Dublin, Ireland; Nanxi Chen, Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai, China and University of Chinese Academy of Sciences, Beijing, China, nanxi.chen@mail.sim.ac.cn; Nima Afraz, School of Computer Science, University College Dublin, Dublin, Ireland; Eric Gyamfi, School of Computer Science, University College Dublin, Dublin, Ireland; Abdollah Malekjafarian, School of Civil Engineering, University College Dublin, Dublin, Ireland; Dominik Schäfer, Syntax Systems GmbH & Co., Weinheim, Germany; Christian Krupitzer, Department of Food Informatics and Computational Science Lab, University of Hohenheim, Stuttgart, Germany.

ACM 1556-4703/2024/5-ART
https://doi.org/10.1145/3664200

CCS Concepts: • **Computer systems organization** → **Embedded and cyber-physical systems**; Real-time system architecture; • **Human-centered computing** → *Ubiquitous and mobile computing theory, concepts and paradigms*; • **Computing methodologies** → *Mobile agents*; • **Networks** → *Network reliability*; *Network architectures*.

Additional Key Words and Phrases: adaptation, edge computing, MAPE-loop, edge-enabled

1 INTRODUCTION

With the rapid development of advanced digital technologies and an increasing number of Internet of Things (IoTs) devices, cities are becoming more interested in the new generation of urban planning tools and techniques to improve their services [13]. The concept of edge-enabled IoT use-cases are closer to reality by employing different electronic devices, sensors, and actuators in various applications such as smart homes, surveillance, environmental pollution, vehicular traffic and transportation, healthcare, weather prediction, and water systems [6]. Such applications require powerful computational resources to process and analyse the sheer amount of data collected by various types of sensors to provide insights that serve as a basis for making decisions.

Cloud computing offers ubiquitous, convenient, and on-demand computing services by allowing resource-constrained nodes of a network (e.g., end devices) to access a shared pool of computing and storage resources [86]. Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS), are the main categories of cloud computing services offered to resource-constrained nodes. Although such services are advantageous in several ways they have multiple limitations when they are required to support low latency, context awareness, mobility, data safety, and privacy requirements [121]. Such requirements are the main reasons for the paradigm shift towards edge computing, particularly in bandwidth-heavy applications with strict tolerable delay requirements such as autonomous vehicles, surveillance systems, and remote surgery. Edge computing provides additional computing resources at the edge of the network close to the employed sensors and actuators in the city reducing the latency, and ensuring the quality of service and user experience. For example, in intelligent transport systems, autonomous vehicles can benefit from low latency communication provided by mobile edge computing (located nearby) to coordinate their behaviour and make decisions in real-time [110] compared to a cloud computing alternative with computing nodes in public cloud perhaps located in another continent.

Many IoT services depend on the collaboration, coordination, and cooperation of multiple devices that may dynamically move in an environment or rely on unstable energy sources while interacting using wireless connections. In such an uncertain and highly dynamic environment, unforeseen events such as failure or unavailability of devices can impact the whole system. Therefore, it is vital for the rest of the nodes to be able to adapt to the changes caused by such unforeseen events and self-organise themselves to stay functional.

For example, in an autonomous driving scenario, a smooth flow of vehicles partially depends on the communication between the vehicle (V2V) and the roadside units (V2I). However, when a rode-side unit (e.g., edge node) fails to respond, the vehicles must be able to coordinate their behaviour (e.g., decreasing their speed) while finding a substitute node to operate as normal. To show such behaviour, the vehicles must be able to identify such changes in their environment (e.g., failure of a node), coordinate and adapt their behaviour while affecting the traffic flow as little as possible, and self-organise themselves by connecting to substitute nodes. However, events that might trigger an adaptation process in the current edge system are not limited to the failure of a node or accidents. Such events include social events which will require activation of a broader range of smart devices and entities for security, connectivity, or logistics.

Therefore, self-adaptability and self-organisation become increasingly important requirements to ensure edge computing systems' performance especially in resource-constrained, dynamic, and open environments while interacting with heterogeneous entities [22, 138]. Self-adaptive systems can address these requirements, by enabling their components (e.g., nodes and devices) to change their behaviour at run-time as a response to changes in their environment or in the system itself [30, 69]. Those systems are able to work in dynamic and

uncertain environments as they adjust to new, unknown situations through adaptation. They integrate, *self-* properties* [107], which can be distinguished into basic properties—*self-awareness* [57] and *context-awareness* [112] to retrieve information about the system resources as well as the surrounding environment and major levels properties, such as self-configuration, self-healing, self-optimisation, and self-protection that use the awareness to offer autonomic system functionality [63]. Multiple self-* properties are integral parts of many edge computing systems by design in the literature, however, there is not a comprehensive study to show which self-* aspects are required and at which level they should be integrated with the system from a self-adaptive and self-organising systems perspective.

This paper will have the following contributions:

- This paper reviews various edge-enabled IoT applications and categorises them into smart infrastructure and smart services. It then discusses why adaptation is essential in these applications.
- Reviews the edge computing systems in this context and analyse their design and implementation compatibility with the existing design guidelines and principles in self-* systems, using a widely accepted taxonomy [69].
- Finally, an in-depth discussion on adaptive edge computing systems' current research challenges and future direction is proposed.

The rest of this paper is organised as follows. Section 2 compares this paper with other review works from the fields of adaptive systems and edge computing. Section 3 presents an overview of fundamental concepts including edge computing, its applications and implementation challenges, as well as self-adaptive systems. Section 4 describes the methodology for the literature review (selection and analysis methods). Section 4.2.1 reviews self-adaptive systems definitions and design principles and offers a taxonomy of implementing self-adaptive systems. We apply the taxonomy to an autonomous driving example. The taxonomy is used to review Edge-enabled IoT-based applications and lists their level of adaptive behaviour in Section 5. Finally, Section 6 discusses the results of the survey, drives open issues and challenges, and presents threats to validity, before Section 7 concludes this paper.

2 RELATED STUDIES

This survey connects the concepts of adaptation and edge computing which seems to be a natural combination given the uncertainty in the environment that is present in both system domains. To the best of our knowledge, there is no study, which discusses the aspect of adaptation within edge computing systems. In this section, we provide an overview of related studies from the area of adaptive systems as well as Edge Computing.

There are many research literature reviews and summaries in the field of adaptive systems, which focus on either the general topic or special system aspects. Salehie and Tahvildari [107] broach the issue of self-adaptive systems as such, provide a taxonomy and describe possible realizations of adaptation actions. The work of Weyns [137] provides an overview of adaptive systems with a focus on software engineering. They summarize the findings in the fields of task automation, architecture-based adaptation, runtime models, goal-driven adaptation, uncertainty management, and control-based adaptation. The authors conclude that control theory can act as a theoretical foundation for adaptations and decision-making. The surveys of Patikirikorala et al. [97] and Shevtsov et al. [117] takes a closer look at this issue. Further introduction material and reference work in the area of adaptive systems is provided by Krupitzer et al. [69], Macias-Escriva et al. [82], or more recently by Wong et al. [140]. More closely to the topic of this paper, Saputri et al. [109] presented an overview of the application of machine learning in self-adaptive systems to assist self-adaptation, but also to analyze the requirement for adaptation. As an outcome of the 2018 GI-Dagstuhl Seminar "Software Engineering for Intelligent and Autonomous Systems", D'Angelo et al. [37] discuss the current state of the art for learning in complex self-adaptive systems. Another survey categorizes design patterns supporting the implementation of adaptive systems [70]. Those design patterns

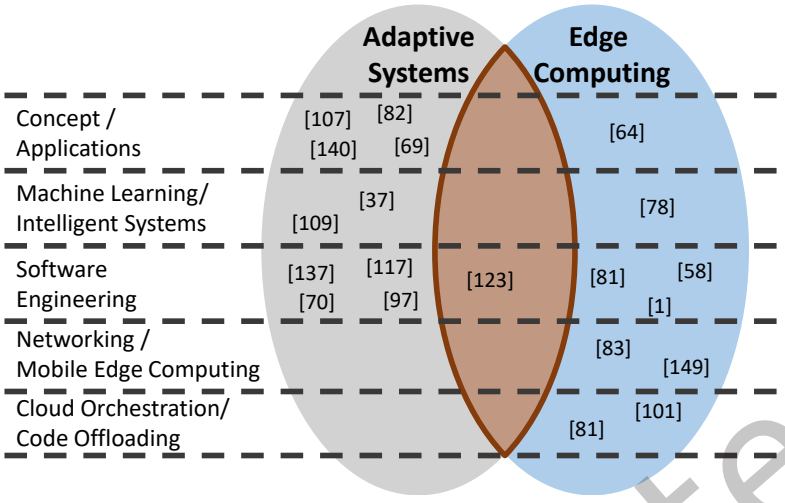

Fig. 1. The related work can be classified into the areas of edge computing and adaptive systems and the cross-cutting topics concept, learning, software engineering, networking, and cloud orchestration. Our work targets the intersection marked in red between adaptive systems and edge computing across all topics.

can be also implemented in the context of adaptive Edge Computing systems. However, none of these studies in the area of (self-)adaptive systems target explicitly Edge Computing systems.

Regarding Edge Computing systems, several surveys exist; each focusing on other aspects. Ren et al. compared emerging computing paradigms (including Transparent Computing, Mobile Edge Computing, Fog Computing, and Cloudlet) from the perspective of hierarchical edge-cloud orchestration [101]. Khan et al. presented recent advancements in Edge computing highlighting the core applications[64]. Yu et al. categorise edge computing applications based on network latency, bandwidth occupation, energy consumption, and overhead as well as propose a framework for security evaluation of IoT networks with edge computing [149]. Another survey provides a comprehensive overview of Edge Computing systems highlighting energy efficiency and deep learning optimisation of edge computing systems [78]. Some surveys target the analysis of architectures for mobile Edge Computing (e.g., [81], [58], and [1]). Mao et al. provide a comprehensive survey of the state-of-the-art MEC research with a focus on joint radio-and-computational resource management, including MEC system deployment, cache-enabled MEC, mobility management for MEC, green MEC, as well as privacy-aware MEC [83]. Other surveys focus on the aspect of code offloading (e.g., [81]).

However, none of those surveys integrate the aspect of adaptation. As shown in Figure 1, only the work of Taherizadeh et al. [123] provides a discussion of the state-of-the-art of monitoring adaptive components in Edge Computing systems, i.e., combines the topics of edge computing and adaptation. However, their work focuses the monitoring only. In contrast, our work targets the whole Monitor-Analyse-Plan-Execute (MAPE) functionality and spans various topics as well as applications. This is represented by the whole intersection marked in red in Figure 1. To the best of the authors' knowledge, this is the first comprehensive survey combining all those aspects of edge computing and adaptive systems.

3 FUNDAMENTAL CONCEPTS

This section briefly reviews the fundamental concepts on which this paper is focused.

3.1 Edge computing

In the past decade, the ability to process data and compute remotely (due to the rapid growth of network capacity) created a new computing approach "cloud computing" that allowed the users to reduce their up-front computing infrastructure cost while in effect delegating the operation, maintenance and monitoring of their data centres to public cloud providers. However, such a level of centralisation introduces challenges such as a single point of failure and larger latency due to the required communcation of data and computed results between the user and the cloud. While this still met the requirements of many applications, in applications where the users (e.g., a Traffic Authority requiring its IoT road infrastructure across the city to communicate and coordinate a latency-sensitive action) require a regional low-latency computing service, the latency exposed by the communication to the cloud servers becomes a bottleneck.

Edge computing is a distributed computing framework whose infrastructures reside at the edge of the networks. The edge is a type of networking device that connects different networks and, often can control the underlying network infrastructure equipment. For example, a router serving as a gateway that connects an internal local area network (LAN) to the Internet can be considered as a network edge. The edge computing infrastructure can host applications and allow them to be placed closer to the sources of data and users. With such proximity, edge computing can reduce response times, provide faster insights, and support better bandwidth availability toward the cloud server. Specifically, it enhances the traditional cloud-to-thing systems with a distributed edge network that consists of shared edge devices (e.g., small edge servers, smart gateways, or 5G base stations [34, 92]) and allows services to be deployed on them.

The most common research areas in edge computing communities over the past few years include cloudlet, Mobile Edge Computing (MEC), multi-access edge computing, fog computing and multi-tier computing. All these areas aim to place computation capabilities and intelligence closer to the data source, each focusing on a particular layer of the network. **Cloudlet** [111] is a small-scale data center that provides cloud computing power and can be considered as a simplified implementation of cloud services. A cloudlet data center is a trusted, resource-rich computer or cluster of computers that can offload users' computing tasks but supports limited tasks and fewer users within a close geographical scope compared to the Cloud. The initial concept of cloudlet does not consider it as a part of the network infrastructure, However, recent research has been using it to enable mobile edge computing [102]. **Mobile Edge Computing (MEC)** provides a better application performance to cellular network users by enabling cloud computing power and IT services at the edge of the cellular network to reduce latency. Related standards about MEC are defined by MEC International Standard Group (MEC ISG) in ETSI. In 2017, MEC ISG extended the definition of MEC to Multi-access Edge Computing by adding the notion of heterogeneous networks into the original MEC framework. This makes the access network of MEC support cellular users as well as Wi-Fi and broadband user. **Fog computing** [29] shares a very similar concept to the extended MEC, with a particular focus on providing communication interfaces and building a unified platform over heterogeneous devices (e.g., smart routers, network storage devices, small servers, etc.) in the computing network. MEC technology, introduced by the ETSI ISG on MEC [41], refers to cloud-computing capabilities deployed directly at cellular base stations, allowing computing to occur at the edge of the mobile network [44]. In contrast, fog computing supports compute, storage, and networking services between edge devices and cloud data centers [25], and it refers to a hierarchical architecture that extends cloud capabilities closer to the network edge, often involving intermediate nodes with significant computational resources.

Furthermore, fog-to-cloud continuum [28] is an extension of fog computing, which emphasises the coordination between fog devices and the cloud to support cross-domain and large-scale applications. **Multi-tier computing** [54, 146] extends fog computing and proposes a hierarchical network structure that allows services to be deployed in the most suitable place in the network according to the user requirements. It focused on throughput, analytics, and intermittent connectivity for heterogeneous devices besides the general latency-related issue.

This paper will focus on enabling adaptation of edge-related technologies in the context of IoT applications. Here, we refer to Industrial Internet Consortium's (IIC) definition and generalise the edge computing model as a fully distributed paradigm that lies between physical things in the real world as monitored and controlled by IoT devices (sensors and actuators), via layers of edge nodes connected to the data centre [32].

3.2 Edge computing and smart applications

A wide range of applications has embraced edge computing to meet their computing requirements and move them closer to their users. While the main motivation behind migrating from cloud-centric deployments to Edge-based or hybrid deployments is to perform the computational tasks closer to the users, however, various applications have additional reasons for adopting edge computing. These include among others service locality, data proximity, privacy preservation, low latency, and communication infrastructure constraints. Multiaccess edge computing (MEC) technology optimizes resource allocation and offloading for IoT devices, resulting in innovative edge computing platforms. These platforms use adaptive resource allocation and computation offloading to maintain load balancing [134]. In this section, we divide such applications into applications related to smart infrastructure and smart services that have adopted edge computing while their reason for adoption might vary (see Table 1, which includes some of such requirements for the applications reviewed in this paper). IoT devices and AI-powered applications are driving the need for edge computing in 5G wireless communications, which overcomes cloud-RAN latency limitations [91]. Examples of such applications include intelligent transport systems, autonomous vehicles, smart energy grids, smart homes, structural health monitoring, water and waste management, and food supply chain management [10, 50, 93]. Dealing with data explosion and network traffic, a need for more intelligent computation paradigms and sustainable energy consumption are other motivations for many applications to choose edge computing paradigms [131]. The IoT applications have increased the massive amount of data produced at the edge of the network. This data often includes sensitive user information and is currently stored in data centers, causing data owners to relinquish control. Edge computing offers a solution by processing and analyzing data at the edge of the network. However, data privacy remains a challenge [142]. Due to the massive amount of data produced by the IoT, edge computing requires precise resource management to efficiently allocate resources for IoT devices and ensure high-quality service [17, 42]. The IoT with an edge-assisted system must also be equipped to handle emergency situations for high-priority clients [143].

3.2.1 Smart Infrastructure. To enable any smart services in the context of IoT use cases, we require the system to be able to sense, analyse, communicate and actuate. Edge computing has become a particularly attractive paradigm for many IoT-centric applications. Particularly, it has been used in **intelligent transport systems** and **industrial automation** applications through real-time video analytics, connected traffic counters, NFC, GPS, and smartphones. Recently, AI and edge computing have revolutionized live-dependent systems like vehicular technology, emergency response systems, and communication technologies by enforcing low-latency communication channels at the edge of the network [153]. For example, pedestrians and traffic counting, road usage monitoring, connected vehicle technology, smart parking, and real-time traffic routing and management are amongst many such applications offering a better representation of the traffic and data-driven models satiable for real-time decision-making [10]. In particular, video processing has become an important component in intelligent transport systems, autonomous driving, and other applications such as **surveillance systems and virtual/augmented reality (VR/AR)[150],** which require privacy, low latency computation, and location-awareness for efficient and effective functioning. With edge computing, video processing systems operate more efficiently by reducing huge network communication overhead and providing real-time and accurate video analytic solutions.

Structural health monitoring normally refers to a process in which an engineering system (such as bridges, or buildings) is being observed using sampled response measurements to monitor its structural condition [85].

Table 1. This table summarises the presented edge computing applications, the corresponding application domains and their requirements which might be supported by adaptation.

Application	Application Domain	Requirements
Intelligent Transport Systems	Smart Infrastructure	• locality of services, proximity of data, privacy, ultra-low latency, geo-distributed services, location-awareness
Industrial automation	Smart Infrastructure	• locality of services, proximity of data, privacy, ultra-low latency, location-awareness
Surveillance systems, VR/AR	Smart Healthcare/Infrastructure	• high bandwidth, ultra-low latency, high-speed processing of information, reactiveness
Vehicular Traffic/Transportation Monitoring	Smart Infrastructure	• privacy, high data volume, cloud integration
Healthcare	Smart healthcare	• heterogeneous resources, privacy, data protection, real-time analysis, varying network conditions
Smart Home	Smart Services	• privacy, heterogeneous resources, cloud integration
Smart Energy Grid	Smart Infrastructure	• geo-distributed services, low latency, stable communication, data protection, privacy, spatial data
Structural Health Monitoring	Smart Infrastructure	• low latency, heterogeneous communication infrastructures, high data volume
Water management	Water, food, and waste management	• geo-distributed services, spatial data
Waste management	Water, food, and waste management	• geo-distributed services, data protection, privacy, spatial data
Emergency response management	Smart Services	• responsiveness, privacy, geo-distributed service, heterogeneous communication infrastructures, low latency
Food supply chain	Water, food, and waste management	• heterogeneous resources & communication infrastructures, communication delays, fragmented data

The process normally includes the installation of several sensors on the structure which are connected to a cloud using wireless systems [18]. For Civil Engineering structures, the target structure is usually enormous in size. The conventional approach is to collect all the data and transmit them to a cloud where the decisions are being made. However, this approach could be extremely inefficient, when there are many sensors on the structure and the transmission bandwidth is limited [141]. In some cases, instead of using sensors or manual inspections, swarms of autonomous inspection robots are used which could be examples of edge devices. In this case, optimising the damage detection algorithms used at the edges is critical as these devices may have limited memory and computational resources. A few studies have developed novel solutions based on the concept of edge computing to tackle these challenges. In these methods, edge devices are deployed that have the capability to do the signal processing and decision-making without the support of the cloud server [141]. In those scenarios, acceleration sensors measure the vibration. Such sensors generate large volumes of data, and for accurate predictions, sampling data at a high rate is necessary. This makes the cloud-only solution very inefficient. Fog and edge computing are becoming popular in this domain due to the need for real-time data processing in low-latency applications like structural health monitoring and surveillance. These solutions bring resources closer to the sensors and actuators used in monitoring, resulting in efficient, high-performance data processing that outperforms traditional cloud computing [84].

Smart energy grid includes the technologies to incorporate real-time monitoring of electricity usage to improve its control over the generation, and distribution. Demand-response mechanisms optimise energy consumption patterns by shifting non-critical demand to times of low electricity demand [48]. Distributed learning, negotiation, collaboration, and optimisation methods are commonly used to coordinate consumers' behaviour. These methods assume that the entities in the system are equipped with sensors and are able to communicate,

process data, and make decisions on the fly [108]. Edge computing is an enabler technology that realises the aforementioned objectives through its privacy-preserving low-latency computational power.

3.2.2 Smart Services. **Emergency response management** services, minimise the response time to reduce the damage that significantly increases the injured survival [49]. Understanding the severity of such emergencies, locating and routing the essential services impact the response teams' efficiency and resource management. Transportation network and traffic management are important issues when emergency services should be delivered using already jammed roads. Various intelligent techniques such as adaptive control and monitoring, learning, optimisation use vehicle-to-vehicle and vehicle-to-infrastructure technologies to improve the response time by coordinating the vehicles' behaviour when an emergency happens. In this application, low-latency computation and communication are key requirements.

Smart Industry; The introduction of Industry 4.0 marks a new era in the manufacturing industry, combining modern AI and edge computing with existing technology to increase productivity. Edge computing is crucial for achieving high-performance state estimation in industrial IoT systems [80]. It enables task offloading and data processing at the edge of the network, which are essential for optimal performance. As a result, traditional manufacturing industries are gradually being replaced with intelligent manufacturing technology supported by edge computing [52]. This shift has led to the creation of strategies such as Europe 2020 [40], Industry 4.0 [73], and China Manufacturing 2025 [154]. The transformation towards intelligent manufacturing will have a profound and lasting effect on the future of manufacturing worldwide. To achieve these advancements, it is crucial to establish ultra low-latency and adaptable network technologies capable of processing industry data at the edge of the network. Additionally, the implementation of smart factories should consider current manufacturing requirements.

Smart home services allow homeowners to manage their devices in a more efficient way to save energy, improve security, and increase conveniences[128]. Edge devices have been used in smart home applications to enable AI services for customisation of the device management, such as light switch [144]. With edge devices' physical isolation from the data centre, they are also used to address privacy issues in smart home applications. Vigilia [128] implements a close network for smart home systems through edge computing. Such restricted network access of devices reduces security risks and makes the users' private data only used to improve their services.

Smart healthcare services are enabled both at individual and community levels using edge computing paradigm, as low latency computation is required to analyse and act upon data gathered from medical devices, wearable IoT devices and other relevant sensors. Remote health monitoring, fitness tracking, processing-patient generated data through wearable devices, and elderly care are a few examples of applications where data is required to be processed at the proximity of its resources to help low latency decision-making and actuation. Such applications will reduce cost, increase the quality of life and improve users' experience [60, 96]. It is now possible to monitor patients' symptoms in real-time using IoT and edge computing technology, allowing for the quick detection and reporting of sudden and dangerous changes to healthcare providers [129].

3.2.3 Water, food, and waste management. Services use IoT for the optimum management of resources such as water [132], more efficient food supply chain and waste management. Several researchers have integrated edge computing integrated architecture with the concept of blockchain to facilitate water saving for households in urban areas [125, 145]. In addition, several researchers have used edge computing for smart and sustainable agriculture and breeding to address challenges regarding sustainable food security [7]. The concept of cloud computing for the purpose of agricultural water management has shown to be an inefficient solution when it is integrated with IoT systems. The **food supply chain** might also heavily benefit from edge computing in combination with intelligent and adaptive systems especially after the COVID-19 pandemic [45]. There, seamless control and traceability of food items have to be guaranteed to ensure food safety and avoid food fraud (e.g.,

labelling cow milk as goat milk). This can be supported by intelligent sensor technology; however, it requires machine learning approaches to analyse in real-time, e.g., the chemical composition of food. However, energy is an important aspect here as most operations are done on the field, on the farm, or during transportation. Waste management in urban areas and industrial zones is a growing challenge. Several studies have shown the use of IoT technologies and edge computing capabilities for developing smart waste management systems which can be beneficial in keeping track of disposal violations and recycling [3]. For example, cameras that are connected to edge devices can be attached to waste and recycle bins where the images of disposed items or materials are captured. Therefore the images can be processed locally at the edge of a network to detect if there are violations at the time of disposal [2].

3.3 5G, a Medium of Communication for Edge-enabled IoT Adaptive Systems

The advancement of edge computing has been recognized as the primary facilitator for many IoT applications. To offer a diverse range of innovative services, low latency networks must be established through collaboration between MEC, IoT, and 5G networks [5]. Edge computing is an extensively distributed computing system that effectively interacts with the underlying 5G network to provide a variety of services to multiple IoT devices [51]. This collaboration leads to the formation of a completely new ecosystem and value chain. Moreover, MEC serves as a key technology that enables the seamless integration of IoT with 5G wireless systems, providing valuable insights. 5G networks offer ultra-low latency connectivity [33]. Some of the core technologies that edge computing and 5G networks offer are Network Function Virtualization (NFV), software-defined networks (SDN), Network Slicing, and information-centric networking (ICN). 5G networks increase the flexibility of IoT and edge computing applications by providing high throughput, low latency, and adaptability [88]. 5G communication also enhances smart network functionality deployment and migration capabilities [120].

3.4 Why do we need adaptations?

In addition to domain-specific requirements, several challenges exist when implementing real-world applications that operate in unpredictabile environments, with heterogeneous software and hardware, non-stationary devices, and the ever-changing domain-specific requirements that cannot be predicted at design time. Some of these challenges are detailed as follows.

- **Environment characteristics:** This means that such systems are operating in dynamic, partially observable open environments, where complete domain knowledge is not accessible, and devices can join and leave the system freely.
- **Heterogeneity of nodes:** Systems and entities within require communicating, coordinating, and cooperating with heterogeneous devices, that their appearance in the system might have not been foreseen. This includes devices and nodes as well as heterogeneous software and hardware.
- **Mobility:** Such systems must be able to cope with the unpredictable mobility pattern of devices as they may join or leave the system frequently and unannounced.
- **Dynamic availability of resources:** Such systems should be able to respond to the demand by the available strictly limited computational and communication resources. The mobility of devices that can be both task generators and/or computational resource providers can lead to unpredictable resource availability.
- **Software change:** handling changes in the internal software design of a device caused for example by a software update is another challenge.

The common phenomenon in all these challenges is the "occurrence of a change" in the system. To cope with such changes, the system needs to understand why a particular change has happened, and when and how to react. Therefore, the system requires "intelligence" as well as "observer" units. The observer needs to

collect data about the system and its environment, which can be challenging, especially in distributed systems settings. The intelligence needs to reason about that information. The objective is to detect any events that might influence the system performance as early as possible, in the best case, even to forecast them and act proactively. However, this is a complex task because of a high amount of potentially contradicting information and uncertainty in the dynamic environment. Especially as the domains of applications that we described in the preceding Sections 3.1, 3.2, and 3.3 have different requirements regarding the control period and reaction times. There are applications that have strong real-time requirements, such as intelligent traffic applications (like self-driving vehicles) or cyber-physical production systems in Industry 4.0. Other applications have more relaxed requirements, such as smart homes. The research domain of self-adaptive systems tries to tackle those above-mentioned challenges regarding the dynamics in those systems. In the following section, we shortly introduce such systems.

3.5 Self-adaptive Systems

According to Salehie and Tahvildari [107], *self-* properties* can be ordered hierarchically into three levels: (i) primitive level, (ii) major level, and (iii) general level. On the primitive level, *self-awareness* (e.g., [57]) and *context-awareness* [112] are the basic functionality to retrieve information about the system resources as well as the surrounding environment[1]. To achieve context awareness, the system has to use sensors to collect information about its environment and reason about that information [69].

At the major level, self-* or self-management properties including self-configuration, self-healing, self-optimisation, and self-protection (a.k.a. self-CHOP), use the awareness to offer autonomic system functionality [63]. *Self-configuration* implies that the system is able to configure itself instead of requiring an error-prone manual installation and configuration [72]. *Self-healing* contains detecting and patching issues without human support. This is linked to self-diagnosing and self-repairing. *Self-protection* refers to automatic defence against malicious attacks and actions [107]. *Self-optimisation* is mainly important during the operation phase of a system, where it monitors its own performance in order to adjust tuning parameters to changes in the environment.

At the general level, systems that fulfil the self-adaptiveness property aggregate the functionality of the major level and offer autonomous reactions to changes in the system and the environment. Systems that provide those functionalities are called self-adaptive (software) systems. The participants of the first Dagstuhl seminar on *Software Engineering for Self-adaptive Systems* agreed on the following definition:

> *[Self-adaptive systems] are able to adjust their behaviour in response to their perception of the environment and the system itself.* [30, p. 1]

The "self" indicates that the system decides how to adapt without or only with minimal users' control [16, p. 49] From an architectural point of view, a self-adaptive system is composed of two parts: a managing system as well as a managed system [30]. The managed system is composed of a set of resources of any kind of software and hardware, e.g., servers, laptops, smartphones, robots, or unmanned vehicles. The managing system, also called adaptation logic, is a set of software modules that observes the environment and analyses the need for adaptation, plans such adaptations as well as controls the execution of the adaptation. Various control structures exist in literature, such as the MAPE cycle [63], the sense-plan-act control [67], the autonomic control loop [39], the observer/controller architecture [126], or AIOps [12] [130]. Figure 2 shows the architectural model of a self-adaptive system. In this paper, we particularly focus on how to integrate the functionality of the adaptation logic in various applications supported by edge computing.

[1]Note: Whereas self-awareness, as described in this paper, refers to awareness of the system's self, *Self-aware Computing* includes — according to the outcome of a Dagstuhl seminar — the reasoning on adaptation [66]

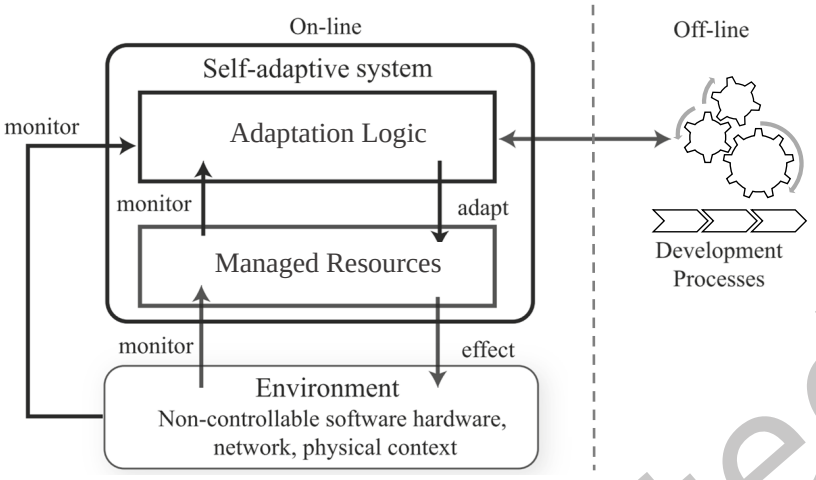

Fig. 2. System model of a self-adaptive system from an architectural point of view (adapted version: in contrast to [113], the adaptation effects the environment).

4 METHODOLOGY

In this survey, we followed a systematic mapping studies approach according to Petersen et al. [98]. This approach helps to systematically analyze a research topic in order to provide an overview of the targeted area and allows the identification of the quantity and type of available research. We decided against a structured literature review (e.g., following Webster and Watson [135]) which usually integrates a very strict keyword-based search for two reasons. First, edge computing is a rather young research topic and, consequently, the understanding of the term is something still vague. Accordingly, it seems critical to just use the term "edge computing" for the search. However, using additional terms in relation to edge computing like fog computing might impact the usefulness of the identified results. Second, adapting computing might have many different facets, too: Hence, also just using the term "adaptive system" seems not appropriate. For the system mapping study, we followed a footnote chasing respectively backward chaining approach [15], meaning that we explicitly followed the references of identified papers. This helps to increase the search space, as we do not have the strict keyword-based approach but rather a venue-focused approach. We defined exclusion and inclusion criteria and filtered the articles based on their titles and abstracts. After identifying the set of relevant papers, bibliography data have been extracted and we applied the taxonomy of self-adaptation from [69] (see Section 4.2.1) to structure the extracted information. In the following, we describe these steps.

4.1 Selection Method

The aim of our study is to analyse the presence of principles from (self-)adaptive systems within edge computing systems. Due to the ambiguity and variants of the terms "edge computing" and "adaptive systems", we avoided a fully open search term-based literature search as this would result in misleading results. Hence, we used a focused, systematic mapping study [98]. In the following, we describe the selection method for building our set of relevant papers. The process is similar to the one used in [37], in which the authors studied the application of learning in collective adaptive systems.

(1) Identification of Relevant Sources. To identify relevant conferences and journals, we performed some initial searches and consolidated experts in the field of study. The selected venues include conferences such as SEC,

IPSN, ICAC, SASOW, SEAMS, ICC, GLOBECOM, and journals such as IEEE Communications Magazine, IEEE Internet of Things Journal, ACM Transactions on Autonomous and Adaptive Systems, IEEE Communications Surveys & Tutorials, Software Engineering for Self-Adaptive Systems, IEEE Transactions on Industrial Informatics, IEEE Transactions on Wireless Communications, IEEE Internet Computing, IEEE Transactions on Services Computing, IEEE Transactions on Vehicular Technology, and IEEE Access.

Accordingly, we used mainly the databases DBLP, IEEExplore, ACM DL, ScienceDirect, and SpringerLink to access the publications of the venues. We analysed papers from January 2016 to December 2021, as before the term "edge computing" was not really present in literature. We only analysed full research papers or articles for this survey and excluded texts such as editorials, demonstrations, short papers, workshop contributions, letters, or posters because those often report work-in-progress. In total, we analysed 8,971 publications from January 2016 to Augest 2023 from which we included 37 after the application of the inclusion and exclusion criteria.

(2) Definition of Coarse-Grained Exclusion and Inclusion Criteria. We screened the identified edge-computing papers for relevant keywords related to adaptive systems. As it is hard to capture the term adaptive systems with all its facets, we used several prefixes because of its vastness. We used amongst others *adapt-*, *self-*, *aware*, *transform-* and *autonom-* as prefixes. However, even if we detected no match with this second group of search targets, we performed a high-level screening of the paper in case we observed a high match with the specified terms. We excluded papers without an explicit relation to adaptation.

(3) Study Selection Procedure. Based on the mentioned criteria, we performed rough filtering of the articles considering their titles, abstracts, and linked keywords, if applicable. For each paper fitting the scope of the inclusion criteria, we conducted a full-text analysis. Three reviewers conducted the paper selection and analysed the sampled studies to confirm their relevance. In case of doubt, advice from the other co-authors was taken into account.

4.2 Analysis Method

For each remaining article, we extracted the properties of the adaptation behaviour. We used the taxonomy of self-adaptation from [69] to structure the information. Accordingly, we extracted information regarding (i) the type of adaptation control mechanism (w.r.t. the approach, adaptation decision criteria, and degree of decentralization), (ii) the reason to adapt, (iii) the applied technique to adjust the system, (iv) the timely scope for the adaptation, as well as (v) the elements of the system where adjustments are applied. Section 4.2.1 describes the applied taxonomy in detail.

This information is used to answer our research question regarding the level of adaptation in edge computing systems.

4.2.1 Self-Adaptive Taxonomy. In this paper, we use a uniform self-adaptation taxonomy introduced in [69], which unlike the other works in the literature [11, 53, 103, 107] provides an integrated view of self-adaptation regardless of domain-specific implementation or system consideration. This section presents a five-dimension taxonomy on self-adaptation (see Figure 3) we use for structuring the literature in this review.

The **Time** dimension describes *when* an adaptation is executed in relation to an event that triggers an adaption. Adaption can be *proactive* or *reactive*. Even though a proactive adaptation is desirable from the users' point of view, reactive approaches are predominant as proactive adaptations are more difficult to develop [53]. It is also possible to apply both adaptations simultaneously, where the reactive adaptation can act as a backup if the proactive adaptation fails. For example, this is the case for a traffic management system. Usually, a load-balancer uses models to predict or forecast the traffic demand at specific times and could adjust traffic lights or route recommendations proactively. However, unexpected events (e.g., accidents), rare events (e.g., soccer games), or wrong predictions might require reactive adjustments - presumably resulting in traffic jams. Additionally, we extracted information about the control period of the system, if available. Such information can be exact

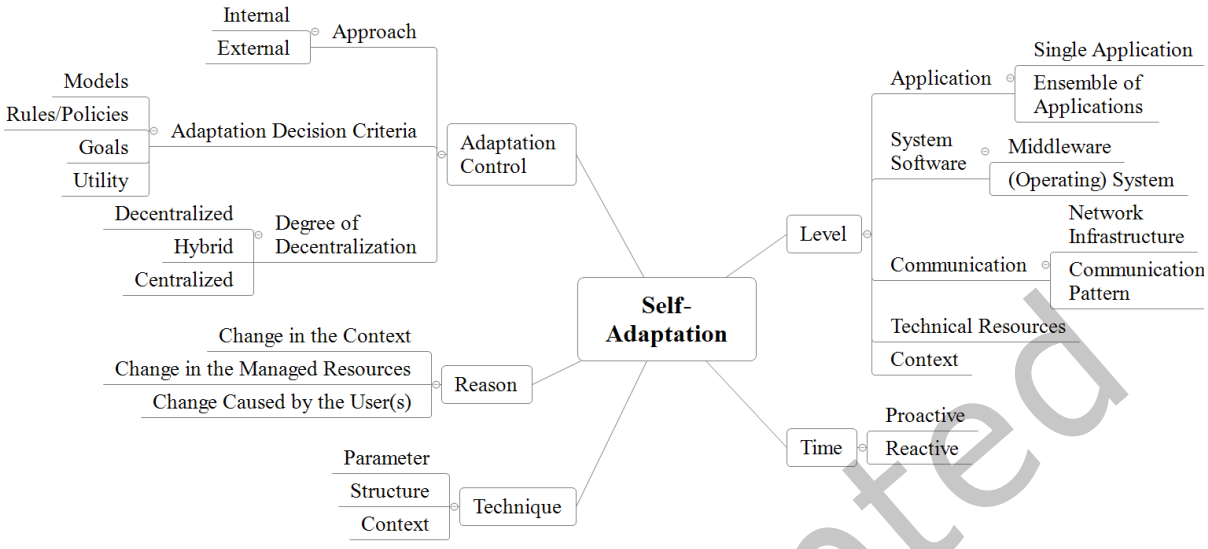

Fig. 3. The taxonomy of self-adaptation [69] describes self-adaptation with the dimensions *time*, *reason*, *level*, *technique*, and *adaptation control*.

quantitative information (such as the length of the feedback cycles, i.e., the frequency of triggering the adaptation control) or qualitative information (such as the regularity of triggering adaptation control).

The **Reason** dimension is required as the system has to locate and analyse the *reasons* for which an adaptation is required. Such reasons can originate from the *environment* in which the system is operating (e.g., when the state of an environment variable changes), its *managed resources* (e.g., hardware or software fault), or system *users* (e.g., a change in the composition of the user group or the user's preference). For example, within the smart energy grid, we have to reason on several dimensions. On the one hand, it is possible that the environment changes, e.g., that the strength of the sunlight is influenced by clouds and resulting in the energy from solar collectors decreasing. On the other hand, the system might compensate for an error in the network, e.g., a breakdown of power lines. Lastly, it is possible that users reflect their behaviour, e.g., when they receive rewards (e.g., a discounted rate) to shift the times for loading electric vehicles from high peak times to lower demand times [65].

The **Level** dimension describes *where* adaptation might induce changes., which can be the *application, system software, communication, technical resources*, or *context*. An *application* or an ensemble of distributed applications is the software with which the users or other back-end systems interact. In distributed applications, *communication* is an important aspect and two perspectives are relevant w.r.t. adaptation: a switch of the logical communication patterns, for example, from a point-to-point communication to a publish-subscribe approach, or a technical adaptation of the network connection, for example, an adaptation of a mobile device's Internet connection from using 3G/4G to Wifi. *System software* which can be the operating system or middleware abstract from hardware. *Technical resources* include all types of hardware. Last, the *context* (i.e., the system environment) can be adapted. For example, an intelligent tunnel lightning system [23, 68] adapts the brightness of the lights according to the light intensity of the surrounding environment.

The **Techniques** dimension includes *parameter* adaptation which is bound to the adjustment of parameters, *structural* adaptation, which triggers changes of the algorithms or system components and *context* adaptation. For example, in the smart home context, a system that automatically shuts the blinds to darken the room when

starting a movie would change the context (i.e., light in the room). Turning on the heating system would represent a parameter adaptation. Automatically integrating a new light into the system would be a structural adaptation.

Whereas the previous four dimensions describe properties of self-adaptation that might be taken into account when reasoning on adaptation, the **Adaptation Control** dimension refers to how to enable the self-adaptation. It is manifested into three characteristics: approach, decision criteria, and degree of decentralisation.

First, internal *approaches* integrate the control for adaptation with the resources that should be adapted, and external ones split them into separated modules [43]. Usually, it is preferable to split the responsibilities for system functionality and adaptation control, hence, follow the external model, for the sake of better maintainability and clear separation of concerns. This way, adaptation is seen as an add-on to a system, e.g., optimised control of traffic lights as a complement to a working traffic light system. Internal approaches intertwine both, system resources and adaptation logic. Those models are usually preferable for small, local adaptation, i.e., error control, such as control of the current functionality of a traffic light and switch to an emergency mode otherwise.

Second, different adaptation *decision criteria* are present in the literature, such as models, rules and policies, goals, and utility functions [72]. Models are all types of information captured in models, e.g., traffic demand models. Rules are usually written in the form of the event - condition - action policies, e.g., identification of increased room temperature (event) above a specified threshold (condition) leading to cooling the room (action). Goals are often represented as goal models. Those models determine the objectives of a system with a quantified metric to measure the achievement of the objective; goals might be composed of different sub-goals and/or (sub-)goals that are only valid for specific situations. A goal for edge computing in IoT applications would be to optimise the performance and improve lives. while maintaining fair access to the grid. Utility functions are composed of different aspects that are all measured individually and aggregated to a single utility value. Those metrics might help to determine the value of an adaptation for the user, e.g., the benefit of a new route recommendation in terms of saved time and/or fuel.

Third, the *degree of decentralisation* for the adaptation control can vary from fully decentralised systems, in which each subsystem has a full adaptation control functionality that controls a specific part of the system, and centralised ones in which only one subsystem controls adaptation of the whole system [139]. Hybrid approaches also exist, where parts of the control functions are centralised, and parts distributed.

Furthermore, we captured some implementation details. First, those include whether the approach integrates the support of GPUs or other technology for enabling AI procedures. Second, we extracted information about the programming paradigm. Mainly, we identified monolithic approaches, microservices, or serverless approaches. Third, we distinguish if approaches have been applied in real-world systems or simulations.

4.2.2 An Autonomous Driving Example. We describe the above taxonomy in more detail using an edge computing-based autonomous driving scenario [94]. Autonomous driving reduces the possibility of accidents by eliminating human errors and increasing the comfort of human drivers. Recent research has shown that cooperative autonomous driving performs better when compared to non-coordinated autonomous driving as cooperation and communication enhance the perspective of vehicles and inform each vehicle on the intended behaviour of others [118]. As shown in Fig. 4, the example scenario is based on an environment equipped with both edge and cloud services including roadside edge units, neighbourhood edge devices, and edge and cloud servers that are used to coordinate and optimise the traffic. Smart traffic lights and vehicles are also equipped with communication technologies and computational resources that allow vehicle-to-vehicle and vehicle-to-infrastructure communication. In this scenario, smart vehicles operate at the edge of the network, while traffic management services run on the cloud. Autonomous vehicles can be implemented as self-adaptive systems, which can make decisions autonomously and react/adapt to changes in their environment; captured by various sensors, such as cameras, LIDAR, or GPS. Such decisions are in accordance with the objectives defined by the users (e.g., minimum travel time, or choosing an environmentally friendly route). In this example, the explicit reason, time, level, technique, and adaptation control are as follows:

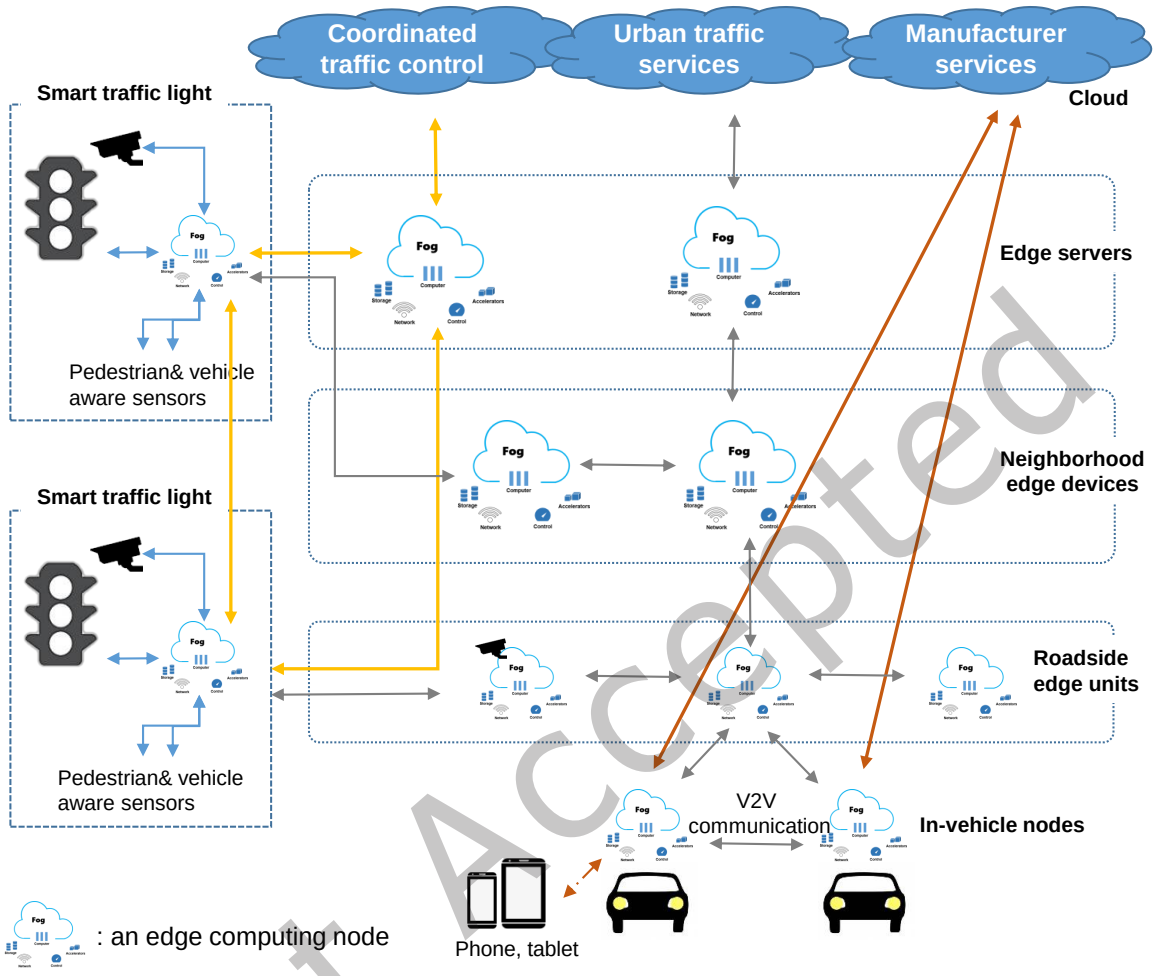

Fig. 4. Traffic Scenario which shows an edge computing communication scenario. V2V = vehicle-to-vehicle.

Reason: adaptations are required when `users`'s objectives or the vehicles' `context` (e.g., sudden traffic in the roads may require rerouting) change.
Time: vehicles may be `reactive` and react to such changes when they occur, or be `proactive` and make changes by predicting a change. For example, using traffic prediction algorithms they can predict a change in their context.
Level: the level can be the technical resources (the vehicle itself, e.g., controlling steering or braking), but also other systems (e.g., communication for coordinated driving) or the network (e.g., switching between communication channels, such as LTE for communication with a traffic recommendation system and direct vehicle-to-vehicle communication like IEEE 802.11p [61] for vehicular communication).
Technique: adaptions such as rerouting changes vehicles' `parameters` such as speed and direction.

Adaptation Control: if the vehicles' reactions are decided by the vehicles themselves, it is considered an `internal` approach, and when such reactions are recommended by the coordination unit of the traffic management system, it is considered an `external` approach. The vehicles' adaptation decision criteria represented as `models` can be defined or derived from learning algorithms (e.g., deep learning).

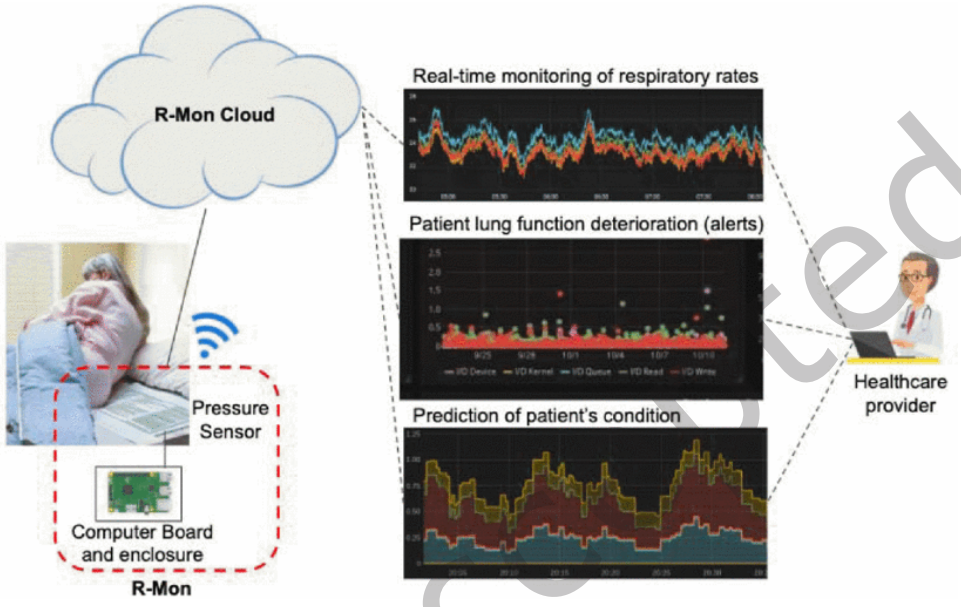

Fig. 5. Framework of R-mon [129]

4.2.3 A Smart Healthcare Example: In this illustration, we are analyzing the utilization of edge computing and the IoT in the healthcare setting, as outlined by Maria et al [129]. The paper introduces the notion of R-Mon, a mobile health tool that offers diagnostic assistance and facilitates real-time communication with healthcare providers. It is suitable for patients in self-isolation and other diseases during pandemics. The authors used a pressure sensor under the bed, connected to an edge computing system, to gather and analyze ballistocardiogram signals. The signal is read by an IoT device, which can store raw data in databases with quick response time (e.g.: InfluxDB) [38] or use a sliding window for nearly real-time processing. The signal-processing algorithms extract the respiration rate, and the data is shared with a remote healthcare professional. Due to the critical nature of smart healthcare, the communication systems involved in this process must have ultra-low latency and real-time response. Fig 5 shows the architecture of the R-mon. **Reason:** The system requires adaptation when the health conditions (Respiration, temperature, etc.) of the patient change.
Time: The edge-enabled IoT device connected to the sensors must be `reactive` to change in the patient's health and **proactive** to process and analyse the captured signals.
Level: We can denote the level as the resources required to process the respiratory rate and pulmonary function of the patients timely. We can also consider the communication resources involved in the whole process.
Technique: The IoT devices adapt to the change in the patient's health by capturing, processing, and analysing the various sensor data captured.
Adaptation Control: After the processing and analysis of the sensor data, if the IoT end-devices using an AI

decide or implement immediate healthcare actions, it is termed the internal approach. However, if the remote healthcare professional proposes treatment using the processed signals, it is considered an external approach.

5 REVIEW ON SELF-ADAPTIVE EDGE COMPUTING SYSTEMS

This section reviews the current status of integration adaptation into edge computing based systems using the taxonomy introduced in Section 4.

5.1 Time

The adaptive system solutions in edge computing can be classified into reactive and proactive according to their adaptation timing. The reactive adaptation normally has a monitor to detect a trigger event that will cause an adaption action. Such an event can be a sudden performance drop or a resource shortage. Proactive adaptation usually predicts the trigger event and prepares adaptation of the system in advance.

5.1.1 Reactive. Ravindra et al. [100] proposed an orchestration platform named ECHO to schedule the data flow on different Edge, Fog, and Cloud resources to support IoT applications. ECHO includes a monitor periodically checking the Quality of Service(QoS) of the application and dynamically migrates tasks from a resource with poor QoS performance to a resource providing sufficient QoS. Jutila [62] introduced adaptive edge devices to provide flow management and QoS awareness. Similar to the ECHO, edge devices include a monitoring solution that collects and measures the data flow from a multitude of network data sources. In this approach, the adaptation happens after changes in the network QoS are detected. Caporuscio et al. [20] employ the classic MAPE-K (with Monitor (M), Analyze (A), Plan (P) and, Execute (E) activities, plus a Knowledge (K)) loop for adaptive service discovery in a fully decentralised manner. Therefore, changes, such as newly offered services or variations of their quality, can be quickly located in the system. Yang et al. [147] propose a reactive privacy preserved approach to adaptively control the access to patients' data according to the changes in a context such as normal and emergency scenarios. In Chabi et al. [24], the authors created a reactive-based distributed system for task offloading in IoT. Their proposed autonomous IoT system adapts swiftly to changes using reinforcement learning. Nassar et al. [91] proposed resource allocation for vehicles and smart city users using the Fog radio access network. They applied the infinite-horizon Markov decision process to create an adaptive system that learns the optimal slicing policy for resource allocation. Bui et al. [17] also presented a reactive adaptive resource allocation deployed at the edge of the IoT network. They modelled a six-factor score-based Match-Making algorithm to balance IoT emergency loads on edge servers. The authors of [42] proposed an adaptive system for managing resources at the edge of the network. Their system reduces communication cost and addresses the problem of small storage capacity, irregular user requests and real-time changes at the edge of the IoT network Wang [134] et al. implement an adaptive system that dynamically allocates computational resources within 36 edge nodes. All the literature mentioned has in common the adaptation follows as a reaction to an identified change in the system.

5.1.2 Proactive. Reacting to trigger events can be unfavourable to some time-sensitive tasks as it is likely to be too late to apply a lightweight adaptation action when the event occurs. For example, in a resource allocation task, if a resource drop happens, the allocated resources may have to be released and reallocated. This will inevitably delay the task, and the monitoring-reacting process itself will spend time, which can also introduce delays in tasks. In addition, resource-constraint systems may not have enough resources to always monitor system status. To overcome the above drawback of the reactive mechanism, many solutions exploded proactive adaptation to anticipate adaptation requirements. Silva et al. [35] introduced a Gaussian process regression to resource allocation in a Fog-Cloud joint infrastructure which allows fog nodes and the cloud to collaborate to provide resources to users. The Gaussian process regression mechanism is used to predict future demands to avoid blocking requests, especially delay-sensitive ones. Tran et al. [127] proposed an adaptive bitrate-aware proactive

cache placement algorithm. This algorithm can increase the cache precision and decrease the backhaul traffic and content access delay compared to the traditional approaches. Chen et al. [25] achieved proactive adaptation by predicting service demand. They proposed an online distributed algorithm named AFC. This algorithm can adapt fog configuration to both temporal and spatial service demand patterns. Wang et al. [122] proposed a simulation toolkit for system evolution to plan the system's actions based on a strategy library before adaptation trigger events occur. Zeng et al. [150] proposed a proactive distributed live video system using the smart camera-edge cluster approach. Their model adaptively balances workload across cameras at the edge of the network. In Wu et al. [142], the authors proposed a decentralized secured IoT framework built on blockchain and intrusion-tolerant at the edge of the network. Xie et al. [143] proposed an adaptive system that compresses the downstream data sent from the IoT devices to reduce the communication cost at the edge of the network.

5.1.3 Summary and Analysis. In adaptation timing, reactive approaches are more dominant than proactive approaches, which is partly because proactive adaptation needs to predict adaptation requirements. Such predictions will require additional monitoring costs and computation efforts. It also requires higher forecast accuracy to avoid additional overhead due to excessive adaptation. Predictive work is more suitable for critical applications that require high time sensitivity.

5.2 Reason

The causes of adaptation actions can be different. In surveyed articles, there are three major triggers: the changes in context, the changes in technical resources, and user-related changes, see Fig. 6. The changes in context occur outside of the edge computing system as changes in network connectivity to the cloud and cloud resources change. The changes in technical resources can be considered as the changes in the system resources, such as network connectivity between edge nodes and edge nodes' computation resources. User-related changes are about changes in users' requirements, locations, and connectivity to edge nodes.

5.2.1 Changes in Context. To manage the dynamic nature of computing environments, Hsieh et al. [59] presented a managed, programmable, and virtualized edge computing platform for IoT analytics in smart cities. To reduce the extra energy cost and moderate deviations for the wireless channel, Sodhro et al. [119] proposed a forward central dynamic and available approach (FCDAA). It optimises the power and extends the battery life with high reliability in AI-based edge computing industrial applications. Liang et al. [31] address the challenges of dynamic change of network status and try to avoid the frequent exchange of information. To do so, they designed a decentralised algorithm that assists users in adaptively selecting optimal video quality levels, which can achieve quick convergence to a modest accuracy of the optimal solution. Laghari et al. [71] address the communication problems in complex Adaptive Networks using cognitive agents, which can perceive their environment, identify changes in the context and react accordingly. Chen et al. [27] proposed a service description method for learning-based applications in fog computing environments. It allows reselecting AI services to better fit the data distribution when the context changes. Similarly, [27] addresses the challenge of efficiently discovering services in dynamic networks, where requirements may change unpredictably. The proposed method groups user requests with similar service requirements and time constraints. This adaptive approach helps manage request peaks and reduces latency in service provisioning.

5.2.2 Changes in Technical Resources. As for changes in technical resources, to manage the mobility of network elements, Daneshfar et al. [36] introduced an integer optimisation formulation to minimise the total cost under a guarantee of service execution despite the uncertainty of resources availability. In the above formulation, each of the edge servers maps to an availability value that captures potential mobility and formulates. Such a value is bound to the QoS of each user's demand or even congestion and service admission rejection at the device or server. Mu et al. [89] adapt the system to the changes in traffic patterns monitored on the network. To

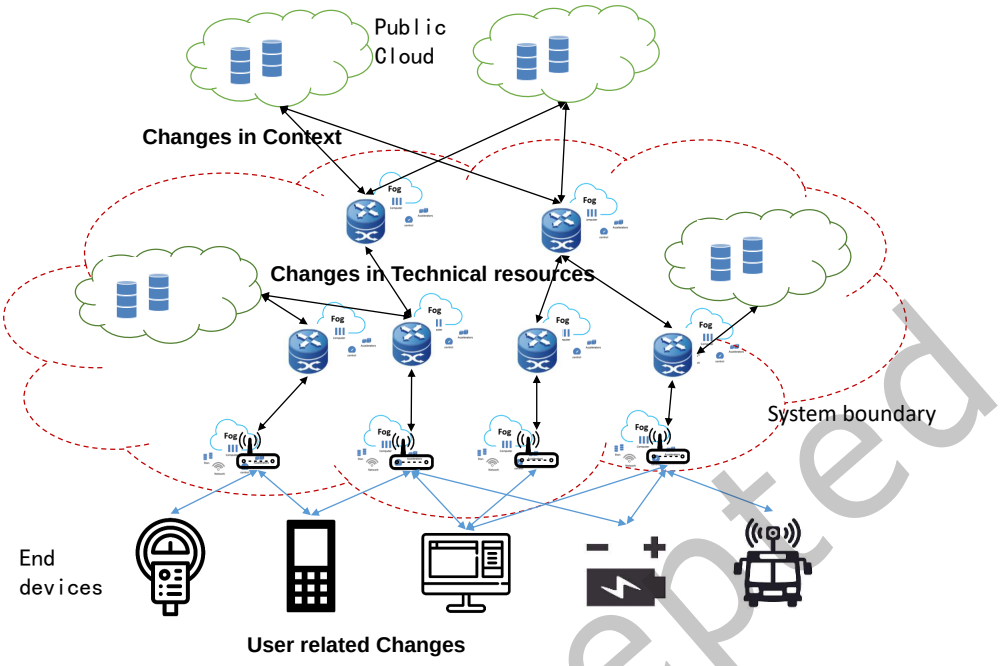

Fig. 6. Three major adaptation triggers: the changes in context, the changes in technical resources, and user-related changes

do so, they presented a novel reinforcement learning model to learn adaptation policies. This model is able to adapt to unknown operating environments and can improve its decision policies using reinforcement obtained through previous interactions with the environment. Hence, it can also minimise the control plane overhead and enhance adaptation precision. NEPTUNE [8, 9] introduces Network- and GPU-aware Management of Serverless Functions at the Edge and manages the changes in the technical resources through adaptability at its three-level control hierarchy, including Topology, Community, and Node levels. This hierarchical structure enables efficient management of function instances and resource allocation while operating independently yet synergistically. NEPTUNE's dynamic resource management capabilities, facilitated by controllers at the Node level, empower the system to scale CPU cores in real-time without service disruption. Similarly, Xu et al. [144] addressed the challenge of incorporating renewable into mobile edge computing and proposed an efficient reinforcement learning-based resource management algorithm to minimise the cost of service delay and operation. This algorithm learns the optimal policy of dynamic workload offloading on the fly to the centralised cloud and edge server provisioning. In such an algorithm, adaptation triggers include dynamics of the information of computation workload, core network congestion state, available battery power, and anticipated renewable power arrival. In [134], insufficient computing capacity of mobile devices, and excessive energy consumption and performance degradation are counted as the reasons for adaptation, and they develop an adaptive edge computing platform, which effectively allocates resources and offloads computations to meet the demands of various IoT applications.

5.2.3 User-Related Changes. Wang et al. [133] tackle user's mobility and propose a comprehensive model that realises online resource allocation in edge cloud systems. Zhao et al. [152] address unexpected changes in user goals and propose a framework that does the adaptation at runtime. The framework includes an offline learning phase, that generates adaptation rules for predefined goal settings, and an online adaptation phase that uses the

generated rules to support adaptation decision-making. Bao Bu et al. [7] also identify the reasons for change to be user related, when the users' preferences may change. These users may have varying levels of priority based on the criticality of their tasks or applications. For example, in the provided scenario of emergency medical services, certain applications such as remote medical assistance or real-time image processing for diagnostics may require immediate access to edge resources due to their critical nature.

5.2.4 Summary and Analysis. For adaptation reasons, we find that many adaptations were triggered by changes in the context or in users, and only a small number of adaptations are due to problems such as insufficient resources (e.g., bandwidth, computation, storage, etc.). This reflects the dynamics of the computing environment of edge systems. Especially the connection between edge nodes and end-users is mostly based on wireless networks, and the uncertainty of network conditions is the main driver of adaptations. Some edge nodes are also powered by batteries, or renewable energy sources that cannot generate power stably (e.g., solar, wind, etc.), so changes in energy conditions are the other main contextual driver for adaptations. Interestingly, any kind of security (e.g., hacking or attacks on critical infrastructure) or safety (e.g., blackouts due to political reasons) related issues are underrepresented.

5.3 Level

Adaptation systems can be used in different levels of the layered system framework like application, Middle-ware, OS, communication, technical resources, and context that indicate where adaptation might induce changes. Primarily, the methodologies we examined resulted in modifications.

5.3.1 Application. In the application level, the setting or structure of the application, such as the service provided to the user, was altered. Wang et al. [133] distribute the workload from each user to any of the edge clouds. Silva et al. [35] allocate resources for new tasks by instantiating a virtual machine. Daneshfar et al. [36] achieve mobile fog service allocation. Mu et al. [89] produce the optimal network forwarding rules from the flow pool. Also,the approaches ECHO [100], FCDAA [119], AFC [25], microservice redeployment system [55], EvolutionSim [122], Distream [150], distributed task offloading in autonomous IoT systems [24], DRL network slicing system [91], Reja (a decentralised blockchain and intrusion tolerance system) [142], and a score-based match-making resource allocation [17] all act on the application level.

5.3.2 Other levels. In other layers, the adaptive computing method proposed by Jutila [62] is used at the technical resources level. IRM-SA method proposed by Gerostathopoulos et al. [47] is used at the context level. The fog orchestrator proposed by Wen et al. [136] is used in Communication. Similarly, Alkhabbas et al. [4] adapts the network deployment to user's goal, and Li et al. [75] changes network topology to reduce energy consumption, which are all adapt at the level of communication. Orsini et al. [95] presented CloudAware as a holistic approach and use this approach at the middleware level. Xie et al. [143] proposed data compression at the network level to reduce communication cost, CampEdge [134] analyse network traffic and allocate them to a suitable edge node, Li et al. [75] used graph-based partition algorithm to improve task offloading in MEC at the network level, and the authors of [4] used goal-driven policy to deploy IoT device.

5.3.3 Summary and Analysis. The current adaption mechanisms are more focused on the application level. This is because most of the current adaptation requirements are user-driven and mainly for providing better QoS to users. Updates at the application level are relatively inexpensive. As mobile operators join the edge computing platform construction, there also exist platform-level adaptive strategies, for example in message middleware, service middleware, network communication protocols, etc. Please note that the paradigm of computation employed in edge computing systems will affect the adaptation result. For instance, monolithic architectures, while robust and simple, may not provide the flexibility and scalability needed for certain edge computing applications.

Adaptation in such systems occur more at communication level to allow the systems to adjust caching speed [74] or select a suitable connection[75]. On the other hand, microservices or function-as-a-service architectures could offer greater modularity and scalability. Adaptive edge syetems based such paradigms are basically adapting at application level [27, 100, 127], but they might also introduce complexity and overhead. Similarly, the type of computation allowed by these systems, such as CPU-only or hybrid CPU and GPU, could influence their adaptation levels and effectiveness in handling specific tasks. For example, hybrid CPU and GPU computations can support machine learning-based application better due to the parallel processing capabilities of GPUs, but may need to deal with adaptation at the level of operating system [8, 9].

5.4 Technique

Adaptation technique can be parametric or structural. Parametric adaptation refers to changing values of a parameter. Structural adaptation subsumes changes in the structure, e.g., starting a module or switching the network connection.

5.4.1 Parameter. In surveyed articles, parameter adaptation is bound to adjusting parameters such as traffic and bandwidth, processing depth (PD), and feasible video rate. Even though parameter adaptation consists of many types of parameters, each parameter adjustment still plays a unique role in the adaptation field. Among the parameters mentioned above, the adjustment of the traffic and bandwidth can optimise and control traffic flows and network resources. The adjustment of PD can track the point of minimum total energy. And adjusting the feasible video rate can maximise QoE. For example, Jutila [62] proposed an adaptive computing method for IoT networking at the network edges to optimise and control traffic flows and network resources. Cao et al. [19] presented a camera-based wireless sensor node with a self-optimising end-to-end computation and communication design. The end-to-end self-optimizing node always tracks the point of minimum total energy by adjusting PD. Rahman et al. [99] presented an edge cloud-assisted rate adaptation solution to enhance the user experience in a cellular network by adjusting the feasible video rate. Such a solution is designed for multi-access edge computing-assisted HTTP adaptive streaming by using edge cloud capabilities. Qiang et al. [74] illustrated a capacity-aware edge caching framework that adapts the fog cache capacity to the base-station connectivity capacity to balance the caching requirement and the resources. Xie et al. [143] have considered 'offloading points', which are candidate offloading points during the adaptation process. The proposed strategy evaluates various offloading points and adjusts parameters accordingly to meet latency requirements while maximizing inference accuracy. Similarly, [17] have proposed a platform for resource allocation in edge-based deployments, to facilitate edge infrastructure analysis, event management, edge node assignment, and resource allocation. The adaptation process here focuses on improving the existing match-making algorithm to handle priority and preemption effectively.

5.4.2 Structure. Structural adaptation approaches in the edge computing domain mainly adjust tasks, services, and containers. Approaches reschedule tasks to minimise task make-span and communication costs. The adjustment of service distributions is usually for satisfying dynamic user demands. Migrating containers will optimise the static deployment to meet the run-time service requirements [71]. For example, Liu et al. [79] proposed an ADGTS algorithm. This algorithm schedules tasks and fog resources collaboratively to minimise the response time of tasks and reduce communication costs. Chen et al. [28] proposed a novel architecture that achieves a cloud-to-things continuum of services and enables adaptive fog service distributions to tackle dynamic user requirements. Tang et al. [124] proposed a novel container migration algorithm and a corresponding architecture to support mobility tasks with various application requirements. ALkhabas et al. [4] provides an Adaptation Manager component that is event-driven and responsible for adapting goal-driven IoT Systems in response to events triggered by the Context Manager. It monitors the knowledge base for unprocessed events, and depending on the events

detected, it may trigger adaptation processes that involve adjusting tasks and services to accommodate changes in the environment or hardware infrastructures. Additionally, Wu et al. [142] introduces adaptation through its ATS scheme, which introduces modifications to specific components related to data governance, access control, and authentication. These modifications may involve updates to the consensus mechanisms, smart contracts, or cryptographic protocols to support the adaptive threshold signature scheme effectively. Thus, the adaptation influences the structure of these components within the overall system architecture, while the core architecture of the decentralized IoT framework remains intact.

5.4.3 Summary and Analysis. We found that in terms of adaptation techniques previous approaches do not reflect a clear preference between parameter adaptation and structural adaptation. This is based on the fact, that selecting an adaptive technology strongly depends on the underlying adaptive task. Adaptive tasks, such as resource allocation or service assignment, tend to employ structural adaptation. In contrast, adaptive tasks in terms of QoS guarantees, such as data caching or energy allocation, tend to be updated at the parameter level without affecting the system structure, etc. It can be seen that most of the adaptive updates that may affect only a single service or task, or the QoS for only a short period of time, are more likely to choose parameter-level techniques, while updates for longer-term and potentially impacting the execution of multiple tasks are more likely to choose structural update techniques. There are also many adaptive updates at the context-level, because there are many interactions with the environment for IoT applications, such as automatic adjustment of, e.g., temperature or lighting.

5.5 Adaptation Control

Adaptation control describes how the decision making is implemented. Three dimensions are important here. The approach dimensions state the integration of decision-making with the system itself. The degree of decentralisation refers to how (de)centralised the decision-making is implemented. Last, there exist different adaptation decision criteria for reasoning on adaptation.

5.5.1 Approach. Adaptation approaches can be classified as internal or external. In an internal approach, there are usually some modules intertwined with the system such as an edge node or an engine in an edge node performing the adaptation. Compare to the internal approach, for the external approach additional modules exist that are dedicated to the adaptation decision-making.

Internal. Gatouillat et al. [46] introduced a quality-of-service driven self-adaptation framework to focus on the robust detection of an emergency medical event especially heart malfunctions and the triggering of such emergency medical response. This framework has an internal cardiac health estimator model during the monitoring process. It can infer cardiac health status and request medical help for cardiac malfunctions. Also, the FCDAA proposed by Sodhro et al. [119], and the IRM-SA method proposed by Gerostathopoulos et al. [47] all used the internal approach. Similarly, in [42], the edge server is responsible for the adaptation process, specifically the paper discusses how edge servers collaborate to handle user requests and relieve system pressure. When an edge server is overloaded or cannot fulfill a user request, it can collaborate with other edge servers within the network to ensure timely data retrieval. This collaborative effort involves adapting to the dynamic demands of users and the network conditions.

External. Concerning the external approach, the ECHO system proposed by Ravindra et al. [100] contains processors which consist of external runtime engines. These processors collect the data flow and send it to the external engines for results. Such engines may be in-memory Java libraries, command-line executables, or a remote Big Data platform. The fog computing system proposed by Chen et al. [28] includes a special outside platform. This platform organises the fog nodes and maintains the structure of the fog network to achieve

horizontal computation across different networks, devices, and semantic domains. The Task Scheduling system proposed by Liu et al. [79] includes a fog computing layer. And this layer is divided into multiple fog groups. The group members communicate with the cloud center of the external network though Muccini et al. [90] offer a hybrid self-adaptive IoT architecture and models an emergency handling system. The external adaptation approach offered in this work either uses a central master-slave pattern or a collaborative regional planning pattern, in both the adaptation control is located in a separate module that requires adaptation. Furthermore, the system proposed in [4], includes module that are solely responsible for the adaptation process, including the goal manager, deployment planner, and context manager that monitor the system and identify the need for adaptation and adapt the system nodes.

5.5.2 Degree of Decentralisation. In surveyed articles, there are four major structures in terms of the degree of centralisation for adaptive edge systems: Centralised, Decentralised, Hierarchical, and Hybrid. Central adaptation control means there is a central node to control the adaptation. Decentralised adaptation control means that parts of the adaptation control are distributed (potentially redundantly) on several nodes or that all nodes for adaptation control have full control functionality. Hierarchical adaptation control relies on edge nodes in different hierarchical levels. The low-level edge nodes control end device-related adaptation, and the high-level edge nodes subsume control over the lower-level edge nodes' adaptation. Hybrid adaptation control combines central functionality and decentralised control, which might also lead to some kind of hierarchy. In the following, we target examples of central and decentralised control.

Centralised. Zhang et al. [151] proposed an inventory management system for a warehousing company that integrates RFID technology and a self-adaptive model. This system contains a central unit and can provide decision support for inventory activities such as inbound and outbound activities, inventory location suggestions and incident handling. Safavifar et al. [104] also employs a centralised reinforcement learning based orchestrator to adapt the workload of the edge nodes according to the dynamic demand. Yigitoglu et al. [148] proposed a framework called Foggy to allocate dynamic resources and automatically deploy the application in fog computing architecture. Foggy employs a central orchestration server to monitor resource usage and dynamically adjust container placement according to resource availability. The orchestration server can also notify the user if a possible bottleneck is found. Wen et al. [136] wrap IoT devices as services and use service composition techniques to orchestrate the devices. Composed services can flexibly deal with services' dynamic variations and transient operational behaviour. An orchestrator fog node is proposed to control the orchestration process. This paper only introduced the centralised realisation of the orchestration process, but the authors claim that it can be implemented in a distributed manner. Seiger et al. [115] use a centralised MAPE-K feedback loop to monitor and analyse process execution, along with graph queries on a knowledge base to understand any inconsistencies and adapt to changes when needed.

Decentralised. Chen et al. [25] developed an adaptive fog configuration model for industrial IoT systems. It applies a distributed joint optimising algorithm based on Gibbs sampling and Lyapunov optimisation. The algorithm allows fog nodes to work decentralized and to jointly optimize service hosting and task admission decisions. It leverages Markov Random Field (MRF) and Graph Coloring to manage information exchange among fog nodes, making the adaptation decision-making always use currently available system information. Through this decentralised adaptive fog configuration, IoT systems can achieve close-to-optimal performance.

Hybrid. The hybrid level of decentralisation refers to solution where adaptation process is handled by both decentralised and centralised entities. For example, Nasser et al. [91] proposed an adaptation process at the hybrid decentralization level because it proposed a coordinated network slicing model based on multiple FNs (Functional Nodes) cooperating through an edge controller (EC). This model combines elements of both centralized and decentralized approaches, where EC acts as a central coordinator, managing and coordinating edge resources

across multiple FNs in an edge cluster. Despite the central coordination provided by the EC, each FN retains autonomy and independence in serving its local area. FNs can make local decisions based on their own resources and capabilities, such as serving a service request locally if resources are available or handing it over to neighboring FNs or the cloud if necessary. This decentralized aspect allows FNs to adapt to local conditions and requirements while still operating within the framework set by the EC. Other works such as [8, 143] have also offered similar solutions. In [8, 9], the system's hierarchical/hybrid control structure, comprising topology, community, and node levels, enables decentralized decision-making across various aspects. At the Topology level, the network is partitioned into communities based on geographic proximity and latency considerations, enabling localized management. Each community, managed independently at the community level, dynamically handles function placement and routing policies to adapt to changing workload demands. Within communities, the node level autonomously allocates resources and manages function instances, ensuring efficient resource utilization and workload management. NEPTUNE's decentralized control architecture, with independent controllers operating at different frequencies, minimizes coordination overhead and facilitates seamless scalability. This decentralized approach enhances system flexibility, adaptability, and resilience, making NEPTUNE well-suited for dynamic edge computing environments.

5.5.3 Adaptation Decision Criteria / Adaptation Goals. Edge Computing systems can have varying adaptation goals, encompassing a range of objectives aimed at enhancing system functionality, reliability, and resource utilization. One primary goal is to minimize network latency by dynamically adjusting processing and data management strategies based on changing network conditions and workload demands. Additionally, adaptation aims to optimize resource consumption by efficiently allocating computational resources and minimizing energy usage, particularly in resource-constrained edge devices. Another key objective is to maximize system availability by implementing fault-tolerant mechanisms and proactive recovery strategies to mitigate disruptions caused by hardware failures or network outages. Moreover, adaptation goals may include devising efficient data migration policies to facilitate seamless data movement between edge devices and centralized data centers, ensuring data consistency and integrity while minimizing communication overhead. Overall, the overarching goal of adaptation in edge computing systems is to enhance system agility, responsiveness, and scalability to meet evolving user requirements and environmental dynamics effectively. The key point that we will be discussing in this section is how these goals are translated as decision criteria for adaptation. For example, how can we model various interrelating parameters of a system that might impact energy efficiency, and response time of the system (both being the goal of the system). Several techniques are identified for articulating and translating the adaptation goals to formalised decision criteria.

These approaches can be classified into four major aspects: rule-based adaptation, model-based adaptation, utility-driven adaptation and goal-driven adaptation. The rule-based adaptation decides how to adapt a system through a particular rule. The model-based adaptation builds a model to simulate or abstract parameters, configurations, and environmental changes in the system. The utility-driven adaptation is usually based on a utility value and aims to maximise the overall system utility by evaluating the utility values of different strategies and selecting the one with the highest utility. The goal-driven adaptation has a specific goal which usually stems from the user's requirement.

Models. Minh et al. [87] proposed an approach that combines a multi-tier fog computing architecture and a service placement mechanism. This approach models the deployment of tasks as the objective function to maximise the number of tasks deployed on the fog landscape while meeting the constraints of resources and QoS. Similarly, Safavifar et al. [105] proposes a model with multiple objectives to minimise the use of edge servers, and utilise the extreme edge nodes while maintaining the required QoS. Rahman et al. [99] proposed a greedy-based algorithm and an Integer Non-linear Programming (INLP) formulation to reason the adaptation plan. The greedy-based algorithm is employed to maximise utility such as the Quality of Experience (QoE) of the

streaming clients. The INLP optimisation model jointly optimise the viewing experience of the competing clients for rate adaptation. Among mentioned articles, Wang et al. [133] formulate the resource allocation problem into a linear program and propose an online greedy approach to solve it. Daneshfar et al.[36] formulate mobile fog service allocation as an integer program with the constraints of the number of servers each user can use, a user budget (e.g. battery, quota, etc.), the total amount of service requests to be sent from various users to server set in fog and the requested QoS Level. They solve the integer program by a general Gurobi Optimisation solver. Chen et al. [25] formulate the adaptive fog configuration problem for Industrial IoT as a mixed-integer nonlinear stochastic optimisation problem with long-term constraints. They also provide an online algorithm based on Lyapunov optimisation to solve it. Lin et al. [76] offer a model-based adaptation technique for data placement in scientific workflows using a particle swarm optimisation algorithm as a model to map particles to data placement. Xie et al. [143] proposed an adaptation strategy that relies on a computational model that comprehensively captures the relationships among key factors influencing system performance, including latency, accuracy, network conditions, and resource constraints. This approach is facilitated by a components such as the instance segmentation network, device-edge inference model, data transmission model, and time delay model.

Rules. Seiger et al. [116] apply the MAPE-K feedback loop to detect and repair cyber-physical inconsistencies. There are two assertions in the proposed MAPE-K loop. The satisfied assertion in the analyse phase defines the required change within relevant context data. The analyser component uses this assertion to evaluate the success of the operation. The compensation assertion defines a condition for initiating the Plan phase if some unexpected behaviour occurs. When the compensation assertion output is true, the system will assume inconsistency exists and request a change to the planner component. Caporuscio et al. [20] introduce a reinforcement learning approach to allow the adaptation system to dynamically study the service selection rule from its experience. To avoid service overloading, this approach uses load-dependent quality attributes during rule learning. This approach aims to guarantee good overall quality for the delivered services. In mentioned article [151], The system proposed by Zhang et al. includes a self-adaption process. The self-adaption consists of two parts, firstly, modifying the base weight of the pre-set knowledge bases of a node according to the decision evaluations from the previous decision-making process, and secondly, determining the time of the local knowledge base regeneration which can be triggered manually by users or automatically by measure the changed base weight.

Goals. The reinforcement learning model proposed by Mu et al. [89] minimises the long-term control plane overhead. The INLP formulation proposed by Rahman et al. [99] can maximise QoE. The ADGTS algorithm proposed by Liu et al. (2018) [79] can minimize the response time of tasks and reduce communication costs. To tackle dynamic user requirements, Chen et al. [28] [26] proposed a fog computing system. To minimise the edge system's expected long-term cost, Xu et al. [144] proposed a reinforcement learning-based resource management algorithm. To develop a model for the total energy of the sensor node, Cao et al. [19] presented a camera-based wireless sensor node.

5.5.4 Summary and Analysis. In terms of adaptation control, most edge computing systems separate the control of adaption and the functionality or resources of the edge system. With the promotion of 5G, the edge system has gradually developed into an open and shared computing platform. The external adaption control will gradually become mainstream to such a platform. This will support not only a separation of concerns in the system design and dedicated modules but also support completely separating the module for control and system functionality also locally due to the high bandwidth of 5G communication. The current adaptation solutions are more centralised. This is mainly because the current task is still relatively small in terms of the surface affected by changes. For example, network changes between an edge node and the end devices it serves only affect the services associated with this edge node, so it is reasonable for this edge node to act as the central node to control adaptation. In other changes that will affect the topological network system composed of edge nodes, adaptive approaches

using a decentralised approach will reduce the workload on the central node (especially regarding the amount of data to process) while preventing single-node failure. With multi-tier edge computing becoming more and more popular, there will be an increasing number of complex services and applications that need to be supported by multiple edge nodes. Although there is no complex adaptive control structure yet, a more finely designed adaptive structure is also expected in the future in order to improve efficiency and reduce the cost of adaptations.

5.6 Implementation details of the surveyed papers

To shed light on the details of the control period implemented in each of the surveyed paper, in this section we dive deep into the details of their implementation approaches and the technologies used as enablers of the proposed solutions. To do so, we have included information on the *programming paradigm* selected for the implementation of the solutions, which includes monolithic approaches, microservices, or serverless approaches. We have also explored weather or not GPUs and artificial intelligence were used either as enablers or enhancer of the adaptation process, as these might have huge impact on the control period and how it is managed. Lastly, we distinguish if approaches have been applied in real-world systems or simulated environments. The details of this analysis is reported in Table 3.

According to Table 3, the surveyed papers include a diverse array of application domains, including user mobility, resource allocation, IoT, SDN flow entry, industrial applications, video caching, safety storage, surveillance systems, smart environments, smart cities, video streaming, and more. Within these domains, a variety of programming paradigms are employed, with microservices and monolithic architectures being predominant, alongside exploration of distributed and decentralized paradigms. While only a few papers leverage advanced technologies like AI and GPU for simulation, and real-world implementation, majority of the works do not explicitly mention such utilization. Control periods duration vary widely across applications, typically ranging from milliseconds to minutes (when such data is available). Furthermore, while many papers focus on simulations of proposed systems, fewer delve into real-world implementations, indicating a gap between theoretical exploration and practical deployment.

Additionally, we can analyze how different programming paradigms might influence the control strategies implemented in adaptive systems [56]. For example, Monolithic architectures, characterized by a single, tightly integrated codebase, may exhibit relatively simpler control mechanisms compared to more distributed or modular architectures. Systems built on monolithic architecture, such as those discussed in Seiger [114] and Silva [35], might have control periods typically in the scale of seconds. This is because monolithic systems often handle all functionalities within a single process, allowing for more straightforward control mechanisms with less overhead. However, there could be exceptions, such as Liu [79], where a monolithic architecture is employed, but the system requires control at a finer granularity, possibly due to specific application requirements like real-time processing in IoT environments. On the other hand, microservices architectures, characterized by decomposing applications into smaller, independent services, offer more flexibility and scalability in terms of control mechanisms. Systems based on microservices, like those discussed in [25] and [27], may have varying control periods depending on the nature of individual services and their communication patterns. Control periods in microservices-based systems can range from milliseconds to seconds, reflecting the diverse processing requirements and communication latencies among distributed services. Finally, systems built on distributed or decentralized architectures, as seen in papers like Minh [87] and Caporuscio [20], often involve coordination among multiple nodes or entities distributed across a network. Control periods in such architectures may vary based on factors like message propagation delays, network conditions, and synchronization requirements. They might range from milliseconds to seconds or even minutes, depending on the scale and dynamics of the distributed system.

Table 2. Adaptation in Edge Computing Systems: Summary of the review (1.Parameter, 2.Structure, 3.Change in Context, 4.Change in the managed resources, 5.Change caused by the users, 6.Internal, 7.External, 8. Rules, 9. Models, 10.Goals, 11.Centralised, 12.Hierarchical, 13.Decentralised, 14.Hybrid, 15.Reactive, 16.Proactive, 17. Application, 18. Middleware, 19. Operating System, 20. Communication, 21. Technical resources, 22. Context

First Author	Technique		Reason			Approach		Decision Criteria			Degree of centralisation				Time		Level					
	Pa[1]	S[2]	CIC[3]	CITR[4]	URC[5]	I[6]	E[7]	R[8]	M[9]	G[10]	Ce[11]	Hi[12]	D[13]	Hy[14]	R[15]	Pro[16]	A[17]	MW[18]	OS[19]	Com[20]	TR[21]	Con[22]
Wang [133]		•			•	•			•		•				•		•					
Silva [35]		•			•		•	•					•			•	•					
Daneshfar [36]		•		•			•		•		•				•		•					
Ravindra [100]		•	•				•		•		•				•		•					
Seiger [114]		•	•				•	•			•				•		•					
Jutila [62]	•		•			•		•			•				•						•	
Mu [89]	•			•		•		•	•		•				•		•					
Cao [19]	•		•			•					•				•							•
Minh [87]		•	•				•		•		•				•		•					
Hsieh [59]		•	•				•	•			•				•		•					
Sodhro [119]	•		•			•		•			•				•		•					
Gerostathopoulos [47]	•		•			•		•			•		•		•							•
Liu [79]		•	•				•	•			•				•		•					
Seiger [116]		•	•				•	•			•				•		•					
Orsini [95]			•									•						•				
Tran [127]	•		•		•	•			•				•		•	•						•
Chen [25]		•			•		•	•		•			•			•	•					
Zhang [151]	•		•			•			•		•				•							•
Chen [28]		•	•		•		•						•		•		•					
Tang [124]		•	•		•		•	•			•				•		•					
Yigitoglu [148]		•	•				•				•				•		•					
Caporuscio [20]		•	•				•	•					•		•		•					
Zhao [152]	•		•		•		•	•			•				•							•
Xu [144]	•			•			•	•			•				•		•					
Wen [136]		•		•			•	•			•		•		•					•		
Rahma [99]	•		•			•			•	•	•				•							•
Liang [31]	•		•		•	•		•			•				•							•
Bing Lin [77]	•		•				•		•		•					•	•					
Mirjami Jutila [62]	•		•				•	•			•				•					•		
Ali Hassan [119]	•		•				•	•			•				•		•					
Roberto Casadei [22]		•	•				•	•						•	•					•		
Roberto Casadei [21]		•	•				•		•					•	•		•			•		
Baresi [9]		•	•	•			•		•					•	•		•		•	•		
Nanxi Chen [27]		•	•		•	•			•			•				•	•					
Xuejun Li [75]		•		•		•		•			•					•				•	•	
Alkhabbas [4]		•	•				•			•	•				•					•	•	
He [55]		•	•			•			•		•				•		•					
Wang [122]		•		•		•		•	•		•				•		•					
Zeng [150]	•		•			•			•				•			•	•					
Chabi [24]		•	•			•			•				•		•		•					
Nassar [91]		•	•			•			•					•	•		•					
Wu [142]		•	•			•			•				•			•	•					
Bui [17]	•				•	•		•			•				•		•					
Fang [42]		•	•			•			•		•				•		•					
Xie [143]	•		•			•			•		•					•				•		
Wang [134]		•		•		•			•				•		•					•		

Table 3. Surveyed papers application areas and details of implementation

Author	Application Domain	Programming Paradigm	GPU/AI	Simulation	Real-world	Control Period/Frequency
Wang [133]	User Mobility	Monolithic	N/A	Y	N	N/A
Silva [35]	Resource Allocation	Distributed	N/A	Y	N	Seconds
Daneshfar [36]	Service Allocation	Distributed	N/A	Y	N	N/A
Ravindra [100]	Hybrid Dataflows	Microservices	GPU	N	Y	N/A
Seiger [114]	IoT	Monolithic	N/A	Y	N	Seconds
Jutila [62]	IoT	Monolithic	N/A	Y	N	N/A
Mu [89]	SDN Flow Entry	Monolithic	AI	Y	N	Seconds
Cao [19]	IoT	Monolithic	N/A	Y	N	N/A
Minh [87]	IoT	Distributed	N/A	Y	N	Seconds
Hsieh [59]	IoT	Microservices	N/A	N	Y	Seconds
Sodhro [119]	Industrial Applications	Serverless	AI	Y	N	N/A
Gerostathopoulos [47]	IoT	Monolithic	N/A	N	Y	N/A
Liu [79]	IoT	Monolithic	N/A	Y	N	Milliseconds
Seiger [116]	IoT	Monolithic	N/A	N	Y	N/A
Orsini [95]	Mobile	Distributed	N/A	Y	N	N/A
Tran [127]	Video Caching	Monolithic	N/A	Y	N	N/A
Chen [25]	Industrial IoT	Microservices	N/A	Y	N	N/A
Zhang [151]	Safety storage	Monolithic	N/A	Y	N	N/A
Chen [28]	Cross-domain	Microservices	N/A	Y	N	N/A
Tang [124]	Cross-domain	Microservices	AI	N	Y	N/A
Yigitoglu [148]	Automated IoT	Microservices	N/A	N	Y	Milliseconds/Seconds
Caporuscio [20]	Cross-domain	Decentralized	N/A	Y	N	N/A
Zhao [152]	Cross-domain	Monolithic	N/A	N	Y	Seconds
Xu [144]	Energy Harvesting	Microservices	N/A	Y	N	Minutes
Wen [136]	IoT	Microservices	N/A	Y	N	N/A
Rahman [99]	Video Streaming	Monolithic	N/A	Y	N	Seconds
Liang [31]	Video Rate Adaptation	Monolithic	N/A	Y	N	Seconds
Bing Lin [77]	Data Placement	Monolithic	N/A	Y	N	Milliseconds
Mirjami Jutila [62]	IoT	Monolithic	N/A	Y	N	Seconds
Ali Hassan [119]	Industrial IoT	Monolithic	AI	Y	N	Seconds
Nanxi Chen [27]	Surveillance systems	Microservices	AI	Y	Y	N/A
Roberto Casadei [22]	Smart Environments	Services	N	Y	N	1 Hz
Roberto Casadei [21]	Smart City	Services	N	Y	N	N/A
Baresi [9]	Cross-domain	Serverless	GPU	Y	Y	5 seconds period
Alkhabbas [4]	Smart home	Monolithic	N	Y	N	N/A
He [55]	Cross-domain	Microservices	N/A	Y	Y	N/A
Wang [122]	Cross-domain	Microservices	N/A	Y	N	seconds
Zeng [150]	Video Analytics	Distributed	GPU	Y	N	Milliseconds
Chabi [24]	IoT	Distributed	AI	Y	N	Seconds
Nassar [91]	IoT	Monolithic	AI	Y	N	Seconds
Wu [142]	IoT	Distributed	N/A	Y	N	Seconds
Bui [17]	IoT	Monolithic	N/A	Y	N	Seconds
Fang [42]	Cross Domain	Monolithic	N/A	Y	N	Hours
Xie [143]	IoT	Monolithic	N/A	Y	N	Seconds

6 DISCUSSION AND FUTURE DIRECTION

This section discusses the current research challenges and provides several avenues for future direction and finally, a discussion on threats to validity is provided.

6.1 Discussion and Research Gaps

Based on the review presented in Section V, we have identified several points of discussion and research challenges related to integrating adaptation principles into edge computing based systems design.

Multi-layer solutions.

In the literature review, we only identified two-hybrid approaches [21, 22] and not a single hierarchical one in our literature analysis. This seems unexpected as for the general control, there are many hybrid or hierarchical models present in edge computing. Especially, in combination with latency and privacy-optimised edge infrastructures, cloud-based microservice architectures can be regionally deployed and aggregate data across regions which introduces flexibility and elasticity. The cloud on the other hand aggregates and processes data from all production plants in a global data space and feds knowledge down to all edge nodes. One reason for the results of our literature analysis might be that the adaptive aspect is a relatively new research direction. Further, it might be possible that researchers in this domain focus on the direct interactions between edge devices and non-edge devices for simplification reasons without targeting a multi-layer setting. However, currently, the automatic scaling of microservice architectures in modern cloud computing systems seems to follow a centralised adaptation approach. Especially as use cases (e.g., the production process in industrial IoT) can benefit from this hybrid architecture (due to the low latency edge infrastructure with rapid reactions to parameter changes) it seems important for the future to study those multi-layer scenarios and their specifics for adaptation.

Combined adaptation techniques. All analysed approaches focus on either parametric or structural adaptation. Consequently, such approaches either change the parameters of existing system modules or change the system composition, e.g., de-activate software modules or change the communication channel. This might be beneficial, as it can reduce the solution space of possible adaptations, decrease the required computation for decision-making, and accelerate the response time for adaptation. However, it also reduces the system's flexibility as fewer adaptation options are feasible. Especially, changing the composition of system elements and optimising the parameter setting in combination can be beneficial.

Proactiveness. In our study, only a few of the literature opt for proactive adaptation. From the user's point of view, proactive adaptation is preferable, since it avoids interruptions in the user's workflow with the system. However, proactive adaptation has several challenges, especially in scenarios in which multiple systems share the context. It is highly dependent on the correctness of predictions, as faulty predictions can cause sub-optimal adaptations. The major challenges here are predicting the time of an event with high enough accuracy, as well as predicting user behaviour and rare events. This is especially complicated due to the incompleteness of the information in distributed edge computing scenarios.

Trigger for adaptation. Besides five works ([36, 89, 136, 136, 144]) which target technical resources as trigger for adaptation and and eight of them ([25, 28, 31, 35, 124, 127, 133, 152]) which integrate the user preferences as adaptation trigger, the rest of the analysed approaches react on context changes. As edge computing systems often operate in uncertain environments, reacting to changes in the environment is definitely an important requirement. However, it is interesting that reactions to issues in the technical resources are less commonly addressed (only in [36, 89, 136, 144]). It could be possible that researchers focus on the interactions in the edge systems and assume the correct technical resources. In addition, as the reaction to system errors is a typical scenario for self-adaptive systems, many approaches exist that might address this aspect.

Human-in-the-loop. Similar to reactions to issues in the technical resources, user-triggered adaptations are less commonly observed in the analysed approaches ([25, 28, 31, 35, 124, 127, 133, 152]). Self-adaptive systems try

to exclude the user per definition from active participation in the adaptation process. However, the user might influence the adaptation by changing the system's goals. Hence, a self-adaptive system requires procedures to (i) capture the relevant information, such as the preferences of the users, and (ii) include this information for reasoning on adaptation. As edge computing systems often integrate humans and machines seamlessly, this requires another type of human-computer interaction and human-in-the-loop integration. There are several challenges when integrating human-in-the-loop in the adaptation processes [14]. First, the systems need to integrate confidential data for the reasoning of adaptation, which might result in privacy issues. Second, the trust of the users in the system's capabilities can be an issue. Third, as the self-adaptive system performs autonomic adaptations, accountability and liability issues might arise. Taking such issues into account reflects another important research challenge for adaptive edge computing systems.

Level of adaptation. Regarding the level of adaptation, only one of the analysed approaches has addressed adaptation of technical resources [62]. However, this is an important aspect that must be addressed in edge computing and particularly in mobile edge computing as devices are mobile and the technical resources that are available might change and influence the available adaptation options on that level. Similarly, communication ([21, 22, 62, 136]) and middleware [95] are less commonly addressed. As those aspects focus on the network infrastructure (i.e., the physical network connection consisting of network cards, routers, etc.) and communication patterns (i.e., the logical communication such as the interaction style between the elements, e.g., event-based communication or pub/sub communication), it seems to be highly important for mobile systems. Furthermore, context adaptation (addressed in [19, 31, 47, 99, 127, 151, 152]), which refers to the inclusion of adapting the context through adjusting actuators, leads to research issues that are less integrated into the analysed approaches. So far, context is mainly included for analysis purposes (see context as a trigger for adaptation). For context adaptation, the context must be integrated into adaptation planning, i.e., information about the context-altering capabilities of the system (i.e., the actuators) is necessary for planning. The reasoning for adaptation must be aware of those adaptation possibilities. Also addressing the mentioned three categories of adaptation levels (technical resources, communication, context) seems to be an open issue for adaptive edge computing systems.

Control Period. The control period in mobile Edge Computing is a critical factor that determines how swiftly and effectively an adaptive system can respond to dynamic environmental changes, including fluctuating network conditions and variable workload demands. It directly impacts the system's ability to meet stringent latency requirements, optimize resource utilization, and ensure energy efficiency and system stability. Different layers of self-adaptation control, ranging from application-level to infrastructure-level, can significantly influence the time horizon of the control period [139], with higher-level adaptations (e.g., application) potentially allowing for larger control periods due to their broader context and objectives, while lower-level adaptations (e.g., network or hardware) may require shorter control periods to respond rapidly to immediate and technical changes. This layered approach enables a nuanced adaptation strategy that balances immediate responsiveness with strategic, long-term adjustments. The control period is inherently system-specific and depending on use-case characteristics, as it is influenced by unique operational dynamics and requirements of each Edge Computing system and the specific applications it supports. Factors such as the criticality of the application, the volatility of the operating environment, and the specific performance and reliability levels dictate the optimal frequency of adaptation, ensuring that the control period is precisely calibrated to meet the nuanced demands of each scenario. Hence, we extracted information about the control period of the system from the papers identified in the literature review (see Table 3). Such information can be exact information naming frequencies or qualitative information describing the regularity of triggering adaptation control. However, for most of the works, it was not possible to extract the relevant information. For those that specified this information, it was not possible to identify a relation between the control period and other analyzed aspects. As understanding and optimizing the control period is essential for the design and implementation of adaptive strategies in the highly dynamic and resource-constrained environment of Edge Computing, we plan further analysis as future work.

6.2 Smart Application and Adaptation: Future Direction

This section is focused on the main characteristics of edge computing-based smart applications, where self-adaptation can leverage more resilient systems, which can better operate in a dynamic environment with heterogeneous entities, mobile devices, and constrained resources.

Heterogeneity of entities (software/hardware): Edge computing systems comprise multiple heterogeneous entities that operate and interact in an environment. Adaptive communication and computation models can address the various changes that such entities' varied communication and computation requirements can cause in the systems. The adaptive models will be able to understand the reason for such changes and choose the right technique to be applied at the right level. For example, an adaptive method for communication can facilitate various communication channels such as WiFi, Bluetooth, etc. According to the discussed taxonomy, such an adaptive model (proactive or reactive) can be implemented using different techniques at different levels.

Multi-modality of sensed data: Heterogeneous entities might also provide multi-modal data to a system, and to fully utilise the available data the processing and computation mechanisms must be able to handle (e.g., store, process, and analyse) the new modes of data. For example, a traditional traffic monitoring system cannot integrate and fully utilise a new set of cameras and monitoring systems with various types of sensors when it is not defined to be adaptive. In such scenarios, for example, different adaptation control strategies can be in place, with different goals and utilities that can direct the systems level of decentralisation and if needed utilise cloud services to analyse the newly collected data (if the available computational capacity is not enough to handle new modes of data).

Interoperability of applications: Various applications can be linked in principle to improve the overall performance of a system. For example, if a disaster management application is linked with structural health monitoring, traffic management, and the healthcare system would probably better serve the citizens' needs by rerouting the traffic dynamically via safer infrastructure and distributing the injured people to hospitals according to their needs. To achieve such interoperability, possible adaptation levels and techniques can be studied to design a system that allows the different applications to interact and adapt parametrically or structurally when the context changes.

Data distribution (training and raw data): Data-driven models like artificial intelligence (AI) will create a large demand for data matching. AI applications rely on labelled data for training to gain the ability to identify the label information in raw data. However, the accuracy of such models depends on the consistency of training data with the raw data, including data features and data distribution, which is also known as the assumption of Independently Identically Distribution (IID). In IoT, this consistency is difficult to achieve due to the dynamic nature of the raw data. To deal with the data mismatch (a.k.a., the non-IID problem), adapting suitable AI services to changes in data distribution and data characteristics becomes an important requirement for edge-enabled IoT applications.

Run time task Orchestration: Low latency computation is one of the crucial requirements of Edge computing-based applications such as traffic management and autonomous driving and requires run time task orchestration. Operating in dynamic environments with dynamic availability of resources requires adaptive task orchestration methods that can facilitate task offloading under different unpredictable conditions [24, 75]. This includes working with new nodes, new communication channels, and new applications or operating systems.

Mobility of sensors (impact on how data is collected): Real-time decision-making is challenging in many edge computing applications and is even more complicated when the data is also collected in real time. In such situations, the availability and quality of data might be compromised due to the sensors mobility. For example, imagine a real-time traffic monitoring map application that updates the traffic flow based on the collected data from devices that consented to participate. If these devices do not distribute evenly in the system the collected data and decisions will not be reliable. In such cases, an adaptive system can understand the reason for adaption

(e.g., change in density of managed data collectors) and source data from other available sources, or utilise prediction techniques to improve the decision-making accuracy.

Unavailability of services: Service can be unavailable due to mobility, edge node failure, etc. In such cases, if a system is designed to be adaptive it can seek alternative solutions. For example, if all available edge nodes are busy, a task orchestrator can re-prioritise the task offloading schedule to minimise service unavailability or to avail of cloud services where possible. When a service fails an adaptive system can handle such a situation by finding alternative services. For example, in an intelligent transport system imagine the traffic light service fails, in this situation the adaptation mechanism can utilise the connected and autonomous vehicles' ability to control the system. This requires a hybrid degree of decentralisation, where as long as a traffic light is responsive all junctions can be centrally signalised, and in case of failure of such services a decentralised traffic control service can be activated.

Composing new services: New value-added services can be composed using services provided by one or more applications. This will require edge computing systems to proactively identify the emerging requirements, and be able to compose the current services provide a new service. A proactive adaptive system design will be able to predict such requirements and act accordingly.

Simulators and Real-world prototypes: It is crucial to take into account the implementation strategies used by the research works that have been reviewed. Most applications in this research domain use simulation instead of real-world prototyping due to the high cost involved and the early developmental stage. However, simulation software plays an essential role in real-world implementation [106]. Setting up hardware for an adaptive edge computing environment is costly and difficult to modify frequently. Hence, various simulators act as the prototype of the actual system. Simulators provide the flexibility of designing prototypes to experiment with the adaptability of edge-enabled IoT applications on different combinations of scenarios and use cases in a controlled environment. It is common knowledge that the real-world implementation of IoT applications poses several challenges due to different manufacturing standards and defects. Therefore, to consider application adaptability in these environments, it is necessary to take into account factors such as the type of environment, evaluation tools, platform specifics, the underlying effects of hardware and software simulators, and the computational and latency cost involved.

6.3 Threats to Validity

In the following, we present relevant threats to validity. Further, we discuss how we try to minize their risk for compromising the results.

Inclusion of Literature. A common issue with systematic literature reviews is the choice of relevant literature. Using a keyword-based search, it cannot be guaranteed that the set of terms covers all relevant publications. Further, domain-specific aspects might not be taken into account. Hence, we concentrated on not solely having a keyword-based approach, but following a footnote chasing respectively backward chaining approach [15], meaning that we explicitly followed the references of analysed papers. Similarly, the choice of sources is highly relevant. We cover the most relevant databases and Google Scholar, however, it is still possible that a relevant publication is not listed in the searched sources. Further, the results may be slightly biased due to the manual steps of our methodology. However, we try to minimise the risk of such effects as several researchers confirmed each step and analysed each paper.

Applied Definitions. Additionally, given that similar concepts might be named differently in papers, or vice versa, slightly different concepts might be captured under the same name, and some uncertainty might be left when using the taxonomy for evaluation of the platooning coordination approaches. This is especially true for the rather new concept *edge computing* which might trigger different understanding for different researchers or, vice versa, other names might exist for the concept/ideas of edge computing.

Applied Taxonomy. Further, we applied in this work the taxonomy of Krupitzer et al. [69] for classifying the identified, relevant works. Since the taxonomy is commonly applied in the field of self-adaptive systems, it provides a sound basis for such a comparison. However, we acknowledge that the taxonomy might not capture all the specifics of MEC systems. Hence, this might be the first starting point for an analysis of the two fields of MEC and adaptive systems. For future work, we plan to derive from our results an adjusted taxonomy for adaptive MEC systems.

Control Period. A significant threat to the validity of our study arises from the integration and analysis of the control period within the adaptive systems of MEC systems. Given the control period's critical role in determining a system's responsiveness to environmental changes, its ability to meet latency requirements, and its impact on resource optimization and system stability, our analysis may not fully capture the complexities and nuances of adaptive strategies in Edge Computing environments. The lack of detailed information on the control period's implementation across different layers of self-adaptation control and its system-specific and use-case specific nature in the reviewed literature limits our ability to comprehensively understand and evaluate the effectiveness of these adaptive systems. This oversight may hinder our ability to provide a complete picture of how adaptive strategies can be optimized for the dynamic and resource-constrained contexts typical of Edge Computing, underscoring the need for future research that more thoroughly investigates this pivotal aspect. We try to counteract this by a first analysis of those properties based on what is stated in the analyzed works. However, a detailed analysis is part of future work.

Analysis Approach. Finally, we base our discussion on the appearance of manifestations of the taxonomy. We acknowledge that those numbers do not necessarily reflect the importance of the dimensions. Therefore, we use these proportions as an entry point into the analysis and discuss why these proportions appear as they do and their implications. Furthermore, the interpretation of these numbers in an isolated view of a single dimension is not meaningful. Additionally, the distribution of publications is related to the topic areas that are deemed as relevant within the academic research community and their funding sources or by industry. This bears little relationship to which topics are most important in order to facilitate real-world implementation. Those important topics are often less valuable from the perspective of academic publications and reviewers, and less likely to be published, so they are less represented in the paper count. However, as this paper fully focuses on the academic literature, this seems to be a systematic issue for a structured review of academic literature. Finally, this paper mainly discusses the computational aspects of the self-adaptive edge computing paradigm, with a focus on the latency requirements. Any changes in the computational resources that are not adaptable can significantly affect the system's performance. However, future works would cover other aspects such as data privacy, data migration and orchestration, and resource allocation.

7 CONCLUSION

Edge Computing systems are designed to place the computational services and resources closer to the user, reducing latency, preserving privacy, and providing context awareness to ensure the quality of service and experience at applications. Latency and context awareness are essential in IoT systems. They assist the IoT system in understanding the device environment and creating complex real-time decisions. For these reasons, a paradigm shift towards Edge Computing systems in various applications such as intelligent transport systems, smart energy grids, and IoT-based services is evident. However, such applications must be resilient and handle the occurrence of an enormous number of changes rooted in environment characteristics, heterogeneity and mobility of their nodes/devices, and dynamic availability of resources to be able to utilise available resources and enhance their performance and interoperability. To address these challenges self-adaptation design principles can be used to identify such changes and integrate suitable adaptation techniques at various levels using various approaches.

This paper presented a literature review on state-of-the-art adaptive Edge Computing systems with a focus on applications that opperate in environments, where ocurrance of change cannot be avoided and in most cases even predicted. These applications include, Industrial IoT and automation, intelligent transport systems, critical structural health monitoring, where several unexpected events might happen at the production line, on the streets, or on the structures that might impact the system performance and ultimately decision making and adaptation process. We identified that most of the analysed approaches target the application level, and react to changes in the context to cope with the dynamics of the environment. Changes caused by the user or in the system itself are less explored as triggers for adaptation. Reasons for this might be that research concentrates on the application level. Further, anticipating user-related triggers is very complex. Both parametric and structural techniques are used within the literature and the external adaptation approach is the most commented on in the reviewed literature. Furthermore, from the various decision criteria highlighted in the taxonomy, the literature has mostly used rules as the adaptation decision criteria and has less explored models and goals as alternatives criteria. Most approaches have chosen centralised and reactive adaptation processes, where decentralised, hybrid and proactive adaptation processes could be better fit for the applications that operate in such dynamic environments, with non-stationary devices and unavoidable changes. Lastly, most approaches have explored application-level adaptation and fewer studies adaptation at levels such as operating systems, middleware technical resources, communication, and context.

Smart applications that are developed using the edge computing paradigm can leverage self-adaptive system design to utilise their resources better and improve their performance, resilience and interoperability. More specifically, the heterogeneity of devices and entities is a core characteristic of such applications which cannot be fully known at application design time, therefore adaptive system design will allow such applications to still be operational even if new devices appear in the system. Moreover, such heterogeneous devices can potentially sense different modes of data, and if such data can be utilised better through a system that can adapt its input and output data more accurate decisions can be made.

In modern cities, smart infrastructure and smart services are interlinked and cannot function in isolation in the context of IoT-assisted applications. Therefore, adaptive application design can improve the interoperability of such systems and services. In such systems, real-time data is collected from the usually mobile devices, and the same data must be processed on the computational resources that are mobile too. This creates several challenges that can be addressed through an adaptive system design, which can adapt its data collection and sampling and task orchestration based on proactive techniques. Therefore, an in-deepth analysis of how these adaptation mechanisms have been implemented would be interesting. However, this would require an in-deepth analysis in the form of comparing the implememtation details. This is not the scope of this first explorative study. For future work, it might be feasible to do such in-deepth analysis of few systems limited to specific applications.

The critical importance of the control period in adaptive MEC, which is pivotal for determining the system's responsiveness to environmental changes, meeting latency requirements, and ensuring efficient resource utilization and system stability, is an important aspect that we wanted to analyze in the literature review. Despite our efforts to extract detailed information on control periods from the literature, we found that such data were often not explicitly reported or lacked a clear relationship with other adaptation aspects. Recognizing the significance of this gap, we aim to conduct a deeper analysis of the control period within adaptive MEC systems in future work. This exploration will seek to uncover potential relationships between the control period and other critical aspects of system design and implementation, addressing the gap in understanding how control frequency impacts the efficacy of adaptive strategies in dynamic and resource-constrained environments.

ACKNOWLEDGMENTS

This project has received funding from the RE-ROUTE Project, the European Union's Horizon Europe research and innovation programme under the Marie Skłodowska-Curie grant agreement No: 101086343.

REFERENCES

[1] Nasir Abbas, Yan Zhang, Amir Taherkordi, and Tor Skeie. 2018. Mobile Edge Computing: A Survey. *IEEE Internet of Things Journal* 5, 1 (2018), 450–465. https://doi.org/10.1109/JIOT.2017.2750180

[2] Eyhab Al-Masri, Ibrahim Diabate, Richa Jain, Ming Hoi Lam, and Swetha Reddy Nathala. 2018. Recycle. io: An IoT-enabled framework for urban waste management. In *2018 IEEE International Conference on Big Data (Big Data)*. IEEE, 5285–5287.

[3] Eyhab Al-Masri, Ibrahim Diabate, Richa Jain, Ming Hoi Lam Lam, and Swetha Reddy Nathala. 2018. A serverless IoT architecture for smart waste management systems. In *2018 IEEE International Conference on Industrial Internet (ICII)*. IEEE, 179–180.

[4] Fahed Alkhabbas, Ilir Murturi, Romina Spalazzese, Paul Davidsson, and Schahram Dustdar. 2020. A Goal-Driven Approach for Deploying Self-Adaptive IoT Systems. In *2020 IEEE International Conference on Software Architecture (ICSA)*. 146–156. https://doi.org/10.1109/ICSA47634.2020.00022

[5] Muhammad Rizwan Anwar, Shangguang Wang, Muhammad Faisal Akram, Salman Raza, and Shahid Mahmood. 2021. 5G-enabled MEC: A distributed traffic steering for seamless service migration of internet of vehicles. *IEEE Internet of Things Journal* 9, 1 (2021), 648–661.

[6] Hamidreza Arasteh, Vahid Hosseinnezhad, Vincenzo Loia, Aurelio Tommasetti, Orlando Troisi, Miadreza Shafie-khah, and Pierluigi Siano. 2016. Iot-based smart cities: a survey. In *2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC)*. IEEE, 1–6.

[7] Cheikh Saliou Mbacke Babou, Bernard Ousmane Sane, Ibrahima Diane, and Ibrahima Niang. 2019. Home edge computing architecture for smart and sustainable agriculture and breeding. In *Proceedings of the 2nd International Conference on Networking, Information Systems & Security*. 1–7.

[8] Luciano Baresi, Davide Yi Xian Hu, Giovanni Quattrocchi, and Luca Terracciano. 2022. NEPTUNE: Network- and GPU-Aware Management of Serverless Functions at the Edge. In *Proceedings of the 17th Symposium on Software Engineering for Adaptive and Self-Managing Systems*. Association for Computing Machinery, New York, NY, USA, 144–155. https://doi.org/10.1145/3524844.3528051

[9] Luciano Baresi, Davide Yi Xian Hu, Giovanni Quattrocchi, and Luca Terracciano. 2024. NEPTUNE: A Comprehensive Framework for Managing Serverless Functions at the Edge. *ACM Transactions on Autonomous and Adaptive Systems* 19 (2024), 1–32.

[10] Johan Barthélemy, Nicolas Verstaevel, Hugh Forehead, and Pascal Perez. 2019. Edge-computing video analytics for real-time traffic monitoring in a smart city. *Sensors* 19, 9 (2019), 2048.

[11] Mahdi Bashari, Ebrahim Bagheri, and Weichang Du. 2017. Dynamic software product line engineering: a reference framework. *International Journal of Software Engineering and Knowledge Engineering* 27, 2 (2017).

[12] Soeren Becker, Florian Schmidt, Anton Gulenko, Alexander Acker, and Odej Kao. 2020. Towards AIOps in Edge Computing Environments. In *2020 IEEE International Conference on Big Data (Big Data)*. 3470–3475. https://doi.org/10.1109/BigData50022.2020.9378038

[13] Simon Elias Bibri and John Krogstie. 2017. Smart sustainable cities of the future: An extensive interdisciplinary literature review. *Sustainable cities and society* 31 (2017), 183–212.

[14] Robert Birke, Javier Cámara, Lydia Y. Chen, Lukas Esterle, Kurt Geihs, Erol Gelenbe, Holger Giese, Anders Robertsson, and Xiaoyun Zhu. 2017. Self-aware Computing Systems: Open Challenges and Future Research Directions. In *Self-Aware Computing Systems*. Springer, 709–722.

[15] Andrew Booth. 2008. Unpacking your literature search toolbox: on search styles and tactics. *Health Information & Libraries Journal* 25, 4 (2008), 313–317. https://doi.org/10.1111/j.1471-1842.2008.00825.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1471-1842.2008.00825.x

[16] Yuriy Brun, Giovanna Di Marzo Serugendo, Cristina Gacek, Holger Giese, Holger Kienle, Marin Litoiu, Hausi A. Müller, Mauro Pezzè, and Mary Shaw. 2009. Engineering Self-Adaptive Systems through Feedback Loops. In *Software Engineering for Self-Adaptive Systems*. LNCS, Vol. 5525. Springer, 48–70.

[17] The Bao Bui, Aly Sakr, Juan Castrillón, and Rolf Schuster. 2021. Six-factors Score-based Match-making Based on Priority and Preemption for Resource Allocation in Edge Computing. In *2021 IEEE International Conference on Edge Computing (EDGE)*. IEEE, 44–50.

[18] Alessio Burrello, Alex Marchioni, Davide Brunelli, and Luca Benini. 2019. Embedding principal component analysis for data reduction in structural health monitoring on low-cost iot gateways. In *Proceedings of the 16th ACM International Conference on Computing Frontiers*. 235–239.

[19] Ningyuan Cao, Saad Bin Nasir, Shreyas Sen, and Arijit Raychowdhury. 2017. Self-Optimizing IoT Wireless Video Sensor Node With In-Situ Data Analytics and Context-Driven Energy-Aware Real-Time Adaptation. *IEEE Transactions on Circuits and Systems I: Regular Papers* 64, 9 (2017), 2470–2480. https://doi.org/10.1109/TCSI.2017.2716358

[20] M. Caporuscio, M. D'Angelo, V. Grassi, and R. Mirandola. 2016. Reinforcement Learning Techniques for Decentralized Self-adaptive Service Assembly. In *Service-Oriented and Cloud Computing*, Marco Aiello, Einar Broch Johnsen, Schahram Dustdar, and Ilche Georgievski (Eds.). Springer International Publishing, Cham, 53–68.

[21] Roberto Casadei, Giancarlo Fortino, Danilo Pianini, Wilma Russo, Claudio Savaglio, and Mirko Viroli. 2019. A development approach for collective opportunistic Edge-of-Things services. *Information Sciences* 498 (2019), 154–169.

[22] Roberto Casadei, Danilo Pianini, Mirko Viroli, and Antonio Natali. 2019. Self-organising coordination regions: a pattern for edge computing. In *International Conference on Coordination Languages and Models*. Springer, 182–199.

[23] M. Ceriotti, M. Corrà, and et. al. 2011. Is there light at the ends of the tunnel? Wireless sensor networks for adaptive lighting in road tunnels. In *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*. ACM/IEEE, 187–198.

[24] Abdel Kader Chabi Sika Boni, Youssef Hablatou, Hassan Hassan, and Khalil Drira. 2023. Distributed Deep Reinforcement Learning Architecture for Task Offloading in Autonomous IoT Systems. In *Proceedings of the 12th International Conference on the Internet of Things* (Delft, Netherlands) *(IoT '22)*. Association for Computing Machinery, New York, NY, USA, 112–118. https://doi.org/10.1145/3567445.3567454

[25] Lixing Chen, Pan Zhou, Liang Gao, and Jie Xu. 2018. Adaptive Fog Configuration for the Industrial Internet of Things. *IEEE Transactions on Industrial Informatics* 14, 10 (2018), 4656–4664. https://doi.org/10.1109/TII.2018.2846549 arXiv:1806.07764

[26] Nanxi Chen, Nicolás Cardozo, and Siobhán Clarke. 2018. Goal-Driven Service Composition in Mobile and Pervasive Computing. *IEEE Transactions on Services Computing* 11, 1 (2018), 49–62. https://doi.org/10.1109/TSC.2016.2533348

[27] Nanxi Chen, Qi Sun, Yanbei Li, Hongfeng Shu, Jiamao Li, and Xiaolin Zhang. 2023. Agile Services Provisioning for Learning-Based Applications in Fog Computing Networks. *IEEE Transactions on Services Computing* 16, 4 (2023), 2423–2436. https://doi.org/10.1109/TSC.2023.3239667

[28] Nanxi Chen, Yang Yang, Tao Zhang, Ming Tuo Zhou, Xiliang Luo, and John K. Zao. 2018. Fog as a Service Technology. *IEEE Communications Magazine* 56, 11 (2018), 95–101. https://doi.org/10.1109/MCOM.2017.1700465

[29] Songqing Chen, Tao Zhang, and Weisong Shi. 2017. Fog Computing. *IEEE Internet Computing* 21, 2 (2017), 4–6. https://doi.org/10.1109/MIC.2017.39

[30] Betty H. Cheng, Rogério Lemos, and et. al. 2009. Software Engineering for Self-Adaptive Systems: A Research Roadmap. In *Software Engineering for Self-Adaptive Systems*. LNCS, Vol. 5525. Springer, 1–26.

[31] Liang Chengchao, He Ying, Yu F. Richard, and Zhao Nan. 2018. Enhancing Video Rate Adaptation with Mobile Edge Computing and Caching in Software-defined Mobile Networks. *IEEE Transactions on Wireless Communications* (2018), 1–1.

[32] Industrial Internet Consortium. 2019-10-24. The Edge Computing Advantage. *An Industrial Internet Consortium White Paper* Version 1.0 (2019-10-24).

[33] Yuya Cui, Honghu Li, Degan Zhang, Aixi Zhu, Yang Li, and Hao Qiang. 2023. Multi-Agent Reinforcement Learning Based Cooperative Multitype Task Offloading Strategy for Internet of Vehicles in B5G/6G Network. *IEEE Internet of Things Journal* (2023).

[34] Yuya Cui, Honghu Li, Degan Zhang, Aixi Zhu, Yang Li, and Hao Qiang. 2023. Multiagent Reinforcement Learning-Based Cooperative Multitype Task Offloading Strategy for Internet of Vehicles in B5G/6G Network. *IEEE Internet of Things Journal* 10, 14 (2023), 12248–12260. https://doi.org/10.1109/JIOT.2023.3245721

[35] Rodrigo A.C. Da Silva and Nelson L.S. Da Fonseca. 2018. Resource allocation mechanism for a fog-cloud infrastructure. *IEEE International Conference on Communications* 2018-May (2018), 1–6. https://doi.org/10.1109/ICC.2018.8422237

[36] Nader Daneshfar, Nikolaos Pappas, Valentin Polishchuk, and Vangelis Angelakis. 2018. Service Allocation in a Mobile Fog Infrastructure under Availability and QoS Constraints. *2018 IEEE Global Communications Conference, GLOBECOM 2018 - Proceedings* (2018), 1–5. https://doi.org/10.1109/GLOCOM.2018.8647488 arXiv:1706.04084

[37] Mirko D'Angelo, Simos Gerasimou, Sona Ghahremani, Johannes Grohmann, Ingrid Nunes, Evangelos Pournaras, and Sven Tomforde. 2019. On Learning in Collective Self-Adaptive Systems: State of Practice and a 3D Framework. In *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. 13–24. https://doi.org/10.1109/SEAMS.2019.00012

[38] Sergio Di Martino, Luca Fiadone, Adriano Peron, Alberto Riccabone, and Vincenzo Norman Vitale. 2019. Industrial internet of things: persistence for time series with NoSQL databases. In *2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*. IEEE, 340–345.

[39] Simon Dobson, Franco Zambonelli, Spyros Denazis, Antonio Fernández, Dominique Gaïti, Erol Gelenbe, Fabio Massacci, Paddy Nixon, Fabrice Saffre, and Nikita Schmidt. 2006. A Survey of Autonomic Communications. *ACM Transactions on Autonomous and Adaptive Systems* 1, 2 (2006), 223–259.

[40] European Commission (EC). 2010. Europe 2020: A strategy for Smart, sustainable and inclusive growth. *Working paper {COM (2010) 2020}* (2010).

[41] ETSI. 2015. Mobile edge computing: A key technology towards 5G. *ETSI World Class Standards* (2015). https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp11_mec_a_key_technology_towards_5g.pdf

[42] Juan Fang, Siqi Chen, and Min Cai. 2021. Mobile Edge Data Cooperative Cache Admission Based on Content Popularity. In *2021 IEEE International Conference on Edge Computing (EDGE)*. IEEE, 111–118.

[43] Jacqueline Floch, Svein Hallsteinsen, Erledn Stav, Frank Eliassen, Ketil Lund, and Eli Gjorven. 2006. Using Architecture Models for Runtime Adaptability. *IEEE Software* 23, 2 (2006), 62–70.
[44] Antony Franklin and Supriya Tambe. 2020. Multi-access edge computing in cellular networks. *CSI Transactions on ICT* 8 (05 2020). https://doi.org/10.1007/s40012-020-00276-6
[45] K. Gai, Z. Fang, R. Wang, L. Zhu, and Kkr Choo. 2020. Edge Computing and Lightning Network Empowered Secure Food Supply Management. *IEEE Internet of Things Journal* PP, 99 (2020), 1–1.
[46] Arthur Gatouillat, Youakim Badr, and Bertrand Massot. 2017. QoS-Driven Self-adaptation for Critical IoT-Based Systems. In *Service-Oriented Computing – ICSOC 2017 Workshops*, Lars Braubach, Juan M. Murillo, Nima Kaviani, Manuel Lama, Loli Burgueño, Naouel Moha, and Marc Oriol (Eds.). Springer International Publishing, Cham, 93–105.
[47] Ilias Gerostathopoulos, Tomas Bures, Petr Hnetynka, Jaroslav Keznikl, Michal Kit, Frantisek Plasil, and Noël Plouzeau. 2016. Self-adaptation in software-intensive cyber–physical systems: From system goals to architecture configurations. *Journal of Systems and Software* 122 (2016), 378–397. https://doi.org/10.1016/j.jss.2016.02.028
[48] Fatemeh Golpayegani, Ivana Dusparic, Adam Taylor, and Siobhán Clarke. 2016. Multi-agent collaboration for conflict management in residential demand response. *Computer Communications* 96 (2016), 63–72.
[49] Fatemeh Golpayegani, Saeedeh Ghanadbashi, and Maha Riad. 2021. Urban Emergency Management using Intelligent Traffic Systems: Challenges and Future Directions. In *2021 IEEE International Smart Cities Conference (ISC2)*. 1–4. https://doi.org/10.1109/ISC253183.2021.9562937
[50] Fatemeh Golpayegani, Maxime Gueriau, Pierre-Antoine Laharotte, Saeedeh Ghanadbashi, Jiaying Guo, Jack Geraghty, and Shen Wang. 2022. Intelligent shared mobility systems: A survey on whole system design requirements, challenges and future direction. *IEEE Access* 10 (2022), 35302–35320.
[51] Yan Guo, Qiang Duan, and Shangguang Wang. 2020. Service orchestration for integrating edge computing and 5G network: State of the art and challenges. In *2020 IEEE World Congress on Services (SERVICES)*. IEEE, 55–60.
[52] Eric Gyamfi and Anca Delia Jurcut. 2022. Novel online network intrusion detection system for industrial iot based on oi-svdd and as-elm. *IEEE Internet of Things Journal* 10, 5 (2022), 3827–3839.
[53] Marcus Handte, Gregor Schiele, Verena Matjuntke, Christian Becker, and Pedro José Marrón. 2012. 3PC: System Support for Adaptive Peer-to-Peer Pervasive Computing. *ACM Transactions on Autonomous and Adaptive Systems* 7, 1 (2012), Art. 10.
[54] J. He, J. Wei, K. Chen, Z. Tang, Y. Zhou, and Y. Zhang. 2017. Multi-tier Fog Computing with Large-scale IoT Data Analytics for Smart Cities. *IEEE Internet of Things Journal* PP, 99 (2017), 1–1.
[55] Xiang He, Zhiying Tu, Xiaofei Xu, and Zhongjie Wang. 2019. Re-deploying Microservices in Edge and Cloud Environment for the Optimization of User-Perceived Service Quality. In *International Conference on Service-Oriented Computing*.
[56] Xiang He, Zhiying Tu, Xiaofei Xu, and Zhongjie Wang. 2021. Programming framework and infrastructure for self-adaptation and optimized evolution method for microservice systems in cloud–edge environments. *Future Generation Computer Systems* 118 (2021), 263–281.
[57] M.G. Hinchey and R. Sterritt. 2006. Self-Managing Software. *IEEE Computer* 39, 2 (2006), 107–109.
[58] Cheol-Ho Hong and Blesson Varghese. 2019. Resource Management in Fog/Edge Computing: A Survey on Architectures, Infrastructure, and Algorithms. *ACM Comput. Surv.* 52, 5, Article 97 (sep 2019). https://doi.org/10.1145/3326066
[59] Yu Chen Hsieh, Hua Jun Hong, Pei Hsuan Tsai, Yu Rong Wang, Qiuxi Zhu, Md Yusuf Sarwar Uddin, Nalini Venkatasubramanian, and Cheng Hsin Hsu. 2018. Managed edge computing on Internet-of-Things devices for smart city applications. *IEEE/IFIP Network Operations and Management Symposium: Cognitive Management in a Cyber World, NOMS 2018* (2018), 1–2. https://doi.org/10.1109/NOMS.2018.8406133
[60] SM Riazul Islam, Daehan Kwak, MD Humaun Kabir, Mahmud Hossain, and Kyung-Sup Kwak. 2015. The internet of things for health care: a comprehensive survey. *IEEE Access* 3 (2015), 678–708.
[61] Daniel Jiang and Luca Delgrossi. 2008. IEEE 802.11p: Towards an International Standard for Wireless Access in Vehicular Environments. In *VTC Spring 2008 - IEEE Vehicular Technology Conference*. 2036–2040. https://doi.org/10.1109/VETECS.2008.458
[62] Mirjami Jutila. 2016. An Adaptive Edge Router Enabling Internet of Things. *IEEE Internet of Things Journal* 3, 6 (dec 2016), 1061–1069. https://doi.org/10.1109/JIOT.2016.2550561
[63] Jeffrey O. Kephart and David M. Chess. 2003. The Vision of Autonomic Computing. *IEEE Computer* 36, 1 (2003), 41–50.
[64] Wazir Zada Khan, Ejaz Ahmed, Saqib Hakak, Ibrar Yaqoob, and Arif Ahmed. 2019. Edge computing: A survey. *Future Generation Computer Systems* 97 (2019), 219–235. https://doi.org/10.1016/j.future.2019.02.050
[65] Benedikt Kirpes, Sonja Klingert, Markus Eider, Robert Basmadjian, Hermann de Meer, and Maria Perez Ortega. 2017. EV Charging Coordination to secure Power Grid Stability. In *1st E-Mobility Power System Integration Symposium (Berlin)*. Association for Information Systems.
[66] Samuel Kounev, Peter Lewis, Kirstie L. Bellman, Nelly Bencomo, Javier Camara, Ada Diaconescu, Lukas Esterle, Kurt Geihs, Holger Giese, Sebastian Götz, Paola Inverardi, Jeffrey O. Kephart, and Andrea Zisman. 2017. The Notion of Self-aware Computing. In *Self-Aware Computing Systems*. Springer, Cham, 3–16.

[67] Jeff Kramer and Jeff Magee. 2007. Self-Managed Systems: an Architectural Challenge. In *Proceedings of the Future of Software Engineering (FOSE)*. 259–268.
[68] Christian Krupitzer, Felix Maximilian Roth, Christian Becker, Markus Weckesser, Malte Lochau, and Andy Schürr. 2016. FESAS IDE: An Integrated Development Environment for Autonomic Computing. In *2016 IEEE International Conference on Autonomic Computing (ICAC)*. 15–24. https://doi.org/10.1109/ICAC.2016.49
[69] Christian Krupitzer, Felix Maximilian Roth, Sebastian VanSyckel, Gregor Schiele, and Christian Becker. 2015. A Survey on Engineering Approaches for Self-adaptive Systems. *Pervasive Mob. Comput.* 17, PB (Feb. 2015), 184–206. https://doi.org/10.1016/j.pmcj.2014.09.009
[70] Christian Krupitzer, Timur Temizer, Thomas Prantl, and Claudia Raibulet. 2020. An Overview of Design Patterns for Self-Adaptive Systems in the Context of the Internet of Things. *IEEE Access* 8 (2020), 187384–187399.
[71] Samreen Laghari and Muaz A Niazi. 2016. Modeling the internet of things, self-organizing and other complex adaptive communication networks: a cognitive agent-based computing approach. *PloS one* 11, 1 (2016), e0146760.
[72] Philippe Lalanda, Julie A. McCann, and Ada Diaconescu. 2013. *Autonomic Computing*. Springer.
[73] Heiner Lasi, Peter Fettke, Hans-Georg Kemper, Thomas Feld, and Michael Hoffmann. 2014. Industry 4.0. *Business & information systems engineering* 6 (2014), 239–242.
[74] Qiang Li, Yuanmei Zhang, Yingyu Li, Yong Xiao, and Xiaohu Ge. 2020. Capacity-Aware Edge Caching in Fog Computing Networks. *IEEE Transactions on Vehicular Technology* 69, 8 (2020), 9244–9248. https://doi.org/10.1109/TVT.2020.3001301
[75] Xuejun Li, Tianxiang Chen, Dong Yuan, Jia Xu, and Xiao Liu. 2023. A Novel Graph-Based Computation Offloading Strategy for Workflow Applications in Mobile Edge Computing. *IEEE Transactions on Services Computing* 16, 2 (2023), 845–857. https://doi.org/10.1109/TSC.2022.3180067
[76] Bing Lin, Fangning Zhu, Jianshan Zhang, Jiaqing Chen, Xing Chen, Naixue N Xiong, and Jaime Lloret Mauri. 2019. A time-driven data placement strategy for a scientific workflow combining edge computing and cloud computing. *IEEE Transactions on Industrial Informatics* 15, 7 (2019), 4254–4265.
[77] Rongping Lin, Zhijie Zhou, Shan Luo, Yong Xiao, Xiong Wang, Sheng Wang, and Moshe Zukerman. 2020. Distributed Optimization for Computation Offloading in Edge Computing. *IEEE Transactions on Wireless Communications* 19, 12 (2020), 8179–8194. https://doi.org/10.1109/TWC.2020.3019805
[78] Fang Liu, Guoming Tang, Youhuizi Li, Zhiping Cai, Xingzhou Zhang, and Tongqing Zhou. 2019. A Survey on Edge Computing Systems and Tools. *Proc. IEEE* 107, 8 (2019), 1537–1562. https://doi.org/10.1109/JPROC.2019.2920341
[79] Qianyu Liu, Yunkai Wei, Supeng Leng, and Yijin Chen. 2018. Task scheduling in fog enabled Internet of Things for smart cities. *International Conference on Communication Technology Proceedings, ICCT* 2017-Octob (2018), 975–980. https://doi.org/10.1109/ICCT.2017.8359780
[80] Ling Lyu, Lihong Zhao, Yanpeng Dai, Nan Cheng, Cailian Chen, Xinping Guan, and Xuemin Shen. 2022. Adaptive Edge Sensing for Industrial IoT Systems: Estimation Task Offloading and Sensor Scheduling. *IEEE Internet of Things Journal* 10, 1 (2022), 391–402.
[81] Pavel Mach and Zdenek Becvar. 2017. Mobile Edge Computing: A Survey on Architecture and Computation Offloading. *IEEE Communications Surveys & Tutorials* 19, 3 (2017), 1628–1656. https://doi.org/10.1109/COMST.2017.2682318
[82] Frank Macías-Escrivá, Rodolfo Elias Haber Guerra, Raúl del Toro, and Vicente Hernández. 2013. Self-adaptive systems: A survey of current approaches, research challenges and applications. *Expert Systems with Applications* 40 (12 2013), 7267–7279. https://doi.org/10.1016/j.eswa.2013.07.033
[83] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B. Letaief. 2017. A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Communications Surveys & Tutorials* 19, 4 (2017), 2322–2358. https://doi.org/10.1109/COMST.2017.2745201
[84] Cristian Martín, Daniel Garrido, Luis Llopis, Bartolomé Rubio, and Manuel Díaz. 2022. Facilitating the monitoring and management of structural health in civil infrastructures with an Edge/Fog/Cloud architecture. *Computer Standards & Interfaces* 81 (2022), 103600.
[85] Daniel Martinez, Abdollah Malekjafarian, and Eugene OBrien. 2020. Bridge health monitoring using deflection measurements under random traffic. *Structural Control and Health Monitoring* 27, 9 (2020), e2593.
[86] P M Mell and T Grance. 2011. *The NIST definition of cloud computing*. Technical Report NIST SP 800-145. National Institute of Standards and Technology, Gaithersburg, MD. NIST SP 800–145 pages. https://doi.org/10.6028/NIST.SP.800-145
[87] Quang Tran Minh, Duy Tai Nguyen, An Van Le, Hai Duc Nguyen, and Anh Truong. 2017. Toward service placement on fog computing landscape. *2017 4th NAFOSTED Conference on Information and Computer Science, NICS 2017 - Proceedings* 2017-Janua (2017), 291–296. https://doi.org/10.1109/NAFOSTED.2017.8108080
[88] Ning Mu, Shulei Gong, Wanqing Sun, and Quan Gan. 2020. The 5G MEC applications in smart manufacturing. In *2020 IEEE International Conference on Edge Computing (EDGE)*. IEEE, 45–48.
[89] Ting Yu Mu, Ala Al-Fuqaha, Khaled Shuaib, Farag M. Sallabi, and Junaid Qadir. 2018. SDN flow entry management using reinforcement learning. *ACM Transactions on Autonomous and Adaptive Systems* 13, 2 (2018). https://doi.org/10.1145/3281032 arXiv:1809.09003
[90] Henry Muccini, Romina Spalazzese, Mahyar T Moghaddam, and Mohammad Sharaf. 2018. Self-adaptive IoT architectures: An emergency handling case study. In *Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings*. 1–6.

[91] Almuthanna Nassar and Yasin Yilmaz. 2021. Deep reinforcement learning for adaptive network slicing in 5G for intelligent vehicular systems and smart cities. *IEEE Internet of Things Journal* 9, 1 (2021), 222–235.
[92] Almuthanna Nassar and Yasin Yilmaz. 2022. Deep Reinforcement Learning for Adaptive Network Slicing in 5G for Intelligent Vehicular Systems and Smart Cities. *IEEE Internet of Things Journal* 9, 1 (2022), 222–235. https://doi.org/10.1109/JIOT.2021.3091674
[93] Feyza Yildirim Okay and Suat Ozdemir. 2016. A fog computing based smart grid model. In *2016 international symposium on networks, computers and communications (ISNCC)*. IEEE, 1–6.
[94] OpenFog Consortium. 2017. *OpenFog Reference Architecture for Fog Computing*. Technical Report. Architecture Working Group.
[95] Gabriel Orsini, Dirk Bade, and Winfried Lamersdorf. 2016. Cloud aware: A context-Adaptive middleware for mobile edge and cloud computing applications. In *Proceedings - IEEE 1st International Workshops on Foundations and Applications of Self-Systems, FAS-W 2016*. Institute of Electrical and Electronics Engineers Inc., 216–221. https://doi.org/10.1109/FAS-W.2016.54
[96] Pasquale Pace, Gianluca Aloi, Raffaele Gravina, Giuseppe Caliciuri, Giancarlo Fortino, and Antonio Liotta. 2018. An edge-based architecture to support efficient applications for healthcare industry 4.0. *IEEE Transactions on Industrial Informatics* 15, 1 (2018), 481–489.
[97] T. Patikirikorala, A. Colman, J. Han, and L. Wang. 2012. A systematic survey on the design of self-adaptive software systems using control engineering approaches. In *2012 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. 33–42.
[98] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. 2008. Systematic Mapping Studies in Software Engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering* (Italy) *(EASE'08)*. BCS Learning & Development Ltd., Swindon, UK, 68–77. http://dl.acm.org/citation.cfm?id=2227115.2227123
[99] W. U. Rahman, C. S. Hong, and E. Huh. 2019. Edge Computing Assisted Joint Quality Adaptation for Mobile Video Streaming. *IEEE Access* 7 (2019), 129082–129094.
[100] Pushkara Ravindra, Aakash Khochare, Siva Prakash Reddy, Sarthak Sharma, Prateeksha Varshney, and Yogesh Simmhan. 2017. An Adaptive Orchestration Platform for Hybrid Dataflows across Cloud and Edge. In *Service-Oriented Computing*, Michael Maximilien, Antonio Vallecillo, Jianmin Wang, and Marc Oriol (Eds.). Springer International Publishing, Cham, 395–410.
[101] Ju Ren, Deyu Zhang, Shiwen He, Yaoxue Zhang, and Tao Li. 2019. A Survey on End-Edge-Cloud Orchestrated Network Computing Paradigms: Transparent Computing, Mobile Edge Computing, Fog Computing, and Cloudlet. *ACM Comput. Surv.* 52, 6, Article 125 (oct 2019), 36 pages. https://doi.org/10.1145/3362031
[102] Bhaskar Prasad Rimal, Dung Pham Van, and Martin Maier. 2017. Cloudlet Enhanced Fiber-Wireless Access Networks for Mobile-Edge Computing. *IEEE Transactions on Wireless Communications* 16, 6 (2017), 3601–3618. https://doi.org/10.1109/TWC.2017.2685578
[103] Matthias Rohr, Simon Giesecke, Wilhelm Hasselbring, Marcel Hiel, Willem-Jan van den Heuvel, and Hans Weigand. 2006. A Classification Scheme for Self-adaptation Research. In *Proceedings of the International Conference on Self Organization and Autonomous Systems in Computing and Communications (SOAS)*.
[104] Zahra Safavifar, Saeedeh Ghanadbashi, and Fatemeh Golpayegani. 2021. Adaptive workload orchestration in pure edge computing: A reinforcement-learning model. In *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 856–860.
[105] Zahra Safavifar, Charafeddine Mechalikh, Junfei Xie, and Fatemeh Golpayegani. 2023. Enhancing VRUs Safety Through Mobility-Aware Workload Orchestration with Trajectory Prediction using Reinforcement Learning. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 6132–6137.
[106] Maria Salama, Yehia Elkhatib, and Gordon Blair. 2019. IoTNetSim: A modelling and simulation platform for end-to-end IoT services and networking. In *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*. 251–261.
[107] Mazeiar Salehie and Ladan Tahvildari. 2009. Self-Adaptive Software: Landscape & Research Challenges. *ACM Transactions on Autonomous and Adaptive Systems* 4, 2 (2009), Art. 14.
[108] Farzad Samie, Lars Bauer, and Jörg Henkel. 2019. Edge computing for smart grid: An overview on architectures and solutions. *IoT for Smart Grids* (2019), 21–42.
[109] Theresia Ratih Dewi Saputri and Seok-Won Lee. 2020. The Application of Machine Learning in Self-Adaptive Systems: A Systematic Literature Review. *IEEE Access* 8 (2020), 205948–205967. https://doi.org/10.1109/ACCESS.2020.3036037
[110] Kengo Sasaki, Naoya Suzuki, Satoshi Makido, and Akihiro Nakao. 2016. Vehicle control system coordinated between cloud and mobile edge computing. In *2016 55th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*. IEEE, 1122–1127.
[111] Mahadev Satyanarayanan, Paramvir Bahl, Ramon Caceres, and Nigel Davies. 2009. The Case for VM-Based Cloudlets in Mobile Computing. *IEEE Pervasive Computing* 8, 4 (2009), 14–23. https://doi.org/10.1109/MPRV.2009.82
[112] B. Schilit, N. Adams, and R. Want. 1994. Context-Aware Computing Applications. In *Proceedings of the First Workshop on Mobile Computing Systems and Applications (WMCSA)*. IEEE, 85–90.
[113] Bradley Schmerl, Jesper Andersson, Thomas Vogel, Myra B Cohen, Cecilia M F Rubira, Yuriy Brun, Alessandra Gorla, Franco Zambonelli, and Luciano Baresi. 2017. Challenges in Composing and Decomposing Assurances for Self-Adaptive Systems. In *Software Engineering for Self-Adaptive Systems III. Assurances*. Vol. 9640. Springer, 64–89.

[114] Ronny Seiger, Steffen Huber, Peter Heisig, and Uwe Assmann. 2016. Enabling self-adaptive workflows for cyber-physical systems. In *Lecture Notes in Business Information Processing*, Vol. 248. Springer Verlag, 3–17. https://doi.org/10.1007/978-3-319-39429-9_1

[115] Ronny Seiger, Steffen Huber, Peter Heisig, and Uwe Aßmann. 2019. Toward a framework for self-adaptive workflows in cyber-physical systems. *Software & Systems Modeling* 18, 2 (2019), 1117–1134.

[116] Ronny Seiger, Steffen Huber, Peter Heisig, and Uwe Aßmann. 2019. Toward a framework for self-adaptive workflows in cyber-physical systems. *Software and Systems Modeling* 18, 2 (apr 2019), 1117–1134. https://doi.org/10.1007/s10270-017-0639-0

[117] S. Shevtsov, M. Berekmeri, D. Weyns, and M. Maggio. 2018. Control-Theoretical Software Adaptation: A Systematic Literature Review. *IEEE Transactions on Software Engineering* 44, 8 (Aug 2018), 784–810. https://doi.org/10.1109/TSE.2017.2704579

[118] Steven E. Shladover. 2006. Automated vehicles for highway operations (automated highway systems). *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering* 219 (2006), 53–75.

[119] Ali Hassan Sodhro, Sandeep Pirbhulal, and Victor Hugo C. De Albuquerque. 2019. Artificial Intelligence-Driven Mechanism for Edge Computing-Based Industrial Applications. *IEEE Transactions on Industrial Informatics* 15, 7 (jul 2019), 4235–4243. https://doi.org/10.1109/TII.2019.2902878

[120] Francesco Spinelli and Vincenzo Mancuso. 2020. Toward enabled industrial verticals in 5G: A survey on MEC-based approaches to provisioning and flexibility. *IEEE Communications Surveys & Tutorials* 23, 1 (2020), 596–630.

[121] Xiang Sun and Nirwan Ansari. 2016. EdgeIoT: Mobile edge computing for the Internet of Things. *IEEE Communications Magazine* 54, 12 (2016), 22–29.

[122] Z.Tu M. Wagner .Xu T. Wang, X. He and Z.Wang. 2023. EvolutionSim: An Extensible Simulation Tookit for Microservice System Evolution. In *2023 IEEE International Conference on Web Services (ICWS)*. on press.

[123] Salman Taherizadeh, Andrew C. Jones, Ian Taylor, Zhiming Zhao, and Vlado Stankovski. 2018. Monitoring self-adaptive applications within edge computing frameworks: A state-of-the-art review. *Journal of Systems and Software* 136 (2018), 19–38. https://doi.org/10.1016/j.jss.2017.10.033

[124] Zhiqing Tang, Xiaojie Zhou, Fuming Zhang, Weijia Jia, and Wei Zhao. 2019. Migration Modeling and Learning Algorithms for Containers in Fog Computing. *IEEE Transactions on Services Computing* 12, 5 (2019), 712–725. https://doi.org/10.1109/TSC.2018.2827070

[125] Tanvi Thakur, Aryan Mehra, Vikas Hassija, Vinay Chamola, Rallapalli Srinivas, Karunesh K Gupta, and Ajit Pratap Singh. 2021. Smart water conservation through a machine learning and blockchain-enabled decentralized edge computing network. *Applied Soft Computing* 106 (2021), 107274.

[126] Sven Tomforde, Holger Prothmann, Jürgen Branke, Jörg Hähner, Moez Mnif, Christian Müller-Schloer, Urban Richter, and Hartmut Schmeck. 2011. Observation and Control of Organic Systems. In *Organic Computing – A Paradigm Shift for Complex Systems*. Springer, 325–338.

[127] Tuyen X. Tran and Dario Pompili. 2019. Adaptive Bitrate Video Caching and Processing in Mobile-Edge Computing Networks. *IEEE Transactions on Mobile Computing* 18, 9 (2019), 1965–1978. https://doi.org/10.1109/TMC.2018.2871147

[128] R. Trimananda, A. Younis, B. Wang, B. Xu, and G. Xu. 2018. Vigilia: Securing Smart Home Edge Computing. In *2018 IEEE/ACM Symposium on Edge Computing (SEC)*.

[129] Maria Valero, Hossain Shahriar, and Sheikh Iqbal Ahamed. 2020. R-mon: An mhealth tool for real-time respiratory monitoring during pandemics and self-isolation. In *2020 IEEE World Congress on Services (SERVICES)*. IEEE, 17–21.

[130] Latha Narayanan Valli, Sujatha N., and V. Geetha. 2023. Importance of AIOps for Turn Metrics and Log Data: A Survey. In *2023 2nd International Conference on Edge Computing and Applications (ICECAA)*. 799–802. https://doi.org/10.1109/ICECAA58104.2023.10212414

[131] Blesson Varghese, Nan Wang, Sakil Barbhuiya, Peter Kilpatrick, and Dimitrios S Nikolopoulos. 2016. Challenges and opportunities in edge computing. In *2016 IEEE International Conference on Smart Cloud (SmartCloud)*. IEEE, 20–26.

[132] Prachet Verma, Akshay Kumar, Nihesh Rathod, Pratik Jain, S Mallikarjun, Renu Subramanian, Bharadwaj Amrutur, MS Mohan Kumar, and Rajesh Sundaresan. 2015. Towards an IoT based water management system for a campus. In *2015 IEEE First International Smart Cities Conference (ISC2)*. IEEE, 1–6.

[133] Lin Wang, Lei Jiao, Jun Li, and Max Muhlhauser. 2017. Online Resource Allocation for Arbitrary User Mobility in Distributed Edge Clouds. *Proceedings - International Conference on Distributed Computing Systems* (2017), 1281–1290. https://doi.org/10.1109/ICDCS.2017.30

[134] Zhong Wang, Guangtao Xue, Shiyou Qian, and Minglu Li. 2020. CampEdge: Distributed computation offloading strategy under large-scale AP-based edge computing system for IoT applications. *IEEE internet of things journal* 8, 8 (2020), 6733–6745.

[135] Jane Webster and Richard T. Watson. 2002. Analyzing the Past to Prepare for the Future: Writing a Literature Review. *MIS Q.* 26, 2 (June 2002), xiii–xxiii. http://dl.acm.org/citation.cfm?id=2017160.2017162

[136] Zhenyu Wen, Renyu Yang, Peter Garraghan, Lin Tao, Xu Jie, and Michael Rovatsos. 2017. Fog Orchestration for Internet of Things Services. *IEEE Internet Computing* 21, 2 (2017), 16–24.

[137] Danny Weyns. 2019. Software Engineering of Self-adaptive Systems. In *Handbook of Software Engineering*. Springer, 399–443.

[138] Danny Weyns, Gowri Sankar Ramachandran, and Ritesh Kumar Singh. 2018. Self-managing internet of things. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 10706 LNCS. Springer Verlag, 67–84. https://doi.org/10.1007/978-3-319-73117-9_5

[139] Danny Weyns, Bradley R. Schmerl, Vincenzo Grassi, Sam Malek, Raffaela Mirandola, Christian Prehofer, Jochen Wuttke, Jesper Andersson, Holger Giese, and Karl M. Göschka. 2013. On Patterns for Decentralized Control in Self-Adaptive Systems. In *Software Engineering for Self-Adaptive Systems II*. LNCS, Vol. 7475. Springer, 76–107.

[140] Terence Wong, Markus Wagner, and Christoph Treude. 2021. Self-Adaptive Systems: A Systematic Literature Review Across Categories and Domains. arXiv:2101.00125 [cs.SE]

[141] Rih-Teng Wu, Ankush Singla, Mohammad R Jahanshahi, Elisa Bertino, Bong Jun Ko, and Dinesh Verma. 2019. Pruning deep convolutional neural networks for efficient edge computing in condition assessment of infrastructures. *Computer-Aided Civil and Infrastructure Engineering* 34, 9 (2019), 774–789.

[142] Yusen Wu, Jinghui Liao, Phuong Nguyen, Weisong Shi, and Yelena Yesha. 2022. Bring Trust to Edge: Secure and Decentralized IoT Framework with BFT and Permissioned Blockchain. In *2022 IEEE International Conference on Edge Computing and Communications (EDGE)*. IEEE, 104–113.

[143] Yuanyan Xie, Yu Guo, Zhenqiang Mi, Yang Yang, and Mohammad S Obaidat. 2022. Edge-assisted real-time instance segmentation for resource-limited iot devices. *IEEE Internet of Things Journal* 10, 1 (2022), 473–485.

[144] Jie Xu, Lixing Chen, and Shaolei Ren. 2017. Online Learning for Offloading and Autoscaling in Energy Harvesting Mobile Edge Computing. arXiv:1703.06060 [cs.LG]

[145] Ruizhe Yang, F Richard Yu, Pengbo Si, Zhaoxin Yang, and Yanhua Zhang. 2019. Integrated blockchain and edge computing systems: A survey, some research issues and challenges. *IEEE Communications Surveys & Tutorials* 21, 2 (2019), 1508–1532.

[146] Y. Yang. 2019. Multi-tier computing networks for intelligent IoT. *Nature Electronics* 2, 1 (2019).

[147] Yang Yang, Xianghan Zheng, Wenzhong Guo, Ximeng Liu, and Victor Chang. 2019. Privacy-preserving smart IoT-based healthcare big data storage and self-adaptive access control system. *Information Sciences* 479 (2019), 567–592.

[148] Emre Yigitoglu, Mohamed Mohamed, Ling Liu, and Heiko Ludwig. 2017. Foggy: A Framework for Continuous Automated IoT Application Deployment in Fog Computing. *Proceedings - 2017 IEEE 6th International Conference on AI and Mobile Services, AIMS 2017* (2017), 38–45. https://doi.org/10.1109/AIMS.2017.14

[149] Wei Yu, Fan Liang, Xiaofei He, William Grant Hatcher, Chao Lu, Jie Lin, and Xinyu Yang. 2018. A Survey on the Edge Computing for the Internet of Things. *IEEE Access* 6 (2018), 6900–6919. https://doi.org/10.1109/ACCESS.2017.2778504

[150] Xiao Zeng, Biyi Fang, Haichen Shen, and Mi Zhang. 2020. Distream: Scaling Live Video Analytics with Workload-Adaptive Distributed Edge Intelligence. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems* (Virtual Event, Japan) *(SenSys '20)*. Association for Computing Machinery, New York, NY, USA, 409–421. https://doi.org/10.1145/3384419.3430721

[151] Lizong Zhang, Nawaf Alharbe, and Anthony S. Atkins. 2017. An IoT Application for Inventory Management with a Self-Adaptive Decision Model. In *Proceedings - 2016 IEEE International Conference on Internet of Things; IEEE Green Computing and Communications; IEEE Cyber, Physical, and Social Computing; IEEE Smart Data, iThings-GreenCom-CPSCom-Smart Data 2016*. Institute of Electrical and Electronics Engineers Inc., 317–322. https://doi.org/10.1109/iThings-GreenCom-CPSCom-SmartData.2016.77

[152] T. Zhao, W. Zhang, H. Zhao, and Z. Jin. 2017. A Reinforcement Learning-Based Framework for the Generation and Evolution of Adaptation Rules. In *2017 IEEE International Conference on Autonomic Computing (ICAC)*. 103–112.

[153] Fanqin Zhou, Lei Feng, Peng Yu, Wenjing Li, Xiaoyu Que, and Luoming Meng. 2021. DRL-based low-latency content delivery for 6G massive vehicular IoT. *IEEE Internet of Things Journal* 9, 16 (2021), 14551–14562.

[154] Ji Zhou. 2015. Intelligent Manufacturing——Main Direction of" Made in China 2025". *China mechanical engineering* 26, 17 (2015), 2273.