Towards Distributed Edge-based Systems

Schahram Dustdar
Distributed Systems Group
TU Wien
Vienna, Austria

Ilir Murturi
Distributed Systems Group
TU Wien
Vienna, Austria

2020 IEEE Second International Conference on Cognitive Machine Intelligence (CogMI) | 978-1-7281-4144-2/20/$31.00 ©2020 IEEE | DOI: 10.1109/COGMI50398.2020.00021

***Abstract*—In the past few years, researchers from academia and industry stakeholders suggest adding more computational resources (i.e., storage, networking, and processing) closer to the end-users and IoT domain, respectively, at the edge of the network. Such computation entities perceived as *edge devices* aim to overcome high-latency issues between the cloud and the IoT domain. Thus, processing IoT data streams closer to the end-users and IoT domain can solve several operational challenges. Since then, a plethora of application-specific IoT systems are introduced, mainly hard-coded, inflexible, and limited extensibility for future changes. Additionally, most IoT systems maintain a centralized design to operate without considering the dynamic nature of edge networks. In this paper, we discuss some of the research issues, challenges, and potential solutions to enable: i) deploying edge functions on edge resources in a distributed manner and ii) deploying and scaling edge applications on-premises of Edge-Cloud infrastructure. Additionally, we discuss in detail the three-tier Edge-Cloud architecture. Finally, we introduce a conceptual framework that aims to enable easy configuration and deployment of edge-based systems on top of heterogeneous edge infrastructure and present our vision within a smart city scenario.**

***Index Terms*—Edge-Cloud Continuum, Edge-based Systems, Distributed Edge Functions**

I. Introduction

The Internet of Things (IoT) is becoming more prominent in our daily life and today's society overall. Many services in various domains such as Industrial Manufacturing, Healthcare, Lifestyle, Automotive, and Smart Building are built on the top of IoT technologies. At the same time, it is well accepted that a centralized architecture does not scale well regarding the enormous number of devices. Even though central computers' system infrastructure processing those data has improved by cloud computing technologies, satisfying IoT applications' stringent requirements has become challenging for a cloud-centric architecture. Sending continuously huge amounts of sensory data to the cloud results in high-latency than the expected IoT systems response. In sharp contrast to a fully distributed and decentralized architecture (e.g., peer-to-peer network), many IoT services need to maintain a partially centralized design to operate the service. The distributed and decentralized architecture promise to satisfy stringent requirements (i.e., high availability, performance, or privacy) of any contemporary IoT system deployed at the edge.

One prominent approach that has recently emerged is to combine the edge, fog, and cloud infrastructures to enable providing low-latency services [1]. Specifically, three-tier architecture shows a seamless opportunity for various applications (e.g., industrial, health, etc.) where low-latency, QoS, reliability, and scalability are their critical requirements. In essence, the three-tier architecture allows distributing a significant portion of decentralization by delegating functions from central servers to fog or edge devices [2].

Edge and fog are positioned as important architectural tiers between the cloud and the IoT domain. Edge devices are low-powered computers placed much closer to the sensors and mobile devices. Such devices enable processing the data streams generated by end-devices. In contrast, fog nodes (physical or virtual) are more powerful computation entities that also provide their computation and storage resources to the other computation entities or IoT resources. Generally speaking, both computation entities aim to reduce the latency (i.e., request or response) between application services and the IoT domain (i.e., sensors, mobile devices, etc.). Specifically, both computation entities can be utilized to process data streams pumped into an IoT system. Thus, such processing can be achieved by providing various analytic capabilities and various decision functions at the edge. For instance, a particular system component (e.g., scheduler) generates eligible deployment plans specifying where to process sensory data produced from an IoT resource (e.g., processing may occur at edge, fog, or cloud infrastructure). Throughout the paper, we refer to an IoT system deployed at the edge as an *edge-based system* and its components as *edge functions*.

The heterogeneity of IoT components is another significant aspect of the IoT era. Many IoT vendors provide different products not only across different layers but also for the same type of components. In essence, different products may have different operating systems, available support for programming languages, resource constraints, etc. In contrast to our expectations, each edge device's functions are hard-coded in most of the current implementations. This causes inflexibility and limited extensibility for future changes. This leads to a plethora of point-to-point solutions being developed based on proprietary protocols. For instance, any change being made to one IoT component leads to many possible changes in many other components or in the whole IoT system [3]. Thus, leading to a situation which resembles the state of software infrastructure in the age of enterprise computing before the emergence of Web services and Service Oriented Architectures and their respective middleware infrastructure such as Enterprise Service Bus. This intuition led us to design novel edge systems into a distributed, decentralized

978-1-7281-4144-2/20/$31.00 ©2020 IEEE
DOI 10.1109/CogMI50398.2020.00021

architecture that is not only scalable in numbers but also concerning its heterogeneity, expandability and enhancement in terms of edge functionalities.

In the past few years, researchers, computer scientists, and system engineers have been mostly focused on proposing multiple centralized techniques for scheduling, controlling, and monitoring edge applications (i.e., *IoT applications*) deployed at the edge. To overcome challenges with resource-constrained edge networks and to fully utilize available resources at the edge, edge applications (i.e., *service*) are divided into a set of independently deployable software components (i.e., *microservices*). Most of the resource management techniques are often deployed on cloud entities or statically placed on a powerful device such as a local server or a fog device residing at the edge [4], [5]. However, placing a set of complex functions on a single edge device is impractical and may produce high latencies than expected. Furthermore, shifting various system functions closer to the edge also requires dynamically placing them in the most suitable nodes. Besides that, edge networks' dynamic nature requires introducing new edge functions continuously and the continuous re-evaluation of placement decisions for the edge functions. Thus, edge-based systems require to cope with the environment's dynamicity and uncertainty, and support expandability with new edge functions.

To that end, this paper identifies some of the research issues and discusses challenges that are not yet fully investigated, such as i) deploying edge functions at the edge, and ii) deploying and scaling edge applications on-premises of the Edge-Cloud infrastructure. These derive mostly from the fact that many aspects of edge computing require further investigation to realize edge computing's underlying premise. Thus, to assist in this process, we propose a conceptual framework at the edge tier that aims to enable easy configuration, deployment, and operation of edge functions and applications on top of heterogeneous edge infrastructure. Furthermore, we advocate decentralization as the system can operate without a static entity for deploying applications at the edge. In essence, the system core functions are distributed over the edge devices available on edge. A self-adaptive mechanism determines the most suitable edge node to enable the system's core functions (i.e., the edge-based system's resource manager and control mechanism). Moreover, the system's core functions assists and provides information (i.e., infrastructure-specific or application-specific metrics) to other system functions running on other edge devices.

The rest of the paper is structured as follows. Section II gives an overview of Edge-Cloud architecture. In Section III, we describe a use case by emphasizing the benefits of the Edge-Cloud continuum and the need for distributed edge applications at the edge. In Section IV, we discuss some of the research challenges that the novel edge systems must conquer to run distributed functions on edge infrastructure. Section V, presents our conceptual framework to orchestrate edge applications in the Edge-Cloud infrastructure and the vision of the future smart cities. Finally, Section VI concludes the paper and outlines future work directions.

II. An Overview of Edge-Cloud Continuum

Edge computing is positioned as one important architectural tier in addition to Fog and Cloud. The model makes it easier to deploy distributed, latency-aware applications at the edge. Furthermore, it can be considered as paramount to systems including (but not limited to) IoT deployments and the cloud, providing data and control facilities to participating IoT devices. So far, several surveys have been conducted to explain the model of Edge computing and its challenges [6]. However, recent studies emphasize three-tier architecture as a useful way to enable more devices closer to the IoT domain [7]. To that end, the Edge-Cloud architecture [1] is split into three tiers: the cloud tier, fog tier, and edge tier (as illustrated in Figure 1). In this section, a comprehensive description of each tier is presented.

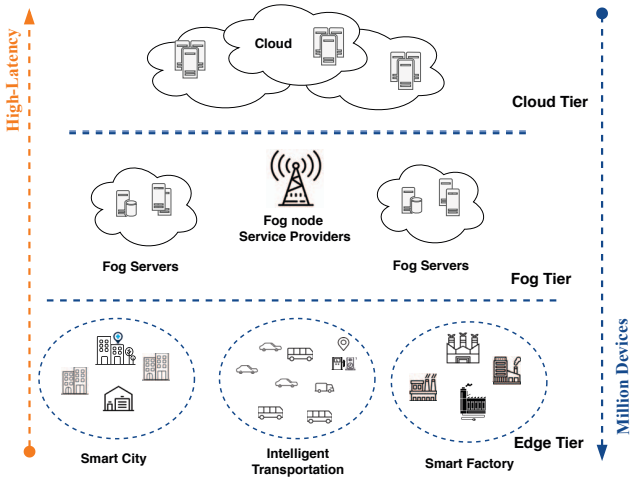

Fig. 1. An overview of Edge-Cloud architecture.

A. The Edge Tier

The edge tier represents the lowest tier of the three-tier architecture. In essence, this tier represents the closest available computation entities called *edge devices* to the IoT domain. As can be denoted in Figure 1, various domains such as smart city, intelligent transportation, or smart factories can benefit from available computation devices closer to the IoT domain. Essentially, this leads to having various edge networks formed for different contexts. Generally speaking, edge networks are highly dynamic, heterogeneous, and resource-constrained environments. Such environments are composed of a set of low-powered edge devices with various computation capabilities (e.g., smartphones, smartwatches, Raspberry Pis, etc.) and IoT resources (e.g., sensors, actuators, etc.) connected to them or available on their surroundings.

Edge devices provide their computation and storing capabilities to process the IoT data streams with very *low-latency* generated by IoT resources. Essentially, these devices allow data streams to be processed as close as possible to the data

sources and to handle the most significant network traffic that may occur. Resource-constrained edge devices, however, do not have the ability to process large volumes of data. As a consequence, processing such an amount of data on a single edge device may cause device overloading resulting in high-latency and poor overall performance. In order to address these challenges raised by resource-constrained devices, the distribution of processes (i.e., edge functions or edge application tasks) at the edge is a critical feature. Specifically, the spectrum of resources is increased by forming peer-to-peer edge networks, and the ability to spread processes between devices becomes feasible.

An *edge network* provides a seamless opportunity for placing edge functions and deploy various edge applications closer to the end-users. Such an approach results in creating more *autonomous environments* and less dependent on the external environments (i.e., cloud or fog). For instance, in a smart home, residents should control their devices and process data locally without depending on the cloud resources. This means that the edge-based system must provide functionalities enabling to achieve such a user goal. Furthermore, the user may deploy an edge application (e.g., smart health app) to provide services for processing residents' health data transmitted from their wearable devices. However, it is worth noting that even though extending resource scope through edge-to-edge collaboration provides many benefits, there are still many complex tasks that cannot be divided into sub-tasks and processed at this tier. Thus, in such situations, the edge devices must forward their processing to the upper tiers (i.e., fog or cloud). Moreover, in the general sense, the number of devices expected at this tier can reach millions.

B. The Fog Tier

The fog tier includes a collection of powerful (physical or virtual) devices responsible for managing, communicating, and exchanging resources between different edge networks. The core component of the fog tier is the *fog node*. This tier essentially represents the fog infrastructure (e.g., smart city network) where several fog devices are connected, offering various services, including computation and storage resources for edge networks and roaming end-devices in proximity. In sharp contrast to the edge networks, the fog infrastructure appears to be composed of stationary and powerful devices (typically managed and provided by Telco operators). In the tier, several functionalities can be deployed at this tier. For example, processing data streams, caching, device management, and privacy protection.

According to [8], fog computing provides similar service model implementation as in the traditional cloud computing model. Thus, the following types of service models can be implemented:

- **Software as a Service (SaaS)**. The fog service providers may host various services (i.e., similar to the cloud computing Software as a Service (SaaS)) on fog infrastructure. In essence, a fog service end-user (i.e., customer) may use providers' service running on fog infrastructure, specifically, on a cluster of federated nodes managed by the provider.
- **Platform as a Service (PaaS)**. The fog service providers allow the end-user to deploy their application onto the platforms (i.e., similar to the cloud computing Platform as a Service (PaaS)) on fog infrastructure. In essence, the end-user controls and configures their deployed applications and running environment. However, they do not control and manage the fog platform(s) and infrastructure.
- **Infrastructure as a Service (IaaS)**. The fog service providers allow the end-user to provision processing, storage, and other infrastructure-specific resources available on fog infrastructure (i.e., similar to the cloud computing Infrastructure as a Service (IaaS)). In essence, the customer can deploy and run various software, while the consumer may control and manage the storage, operating system, and deployed applications.

One can notice that edge and fog tiers provide almost similar features. Both paradigms foresee enabling more computation resources in proximity to the end-users and the IoT domain. However, the most significant difference between the two tiers is *administrative differences* and *responsibilities*. Furthermore, fog nodes (e.g., deployed in base stations) may provide their services for larger geographical areas. For instance, intelligent transportation systems may benefit from connecting and processing vehicle data in fog infrastructure. Nonetheless, both tiers provide low-latency services since the end-devices are closer to the source where the data is produced and consumed.

C. The Cloud Tier

The cloud tier provides "unlimited" computational and storage resources. This tier includes cloud servers that are deployed far away from the end devices and the IoT domain. In essence, cloud-based servers perform computer-intensive operations obtained from the architecture's lower tiers. These environments have advanced features for both service providers and service consumers. For instance, service consumers can configure their runtime environment, configure deployed applications, security control, and so on. Along these lines, the cloud computing utility has been seen as a critical component for designing, deploying, and executing IoT platforms that promise to meet the general population's daily needs.

Despite the numerous resources and advanced features provided, this paradigm faces increasing challenges in meeting new IoT applications' stringent requirements. Specifically, geographically distributed IoT devices with intensive data generation cannot efficiently utilize resources available in cloud environments [1]. On the one hand, transferring such intense and large amounts of data to a centralized cloud over Wide Area Networks (WAN) generates latencies. Additionally, service unavailability due to the non-persistent connectivity or eventually scheduled system maintenance (i.e., cloud-side) is another challenge that critical IoT systems may face during their runtime. On the other hand, real-time distributed apps require quick response time, high-availability, and increased

privacy, often not met by a centralized environment such as the cloud.

III. A Use Case of Edge-Cloud Systems

Understanding the current and future edge application requirements from various domains is critical for any contemporary edge-based systems' success. Thus, through careful analysis of real-world IoT scenarios from various domains, we show the importance of shifting various processes closer to the edge, dynamically placing them in the most suitable nodes, and the necessity to bring the elasticity features at the edge.

A. Smart Health Application

To motivate our subsequent discussion, we consider the well-known falling asleep problem and the smart health application. The falling asleep problem has been discussed previously in many research papers [9]–[11] and used as a motivating example for various proposed solutions. This problem generally belongs to the scenarios that are categorized into residents' comfort and convenience category. Suppose a resident has a smart health application running on his phone responsible for monitoring the resident's vital signs. The health application essentially collects data from a wearable ECG sensor attached to the human body through a smartwatch [12]. Specifically, the health application monitors the incoming data generated by the wearable device and reacts in real-time when a critical event occurs. Usually, for this type of application, a system architecture composed of a smartwatch and a smartphone represents a reasonable choice. However, since smartphones have limited battery energy, shifting data (i.e., health applications [13]) to the nearest edge device may be required when the battery level decreases at a critical level. Nonetheless, transferring data to the cloud directly consumes smartphone energy even faster.

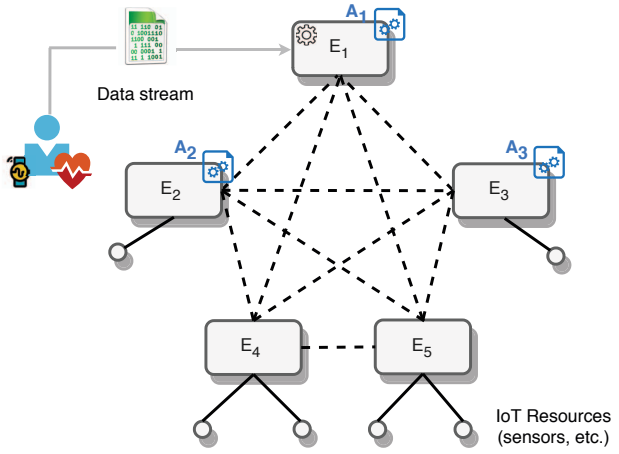

Fig. 2. Smart home health application.

In this scenario, it is assumed that a person in a living room, sitting on a sofa and watching TV, gradually falls asleep. Multiple edge devices deployed in rooms are responsible for changing surroundings accordingly such that the resident can get a comfortable living. In this case, it is the edge device's responsibility in the living room to change surroundings and ensure that the resident can sleep comfortably. These changes include turning off TV, lights, changing air conditioner settings, and locking the entrance door (e.g., see resource coordination [14]). Meanwhile, as the resident forgets plugging the smartphone into power, the battery nearly runs out. Therefore, the smartphone requests the nearest edge device to handle and process the generated data from wearable devices (as illustrated in Figure 2).

Our nearest edge device in the living room utilizes almost all its own processing capabilities to ensure that the resident sleeps comfortably (i.e., measuring room temperature, etc.) as well as runs a similar health application $\mathtt{A}_1$. Since the health application at the edge was idle, it requires more processing power and memory after the data is pumped into the system by the wearable device. Nevertheless, the edge device doesn't have enough computation capabilities to execute both applications concurrently. Thus, the health application faces higher latency to accept and process incoming data. Meanwhile, other edge devices in the smart home may remain idle. To overcome such a gap, the health application should create new instances (e.g., $\mathtt{A}_2$, and $\mathtt{A}_3$) and deploy on other edge devices or on-premises of Edge-Cloud infrastructure to meet application demands at runtime. Thus, it is evident that bringing the *elasticity features* at the edge is crucial.

IV. Distributed Edge Functions and Edge Applications Runtime

This section identifies current system architectures, challenges, and discusses potential solutions for deploying edge functions in a distributed and decentralized manner on edge. Then, we discuss edge application elastic requirements and potential solutions that enable application executing and scaling in the Edge-Cloud infrastructure. Finally, we discuss potential solutions for application runtime which is responsible for executing the configured edge application on an Edge device or in the Cloud.

A. Distributed Edge Functions on Edge

Researchers and computer scientist in edge and fog computing have been mostly focused on proposing multiple techniques for resource allocation problem aiming to minimize various trade-offs such as latency, bandwidth, energy consumption, or maximizing the utilization of resources at the edge. In general, edge applications are deployed according to the following models [15]: i) *everything in the cloud*, ii) *everything in the edge*, and iii) *hybrid edge-cloud model*. Essentially, allocation techniques may deploy software components entirely on a single environment (e.g., cloud or edge) or components are deployed and executed in both cloud and edge [16].

As we explore new edge systems, edge applications, and the heterogeneous edge networks, distributing system compo-

nents and application processes among various computation entities becomes increasingly apparent [17]. For instance, an edge system can comprise a set of dependent functions (i.e., controlling module, scheduler, resource manager, etc.) that can be deployed individually on multiple edge devices. Thus, we analyze and discuss the pros and cons of three main edge system architectures at the edge: i) *centralized*, ii) *distributed*, and iii) *decentralized*.

In the centralized architecture, a single edge device acts as a master device responsible for monitoring and distributing tasks among other available computation entities in the Edge-Cloud continuum. In essence, the master device includes a set of functionalities placed statically on a single edge device. Generally speaking, edge systems with centralizing architecture may be feasible in the context of small and non-dynamic edge networks (e.g., smart homes). However, a centralized architecture's main challenges are that it does not scale easily, the central node requires to run on a resource-constrained device, and the dynamic nature of the edge network makes it impractical. In sharp contrast, the distributed architecture treats all edge devices equally in terms of their system responsibilities. In an edge network without a master device, edge devices in proximity are coordinated and agreed upon to some SLA agreement to execute system functions. Essentially, such coordination can be achieved by using consensus-based algorithms. In practice, distributed solutions may face latency issues when nodes need to find consensus to distribute functions. Moreover, the main challenge is the small number of nodes considered in the network topology. Along these lines, regardless of the approach, both solutions may have plenty of advantages in various IoT scenarios.

In the decentralized architecture, the master device functionalities may be placed *statically* (i.e., at design time) or *dynamically* (i.e., with self-adaptive capabilities). However, the dynamic nature of edge networks requires the continuous re-evaluation of placement decisions for such functionalities. Thus, a feasible approach to overcome such a gap is placing such functionalities dynamically at the edge. To that end, a possible solution is considering election based algorithms. For instance, through initiating an *election* between edge devices, the most suitable node (e.g., in terms of computation power) is elected as a master device. Specifically, nodes in the network arrive at the same outcome independently by sharing election results. Such a solution denotes a very decentralized approach to elect the master device automatically. Moreover, dynamically placing such functionalities overcomes the challenges introduced by mobile devices and possible device failures in edge networks. Besides that, novel techniques are needed to address the barriers to deploy such functions on resource-constrained edge devices. For instance, the elected master device must delegate various functionalities to other nodes to handle the computation and network overheads.

To overcome such aforementioned challenges, a possible solution is to introduce new coordinators (i.e., superpeers [18]) in the system. A coordinator manages a set of nodes perceived as *cluster*. As the network size expands, new coordinators are introduced to the system as well as new clusters are formed. In particular, coordinators provide almost similar functionalities as the master device. The master device becomes a supervising node responsible for managing coordinators, monitoring, and distributing edge functions as well as edge applications among coordinators. For instance, a user can submit a request to the master device for edge application deployment. The master device examines his/her geographically location and asks the closest coordinators to determine if their cluster can meet the edge application requirements. The coordinator who fulfills application demands gets the application and distributes it to the cluster's edge devices.

B. Elasticity as an Edge Function

Future edge-based systems for the described Edge-Cloud architecture need to hide their operational complexity from application developers as well as from the end-users. Specifically, application developers should not have to deal with the heterogeneity, dynamicity, and expandability of the edge network setting. Developers should be able to express the context in which edge application components are allowed to run, their system dependencies, and their requirements (e.g., QoS, elastic requirements, etc.) in a high-level way [19]. For instance, an application developer may specify the elastic requirements of an edge application (e.g., health application). To interpret these requirements and enforce required operations in order to keep the desired service quality, we require the *elasticity function* at the edge. In essence, the elasticity function provides a runtime mechanism that interprets elastic requirements given by the developer. Such function is deployed and enabled automatically at the edge-based system after the user deploys the mentioned edge application.

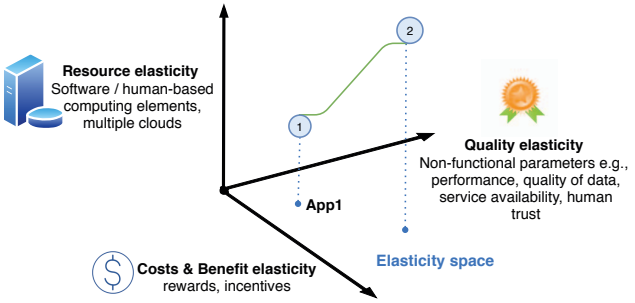

Fig. 3. Elasticity space at the Edge-Cloud architecture.

Bringing elasticity properties to the edge is crucial for the future of edge applications. In the Edge-Cloud architecture, elasticity targets not just resources and their capacity to scale, but also their relations with the different types of costs and quality (as illustrated in Figure 5) [19]. In this context, various stakeholders may be involved in specifying elastic requirements. For instance, the developer could specify that the latency between application components must not reach 20ms without carrying how many resources should be

used to achieve the desired state. The edge network provider could specify its resource utilization schema, for example, that when overall utilization at the edge is higher than 90%, it enables scaling out the edge. As a potential candidates for defining these requirements are *SYBL: Simple Yet Beautiful Language* [20] and *SLOC: Service Level Objectives for Next Generation Cloud Computing* [21]. In this paper, we consider the SYBL language and its runtime mechanism for controlling elasticity in edge applications.

SYBL enables the user to specify elastic requirements at different granularities and enables applications to scale in elasticity space (*cost*, *resources*, and *quality*). In particular, SYBL allows the user to: i) specify *monitoring* metrics required to be monitored, ii) specifying *constraints* through which the monitored metrics are allowed to oscillate, iii) specifying *strategies* representing actions to be taken when a constraint is violated, and iv) specifying constraint *priorities* required to be executed first. Additionally, SYBL allows the developer and the user to achieve various trade-offs, such as specifying demands on the relation between cost, resources, and quality (as illustrated in Figure 5). To simplify developing such specifications, a user-centric interface can be integrated into the development platform through which developers and end-users describe their edge applications for deployment in the Edge-Cloud infrastructure.

C. Runtime Platform for Edge Applications

Executing edge application software components in heterogeneous environments is another significant challenge. An essential requirement is to enable executing components on a homogeneous runtime platform that follows the "*run once, run anywhere*" model. Furthermore, the platform must enable our system components (i.e., resource manager and control mechanism) to access, govern, and orchestrate edge applications. To that end, as a potential candidate for executing edge applications, we consider Docker[1] or Java-based OSGi[2]. In essence, the runtime mechanism is responsible for executing the configured edge application on an edge device, fog device, or in the cloud.

V. Towards Distributed Edge-based Systems

In our conception, an edge network comprises a set of geographically distributed edge networks, as introduced in [1], [22]. A concrete example is a smart city edge network, as illustrated in Figure 4. Various edge networks called *edge neighborhoods* can be formed and join the city's edge network backbone. Generally speaking, such edge neighborhoods can be formed in smart buildings, smart factories, hospitals, etc.

Edge neighborhoods offer a seamless opportunity for the end-users to customize their environments with various services to support their daily activities and improve their living comfort. In essence, an edge neighborhood is formed by connecting multiple edge devices in proximity to each other. The end-users may deploy various edge applications and customize

[1]Docker, https://www.docker.com/
[2]OSGi, https://www.osgi.org/

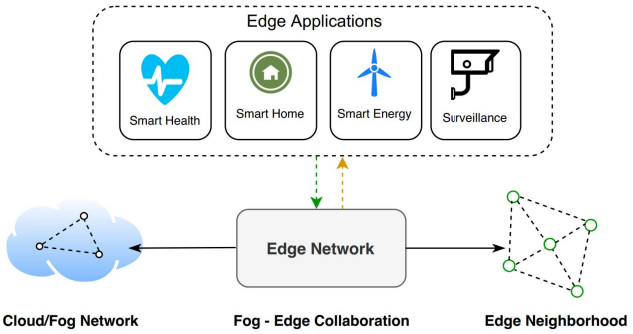

Fig. 4. An overview of Edge-Cloud infrastructure.

their edge neighborhoods based on their needs. For instance, in a smart home, residents may deploy an edge application that provides them a service to back up their smartphones automatically when they're home. Or in a smart building (e.g., museum environments), the system administrator may deploy an edge application that provides a service for visitors to interact with their surrounding objects through virtual reality (VR) [23].

Even though the computation scope is expanded with forming edge neighborhoods, one should note that edge neighborhoods cannot always provide enough resources to execute edge applications (e.g., tasks regarding image processing, etc.). In essence, these environments are resource-constrained and with limited resources. Specifically, resource-demands for a particular running edge application or task may change over time. For instance, the backup service at a smart home cannot provide the desired service quality when the service is used simultaneously by more than three residents. Consequently, this may cause poor overall performance and higher latency than the expected response time in edge applications. To that end, we require a lightweight framework that can be easily deployed on low-powered edge devices and support extending its system functionalities with new edge functions.

To fill this technological gap, we describe the role of each tier involved in the platform design. The cloud tier provides several types of services and contains edge application models that users can download to their edge neighborhoods. An edge application model essentially describes in detail application components and their resource requirements (i.e., hardware requirements and elastic requirements) from the developer perspective. Furthermore, this tier assists edge neighborhoods in identifying the closest fog devices (e.g., location coordinates) as well as provides storing and computation resources. The particular fog services enable for connecting and authenticating newly edge neighborhoods (i.e., authenticated to the city's edge network backbone). The fog tier may provide several types of services (as mentioned in Section II). For instance, a fog-based blockchain platform may provide service to edge neighborhoods to rent their resources.

To achieve the aforementioned objectives, we introduce a conceptual framework at the edge tier that enables easy con-

figuration, deployment, and operation of edge functions and edge applications on top of heterogeneous edge infrastructure (as presented in Figure 5). The proposed framework aims to equip edge devices with core functions such as i) self-adaptive mechanism, ii) application manager and scheduler, iii) edge functions and resource manager, iv) edge monitoring agent, and v) gateway module. We propose that each edge device needs to have such a set of functionalities.

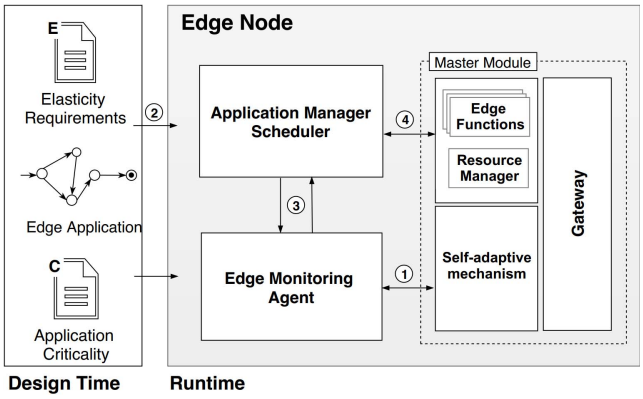

Fig. 5. An overview of a conceptual edge framework.

In edge networks, a critical task (1) is to determine which node should take the responsibility to act as a *resource manager* and *control mechanism*, respectively, to have enabled the *master module* as presented in Figure 5. The master module essentially provides a set of unique functionalities responsible for monitoring available resources (i.e., IoT resources and edge devices) at the edge neighborhood. The resource manager provides infrastructure-specific (i.e., current resource utilization and availability) to the application manager and scheduler. The self-adaptive mechanism provides functionalities for determining which node should have enabled the master module. For this reason, we can consider our efficient approach in [24] to determine the master module in a decentralized manner in dynamic edge neighborhoods. The proposed approach dynamically adapts to the network changes and activates the master module on the most suited edge device in terms of computation and network capabilities.

After the master module is determined, the user may automatically download various edge functions at the edge. In essence, some applications in order to be executed at the edge may require additional and specific system functionalities. For instance, resource coordination service [14] at the edge requires specific functions (e.g., SMT [25]) to generate valid coordination plans that will enable the service to achieve its goals. Or, the edge neighborhood can be expanded with new IoT resources communicating with specific protocols. Thus, the edge-based system needs to be extended with new functions to support IoT resource operation. Furthermore, the user may download an edge application from the cloud and deploy it to the edge neighborhood. In essence, each edge device runs an application manager and a scheduler module (2). This means that the user may request each edge device to deploy an edge application. The edge device that generates the deployment plans also becomes the orchestrator for that edge application. This means that the edge device controls the elasticity requirements and enforce various scaling operation when it is required. Nevertheless, before the deployment, the owner may extend the following user-perspective requirements: i) elastic requirements and ii) application criticality.

The edge application critical requirements determine application dependencies (i.e., on edge functions) specified by the application developer. For instance, to interpret the SYBL elastic requirements, the application developer specifies that the elasticity function should be deployed and available on the edge neighborhood. Such a function can be downloaded after user approval. Further, the edge application critical requirements can be extended by the user with user-specific application configuration settings. For instance, the user may specify where the data is allowed to be processed (i.e., edge, fog, or cloud). Essentially, it determines where the edge application is allowed to run. For instance, the user may specify one of the following deployment models described in Section IV. Furthermore, the user may extend elastic requirements with new user-requirements. This means that the user may specify that to keep the desired service quality all the time, the edge application can scale to the cloud (i.e., paying for the cost of using it). Nevertheless, some users might be unwilling to process their private data at the cloud or edge network (i.e., publicly available resource at the edge).

Along these lines, the scheduler considers all the above-mentioned requirements, gets the infrastructure state from the resource manager, and then generates an eligible deployment plan (if exists) (3-4). Thus, it determines where the edge application components and data need to be processed such that their requirements are fulfilled. To generate such deployments plans in the Edge-Cloud infrastructure, we can consider the approach in [26]. Furthermore, the monitoring agents deployed at each edge device periodically updates the resource manager with infrastructure-specific metrics (i.e., CPU, storage, etc.) and application-specific metrics (i.e., non-functional parameters). The application-specific metrics are real-time values that the elasticity function uses to control whether the edge applications' elasticity aligns with the elastic requirements defined.

The gateway module implements all necessary mechanisms to enable communication between edge devices and connect them with external networks (i.e., cloud and fog). Specifically, it enables edge devices to connect in a peer-to-peer manner and automatically discover edge neighborhood resources (e.g., IoT resources, etc.). Furthermore, the edge neighborhood owner configures the gateway mechanism on how to connect to the cloud. After the successful configuration, the cloud returns a list of closest fog devices (i.e., fog services) to the edge neighborhood. The edge neighborhood is automatically registered in the edge network (e.g., smart city's edge network) with a unique neighborhood ID. Afterward, the owner may download and install specific edge functions, e.g., to enable

resource renting such as computation power or storage along with the payment term. For instance, blockchain platforms (e.g., distributed storage[3,4] or distributed computation power[5]) deployed on fog infrastructure enable edge neighborhood owners to earn money by renting their unused storage and idle CPU cycles to those in need. The premise is simple: the edge owner installs a specific edge function (e.g., blockchain function) on its own edge network and configures it to connect to the fog's blockchain platform. In essence, the owner list resources along with payment terms and get paid when other users in the platform utilize them (i.e., through smart contracts). The mentioned edge function synchronizes continuously such information with the fog's blockchain platform.

VI. Conclusion

Edge computing is considered a fundamental enabler for the proliferation of the Internet of Things (IoT). Compute and storage resources placed at the edge of the network are used to bridge the gap between the Cloud and the IoT domain. Such computation entities perceived as *edge devices* can be used to analyze high-volume IoT data streams and, at the same time, provide control facilities to participating IoT devices. In essence, a wide range of available resources at the edge provides a seamless opportunity to deploy various edge applications providing their services with low-latency. However, for edge applications to operate properly, different edge functions are needed at the edge (e.g., elasticity function).

To that end, in this paper, we discussed some of the research issues, challenges, and potential solutions to enable: i) deploying edge functions on edge resources in a distributed manner and ii) deploying and scaling edge applications on-premises of Edge-Cloud infrastructure. However, with the introduced conceptual framework, the edge-based systems deployed at the edge can be easily extended with new edge functions. Thus, distributing edge functions at the edge paves the way for utilizing available resources, leading to the promised high-quality and low-latency edge-based system solutions. Nevertheless, a challenging task remains to provide a full solution stack for edge applications that are dynamically distributed, elastic, resilient, and executed natively in the Edge–Cloud continuum.

Acknowledgment

Research partially supported by the TUW Research Cluster Smart CT and it has received funding from the EU's Horizon 2020 Research and Innovation Programme under grant agreement No. 871525. EU web site for Fog Protect: https://fogprotect.eu/

References

[1] Schahram Dustdar, Cosmin Avasalcai, and Ilir Murturi. Edge and fog computing: Vision and research challenges. In *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*, pages 96–9609. IEEE, 2019.

[2] Weisong Shi and Schahram Dustdar. The promise of edge computing. *Computer*, 49(5):78–81, 2016.

[3] Xhevahir Bajrami and Ilir Murturi. An efficient approach to monitoring environmental conditions using a wireless sensor network and nodemcu. *e & i Elektrotechnik und Informationstechnik*, 135(3):294–301, 2018.

[4] Olena Skarlat, Vasileios Karagiannis, Thomas Rausch, Kevin Bachmann, and Stefan Schulte. A framework for optimization, service placement, and runtime operation in the fog. In *2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC)*, pages 164–173. IEEE, 2018.

[5] Shanhe Yi, Zijiang Hao, Qingyang Zhang, Quan Zhang, Weisong Shi, and Qun Li. Lavea: Latency-aware video analytics on edge computing platform. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, page 15. ACM, 2017.

[6] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge computing: Vision and challenges. *IEEE internet of things journal*, 3(5):637–646, 2016.

[7] Cosmin Avasalcai, Ilir Murturi, and Schahram Dustdar. Edge and fog: A survey, use cases, and future challenges. *Fog Computing: Theory and Practice*, pages 43–65, 2020.

[8] Michaela Iorga, Larry Feldman, Robert Barton, Michael J Martin, Nedim S Goren, and Charif Mahmoudi. Fog computing conceptual model. Technical report, 2018.

[9] Yuvraj Sahni, Jiannong Cao, Shigeng Zhang, and Lei Yang. Edge mesh: A new paradigm to enable distributed intelligence in internet of things. *IEEE access*, 5:16441–16458, 2017.

[10] Shuo Feng, Peyman Setoodeh, and Simon Haykin. Smart home: Cognitive interactive people-centric internet of things. *IEEE Communications Magazine*, 55(2):34–39, 2017.

[11] Stephen Makonin, Lyn Bartram, and Fred Popowich. A smarter smart home: Case studies of ambient intelligence. *IEEE Pervasive Computing*, 12(1):58–66, 2013.

[12] Marjan Gusev and Schahram Dustdar. Going back to the roots—the evolution of edge computing, an iot perspective. *IEEE Internet Computing*, 22(2):5–15, 2018.

[13] Pasquale Pace, Gianluca Aloi, Raffaele Gravina, Giuseppe Caliciuri, Giancarlo Fortino, and Antonio Liotta. An edge-based architecture to support efficient applications for healthcare industry 4.0. *IEEE Transactions on Industrial Informatics*, 15(1):481–489, 2019.

[14] Christos Tsigkanos, Ilir Murturi, and Schahram Dustdar. Dependable resource coordination on the edge at runtime. *Proceedings of the IEEE*, 107(8):1520–1536, 2019.

[15] Majid Ashouri, Paul Davidsson, and Romina Spalazzese. Cloud, edge, or both? towards decision support for designing iot applications. In *2018 Fifth International Conference on Internet of Things: Systems, Management and Security*, pages 155–162. IEEE, 2018.

[16] Massimo Villari, Maria Fazio, Schahram Dustdar, Omer Rana, and Rajiv Ranjan. Osmotic computing: A new paradigm for edge/cloud integration. *IEEE Cloud Computing*, 3(6):76–83, 2016.

[17] Fahed Alkhabbas, Ilir Murturi, Romina Spalazzese, Paul Davidsson, and Schahram Dustdar. A goal-driven approach for deploying self-adaptive iot systems. In *2020 IEEE International Conference on Software Architecture (ICSA)*, pages 146–156. IEEE, 2020.

[18] Gian Paolo Jesi, Alberto Montresor, and Ozalp Babaoglu. Proximity-aware superpeer overlay topologies. In *IEEE International Workshop on Self-Managed Networks, Systems, and Services*, pages 43–57. Springer, 2006.

[19] Schahram Dustdar, Yike Guo, Benjamin Satzger, and Hong-Linh Truong. Principles of elastic processes. *IEEE Internet Computing*, 15(5):66–71, 2011.

[20] Georgiana Copil, Daniel Moldovan, Hong-Linh Truong, and Schahram Dustdar. Sybl: An extensible language for controlling elasticity in cloud applications. In *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, pages 112–119. IEEE, 2013.

[21] Stefan Nastic, Andrea Morichetta, Thomas Pusztai, Schahram Dustdar, Xiaoning Ding, Deepak Vij, and Ying Xiong. Sloc: Service level objectives for next generation cloud computing. *IEEE Internet Computing*, 24(3):39–50, 2020.

[22] Ilir Murturi, Cosmin Avasalcai, Christos Tsigkanos, and Schahram Dustdar. Edge-to-edge resource discovery using metadata replication. In *2019 IEEE 3rd International Conference on Fog and Edge Computing (ICFEC)*, pages 1–6. IEEE, 2019.

[3] Storj (Distributed Storage), https://storj.io

[4] Filecoin (Distributed Storage), https://filecoin.io

[5] iExec (Distributed Computation), https://iex.ec/

[23] Maria Shehade and Theopisti Stylianou-Lambert. Virtual reality in museums: Exploring the experiences of museum professionals. *Applied Sciences*, 10(11):4031, 2020.

[24] Ilir Murturi, Mohammadreza Barzegaran, and Schahram Dustdar. A decentralized approach for determining configurator placement in dynamic edge networks. In *2020 2nd IEEE International Conference on Cognitive Machine Intelligence*, pages 1–10. IEEE, 2020.

[25] Clark Barrett, Christopher L Conway, Morgan Deters, Liana Hadarean, Dejan Jovanović, Tim King, Andrew Reynolds, and Cesare Tinelli. Cvc4. In *International Conference on Computer Aided Verification*, pages 171–177. Springer, 2011.

[26] Antonio Brogi and Stefano Forti. Qos-aware deployment of iot applications through the fog. *IEEE Internet of Things Journal*, 4(5):1185–1192, 2017.