ASRE – Towards Application-specific Resource Ensembles across Edges and Clouds

Hong-Linh Truong
Department of Computer Science, Aalto University, Finland
linh.truong@aalto.fi

***Abstract*—We research a new abstraction for resources for applications, called Application-specific Resource Ensembles (ASREs), across edge and cloud infrastructures. ASRE encapsulates diverse types of high-level resources, coupled with management APIs, that is provided for application-specific contexts. ASRE is designed with integrated mechanisms to manage its multi-dimensional quality through monitoring and control techniques. Instances of ASRE can be provisioned on-demand, with the elasticity and resilience capabilities, thus help to simplify the resource management in edge-cloud continuum.**

I. Introduction

A very typical scenario today is to develop an edge-cloud application, which analyzes data from various IoT devices in the edge and the cloud or enables the movement of IoT realtime and logs data from the edge to the cloud [1]. For such a scenario, the developer would acquire message broker services (e.g., MQTT), data transfer services (e.g., Apache Nifi instance), flow engines (e.g., Node-RED engine), streaming data analytics (e.g, Apache Flink), cloud storage (e.g., Google Storage), batch processing service (e.g., Hadoop/Spark) and machine learning service (e.g., a regression model deployed as a service using Seldon [2]). In many cases, as observed in the related work and software products, this would be achieved by (i) acquiring various containers and virtual machines (VMs) from edge and cloud services, (ii) deploying suitable software in these containers/VMs [3] to create resources as services, and then (iii) scaling, controlling and managing these resources[1]. However, given that such software services can be automatically provided by providers, the developer would like to manage only high-level resources. Figure 1 shows the applications and their resource providers mentioned above.

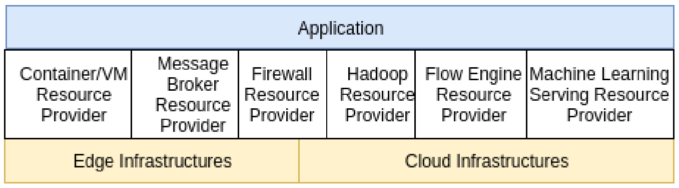
Application
Container/VM Resource Provider
Message Broker Resource Provider
Firewall Resource Provider
Hadoop Resource Provider
Flow Engine Resource Provider
Machine Learning Serving Resource Provider
Edge Infrastructures
Cloud Infrastructures

Fig. 1. A single application utilizing high-level cross-layered resources in edge and cloud infrastructures

Individual providers can return resources to the developer, but currently the developer lacks tools to manage such diverse types of resources within an application context; these resources are not at the same software layer and are spanned across different infrastructures. To solve this problem, we introduce application-specific resource ensembles (ASREs) as a high-level abstraction of different service capabilities for, e.g., function execution, data file transfer, message delivery, database, machine learning serving, and security protection. ASRE runtime will coordinate resources to deliver these capabilities to applications. Crucial features, like provisioning, security, monitoring and reliability engineering, will be associated with ASRE runtime for managing ASREs lifecycle, and an ASRE Management Service will provision elastic and resilient ASREs for application-specific context.

In this paper, we will present some key aspects of ASREs. We will explain the key requirements for ASREs (Section II) that ASREs must combine various types of high-level edge and clouds resources, also IoT data sources and network functions, for application-specific needs. We can use "resource slices" [4] to include various types of resources across different infrastructures to a virtual one. However, resource slices are not enough for our ASRE concept. Therefore, in our initial design and implementation (Section III and Section IV), ASREs must establish their resource slices equipped with elastic and resilient automation mechanisms. Furthermore, we need to leverage automation mechanisms for ASREs to automate the management of disparate resources. Thus, an ASRE Management Service interfaces to a variety of service providers in edge-cloud infrastructures and interacts with them for provisioning resources for specific application contexts, while allowing the developer to focus on utilizing resources from best providers, given application requirements. We will give examples of ASRE scenarios (Section V), discuss related work (Section VI), and outline our future plan (Section VII).

II. Key Requirements for ASRE

The requirements for ASREs are from the need to acquire resources across systems and layers and then to manage them as a whole under a high-level virtual entity in edge-cloud continuum. The resource management for edge-cloud application developers needs to support a high-level abstraction of resources for their applications, while utilizing resources from different providers. Thus, the developers can leverage the best offered quality, reliability, security and functionality for resources from existing providers.

ASRE as a resource as a service: Resources are defined as high-level service functionalities that are required by applica-

[1]Of course, for certain services we can request suitable resources, like MQTT broker or Hadoop service, in a single step.

tions. At the lowest level, resources in ASREs can be execution environments based on VMs and containers. High-level resources are, e.g., container orchestration services, process/-workflow engines, message brokers, network functions (e.g. firewall), secret-as-a-service for authentication and identity management, data transformation services, file-based storage services, database services, and machine learning models serving as services. Such resources are provisioned either by different providers or by combining resources from these providers with runtime software deployment from existing software artifacts.

Monitoring capabilities: Monitoring capabilities must be applied to ASRE as a whole. Currently, individual resources can be monitored in different ways and a lot of monitoring data can be collected. Typical monitoring data are CPU/memory usage, number of containers and processes, number of service requests, response time, etc. However, they might not indicate the behavior of ASREs as a whole. For example, it does not make sense in looking at memory and CPU usages of all individual resources when they do not shed light on why the whole ASRE acts poorly. For monitoring ASREs, it is possible that we have to deploy additional monitoring tools for ASREs to work with existing monitoring of individual providers and resources. Furthermore, we need to understand the dependencies among individual resources within an ASRE instance in order to evaluate the service quality of the ASRE instance, such as its reliability, availability and cost.

Elasticity and Resilience: For resilience, we need to change resources or recover resource failures at runtime. For this purpose, we must leverage dynamic provisioning (request new resources for failed ones) of resources in combination with resource reconfiguration at different levels, without interrupting the execution of applications. For elasticity support, typically operations are to add and remove resources accordingly as well to change the non-functional capabilities of resources. In addition, functional capabilities can be added, e.g., a new data transformation might be deployed, or removed, e.g., an IoT protocol bridge can be shutdown. This means that the elasticity will be carried out at two levels: the ASRE level for dealing with functional capabilities of the ASRE (cross-resource and cross-layer elasticity) and the resource level for internal elasticity of non-functional capabilities.

III. ASRE Models

From the service viewpoint, ASREs should be provisioned to the developer through an ASRE Management Service; different ASRE instances have different capabilities for different application contexts. This leads to a generic view of ASREs and an ASRE Management Service, shown in Figure 2. We will briefly discuss these issues in the following:

ASRE's Resource Slice: A *resource slice* captures information about resources abstracted by ASRE for a specific application. Since ASREs support different types of resources, we devise a high-level information model to encapsulate low-level resource information. Generally, resources in an ASRE are described as in a ”resource slice” in a rich metadata model.

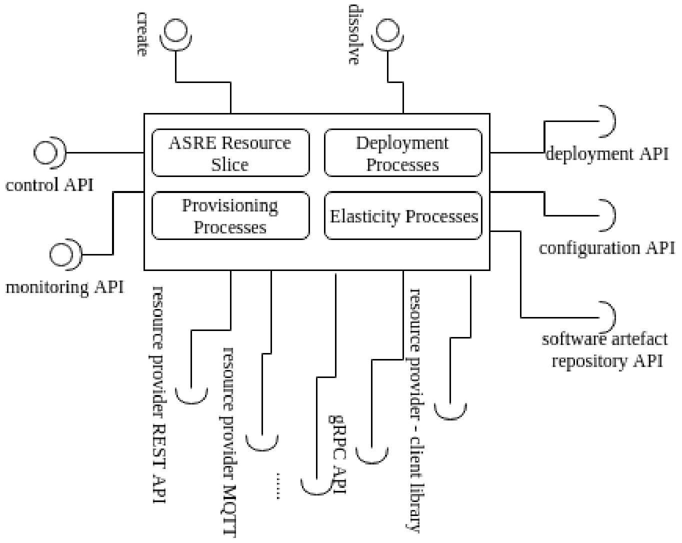

Fig. 2. Simplified view of an ASRE Management Service

Various types of metadata from providers and their types of resources are captured. The data model for ASRE's resource slice is an extension of the work in [4]. Metadata types will be captured by interfacing with service providers and service discovery services (see Section IV). An ASRE's resource slice includes several resources. Such resources are connected through `Connectivity` which specifies in `AccessPoint` and is implemented with different protocols. From the ASRE viewpoint, each resource has classes of `MonitoringPoint`, `ControlPoint`, and `DeploymentPoint` for monitoring, control and deployment features, respectively. Each class has multiple points mapping to concrete APIs and has different types of metadata. Depending on types of resources, a resource offers its business functions via APIs under `ResourceBusinessFunction`. For example:

- for a sensor resource, providing sensor data is its business function and a subset of APIs for obtaining sensor data could be defined under `DataPoint`.
- for a machine learning service resource: providing inference capabilities utilizing different machine learning models is its business function. A subset of APIs for obtaining inferences through different machine learning models could be defined under `InferencePoint`.

There is no standard for metadata so that we cannot automatically map APIs into these points. But using service discovery and metadata, we can map them through software integration activities (see Section IV). From the resource management viewpoint, constructing ASRE requires context-specific requirements of applications. Therefore, creating suitable ASRE instances requires knowledge about providers and resources.

Monitoring API: Monitoring API is designed for the whole ASRE and its constituting resources. At the ASRE level, monitoring API will allow us to deal with existing ASRE instances. Operations applied to ASREs, such as, `creation`, `change` and `delete`, will be logged and structured information of resources can be obtained. ASRE specific metrics will be monitored. Key APIs are shown in Table I.

API	Description
/{asre}/get	obtain the structure of the ASRE
/{asre}/list	list all existing ASREs
/{asre}/logs	obtain logs of ASRE operations
/{asre}/metrics/list	list ASRE metrics
/{asre}/metricSeries	obtain time series of ASRE metrics
/{asre}/pushMetrics	push metrics to external monitoring systems

TABLE I
API FOR MONITORING ASRE RESOURCES

At the resource level, monitoring data is based on the resource's `MonitoringPoint`. To obtain the resource motoring data, we follow the best practices that current are implemented in most resource-as-a-service: (i) the exporter model, such as like the model of Prometheus exporter[2], in which exporters are configured to obtain monitoring data from the resource and (ii) the instrumentation model, in which the resource pushes monitoring data to the monitoring system.

Control API: Control APIs are used to create, modify and remove ASREs based on application requests. For creating an ASRE, we will need an initial set of resource descriptions, which indicate resource types and their connectivity. Based on resource and provider discovery, an ASRE Management Service internally will acquire and/or perform artifact deployment to create resource slices. Main APIs are described in Table II.

API	Description
/{asre}/createResource	creates an ASRE with a specification of resource slice
/{asre}/deleteResources	deletes resources of an ASRE
/{asre}/delete	deletes the whole ASRE
/{asre}/updateResource	updates resources

TABLE II
API FOR CONTROLLING ASRE RESOURCES

Deployment API: the deployment API can be used to deploy existing ASREs, based on saved ASRE descriptions, or new resources in selected providers or atop other resources:

- deploy saved ASREs: as long as ASRE descriptions are saved, we can reuse them when an application needs to acquire similar ASREs. In this case, the API allows the application to select and create new instances of ASREs.
- deploy new resources: this situation happens when we need to deploy a software artefact into an existing resource to create a new (high-level) resource. One example is that ASRE can ask a resource provider, which is a machine learning (ML) serving platform, to deploy a ML model to create a ML-as-a-service resource.

When controlling ASREs, Control API can invoke Deployment API. This feature would require complex operation processes implemented within an ASRE Management Service.

IV. ASRE MANAGEMENT

Resources and Provider Discovery: Being able to obtain resources from different providers requires an ASRE Management Service to discover available providers and resources. This is the subject of a huge body of research and tools so we just reply on existing mechanisms for resources discovery:

[2]https://prometheus.io/docs/instrumenting/exporters/

- for resource providers without discovery support: we capture enough information about the service provider, especially the APIs and libraries for acquiring resources, monitoring, and control.
- for resource providers with discovery support: we interface with discovery systemsand capture metadata about services. This case usually happens for high-level services, VMs and containers managed by a single large-scale provider.

In both situations, while resource information can be obtained automatic, dealing with semantics of the provider and resource information still requires integration work.

Resource Acquisition: For acquiring resources we must consider various real-world integration situations:

- resource acquisition through known APIs and protocols, such as REST and MQTT: in this case, resources can be obtained through a single step of calling corresponding APIs and will be ready for ASREs.
- resource acquisition through known client libraries: client libraries are very popular with existing cloud providers. Such libraries have to be integrated into an ASRE Management Service through software integration tasks.
- resource acquisition through software deployment: required resources are not available but an ASRE Management Service can acquire infrastructural containers and VMs and deploy suitable services to create the required resources. For example, ASRE can request a container from Kubernetes and deploy MQTT into the container.

As discussed before, not all tasks can be done automatic without certain manual integration. `Adaptors` can be used as a component to interact with resource providers and resources. We develop various adaptors to interface to existing types of providers in order to acquire resources. Shown in Figure 3, the way how an adaptor works with resource providers and resources is similar to how the developer would manually program APIs for resource acquisition. However, using a set of adaptors (which can be updated, changed, and reused), we can bring various types of resources for ASRE. Given that resources can be simple or complex (e.g., a Hadoop cluster is a complex resource), ASRE can also interact with resources directly. To implement `Adaptors` we leverage containerized microservices and serverless functions.

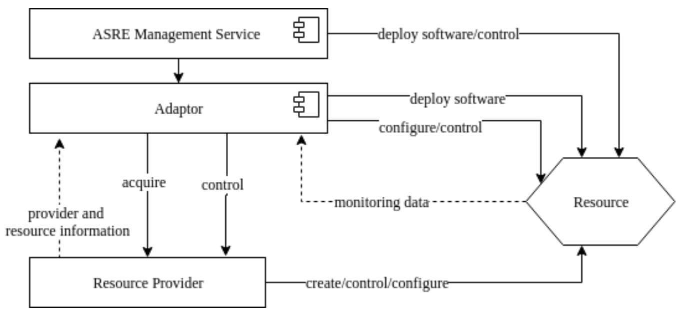

Fig. 3. Resource acquisition for ASREs: both Resource Provider and Resource must offer well-defined APIs (not necessary REST-based APIs)

Management States of Resources: We distinguish two situation in managing states of resource providers and resources:

- monitoring capabilities offered by resource providers and resources: resource providers and resources usually have their own monitoring capabilities. In this case, we just leverage the monitoring capabilities by subscribing APIs. Partially shown in Figure 3, states can be received via monitoring API. When states are published through external monitoring systems, such as Prometheus, we will receive states from these systems.
- monitoring capabilities instrumented: this is for resources whose components are partially deployed by an ASRE Management Service. For example, when an ASRE Management Service requests containers and deploys specific software artifacts onto these containers to create specific resources, we also deploy monitor plugins and configure the plugins to work with a monitoring system.

Monitoring and Metrics: The first aspect is how to obtain enough monitoring data for analyzing and managing ASRE. We develop two different mechanisms for monitoring resources within ASREs. The first one is based on runtime service information from existing providers, whereas the second one is based on a combination of service providers and possible artifact deployment. Monitoring data will be obtained through: (i) querying internal monitoring data from service providers, (ii) intercepting interactions among resources, and (iii) deploying additional monitoring within resources when an ASRE Management Service deploys artifacts. This requires us to perform various integration to existing tools and APIs. In addition, for ASRE, aggregating monitoring data is not enough, we also need to define which metrics can be used to characterize ASRE as a whole. Currently, we support some conventional metrics but develop new evaluation, especially, for the following ASRE-as-a-whole metric:

Metrics	Evaluation
cost	determined from all resources. The cost for the whole ASRE is complex due to the involvement of multiple resources at different levels
availability	determined based on the availability of individual resources and the ASRE topology
reliability	determined based on the reliability of individual resources and the ASRE topology
elasticity	we breakdown the topology of the ASRE resource slice and determine elasticity for each region (e.g., capabilities to add/remove resources and costs) and for the whole ASRE.

To evaluate the metrics, we gather monitoring information and the structure of ASRE's resource slices and define user-defined functions for evaluating metrics.

Elasticity: As ASREs manage resources for the developer, ASRE elasticity must be supported. Hence elasticity for ASREs is not just infrastructural resource elasticity (for containers and VMs) but a multiple dimensional elasticity with a focus on high-level resources. Two important aspects for ASRE elasticity are (i) to change resource capabilities and (ii) add new capabilities. For the first case, ASRE elasticity model is designed based on a coordination of elasticity whereas we leverage the elasticity capability of individual providers. To implement it, severless-based coordination may be a right choice but currently we are only at the design phase for the cross-layered resource coordination algorithm. To support adaptation and change of ASRE functional capabilities at runtime, we utilize service mesh techniques [5].

Resilience: With respect to resource failures, we consider various failures mentioned in [6] but currently for dealing with failures of a resource we just apply a retry mechanism with the resource provider or a new provider, assumed that each provider has its own resilience mechanisms for its resources.

Secret Management for ASRE: Various resources in one ASRE will have different ways for authentication/authorization when resources are invoked (for monitoring/control or business functions). Usually, they have secrets (e.g., API keys, certificates, and communication keys) for communication and management. To manage these secrets for an ASRE in an transparent manner to the application is also important for automating management tasks of ASREs, such as control and deployment of resources. ASRE assumes that such secrets can be managed by leveraging existing secret-as-a-service, such as Vault[3]. We are investigating how to integrate common secret-as-a-service and the SPIFEE[4] to support this feature.

V. Example of an application scenario

The working prototype for ASRE is based on our rsiHub[5] where we leverage existing resource slices and rsiHub management services to implement our ASRE concepts.

Description: A seaport analytics can invoke several types of resources at the edge (inside the seaport), such as containers for running analysis of gate access of trucks, MQTT and RabbitMQ for message brokers to obtain monitoring data about gate access of trucks, data resources about seaport light information, weather information and vessel positions, and Node-RED engines for event analytics. We can have different ASREs for typical analytics with the seaport, such as a seaport schedule, but also for emergency situation, such as an accident with container trucks.

Implementation: Figure 4 shows some key resource providers, which offer resources in the edge (e.g., sensor and MQTT broker) or resources based on real cloud services (e.g., Google Big Query and cloudamqp.com). ASRE resources are created by controlling the corresponding providers. ASRE resources are interacted to support the application feature (such as a realtime sensor-broker-ingestion-database pipeline) and to control infrastructures (e.g., `firewall-resource` controls the edge `kubernetes` system).

VI. Related Work

Existing research works are focused on models for managing VMs and containers for the application, assuming that other high-level services atop containers and VMs are under

[3]https://www.vaultproject.io/

[4]https://github.com/spiffe/spiffe

[5]GitHub: https://github.com/rdsea/HINC

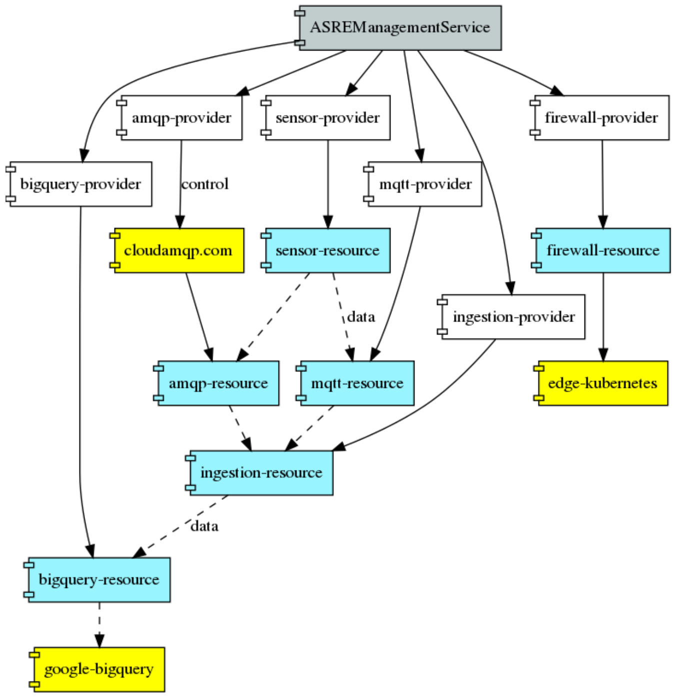

Fig. 4. Different types of ASRE resources in the seaport scenario (a solid line denotes control actions, a dashed line denotes a data flow)

the responsibility of the application. Our work differs from them that we look at a high-level of resources. For example, works in [7], [8], [9] focus mostly on VMs and containers. Such efforts produce information models that can be part of ASREs and thus can be leveraged for abstracting capabilities of IoT services and network services. However, they do not address the modeling of ASREs in our view. Many industrial specifications allow us to define resource structures. Similarly, centered around TOSCA [10], various papers and projects have developed tools for specifying cloud resource topologies for automated deployment. These works are related to resource slices and management in ASREs. However, ASREs are not about resource topologies for deployment but include certain types of information about resources and their properties. An ASRE instance is an abstract entity virtualizing cross-layered cross-system resources for the application.

Recently, there has been work discussing about the intelligent distribution between the edge and the cloud [11]. However, they do not introduce concepts to manage and programming resources for specific IoT services across the edge and the cloud. In cloud and edge computing, many papers have presented various resource scheduling, elasticity and optimization solutions. ASRE will need to address similar solutions in key designs for ASRE. Recently, Baresi et. al presented a mode for managing continuum applications which are deployed in continuum edge and cloud [12]. They focus on the applications, which are executed in edge-cloud platforms, assuming that the platforms will provide resources for the applications. Our work focuses on continuum resources, which can be used for application-specific contexts.

VII. Conclusions and Future Work

ASRE is a high-level concept with the aim to simplify how the developer manages the application-specific resources in edge-cloud environments, which are not just infrastructural containers or VMs. We have explained how ASRE deals with many technical and integration details due to the diversity of edge and cloud resources. We currently elaborate ASRE models, especially, security models and elasticity models to enable elastic edge-cloud mesh of resources. Another aspect is to provide the model of ASRE-as-a-service, which enhances the proposed ASRE Management Service with zero trust designs.

Acknowledgments: We thank Lingfan Gao for implementing preliminary work on resource slices used as the basis for ASRE. We are grateful to the Karajan consortium in giving comments about the concepts of ASRE. A longer version of this paper is available at https://bit.ly/3eRu5yW.

References

[1] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the internet of things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018.

[2] Seldon. Last access: April 4, 2020. [Online]. Available: https://www.seldon.io

[3] S. A. Noghabi, J. Kolb, P. Bodík, and E. Cuervo, "Steel: Simplified development and deployment of edge-cloud applications," in *10th USENIX Workshop on Hot Topics in Cloud Computing, HotCloud 2018, Boston, MA, USA, July 9, 2018.*, 2018.

[4] H. Truong, L. Gao, and M. Hammerer, "Service architectures and dynamic solutions for interoperability of iot, network functions and cloud resources," in *Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings, ECSA 2018, Madrid, Spain, September 24-28, 2018*, 2018, pp. 2:1–2:4.

[5] O. Sheikh, S. Dikaleh, D. Mistry, D. Pape, and C. Felix, "Modernize digital applications with microservices management using the istio service mesh," in *Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering*, ser. CASCON '18. Riverton, NJ, USA: IBM Corp., 2018, pp. 359–360.

[6] D. G. Choudhury and T. Perrett, "Designing cluster schedulers for internet-scale services," *Queue*, vol. 16, no. 1, pp. 30:98–30:119, Feb. 2018.

[7] V. Medel, O. Rana, J. . Baares, and U. Arronategui, "Modelling performance amp; resource management in kubernetes," in *2016 IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC)*, Dec 2016, pp. 257–262.

[8] R.-A. Cherrueau, A. Lebre, D. Pertin, F. Wuhib, and J. M. Soares, "Edge computing resource management system: a critical building block! initiating the debate via openstack," in *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*. Boston, MA: USENIX Association, 2018.

[9] X. Masip-Bruin, E. Marín-Tordera, A. Juan-Ferrer, A. Queralt, A. Jukan, J. Garcia, D. Lezzi, J. Jensen, C. Cordeiro, A. Leckey, A. Salis, D. Guilhot, and M. Cankar, "mf2c: Towards a coordinated management of the iot-fog-cloud continuum," in *Proceedings of the 4th ACM MobiHoc Workshop on Experiences with the Design and Implementation of Smart Objects*, ser. SMARTOBJECTS '18. New York, NY, USA: ACM, 2018, pp. 8:1–8:8.

[10] T. Binz, G. Breiter, F. Leyman, and T. Spatzier, "Portable cloud services using tosca," *IEEE Internet Computing*, vol. 16, no. 3, pp. 80–85, May 2012.

[11] F. Jalali, O. J. Smith, T. Lynar, and F. Suits, "Cognitive iot gateways: Automatic task sharing and switching between cloud and edge/fog computing," in *Proceedings of the SIGCOMM Posters and Demos*, ser. SIGCOMM Posters and Demos '17. New York, NY, USA: ACM, 2017, pp. 121–123.

[12] L. Baresi, D. F. Mendonça, M. Garriga, S. Guinea, and G. Quattrocchi, "A unified model for the mobile-edge-cloud continuum," *ACM Trans. Internet Technol.*, vol. 19, no. 2, pp. 29:1–29:21, Apr. 2019.