A Serverless Computing Platform for Software Defined Networks

Fatemeh Banaie[0000−0002−8382−0113] and Karim Djemame[0000−0001−5811−5263]

School of Computing, University of Leeds, Leeds LS2 9JT, UK
K.Djemame@leeds.ac.uk

Abstract. Recent advances in network management strategies, namely the possibility of network programmability through the use of Software-Defined Networking (SDN) increase the velocity of network evolutions. SDN promises a software-based networking approach, where software modules are used to abstract network functionalities. To achieve this aim, the virtualization paradigm can be used in these modules within the context of Network Function Virtualisation (NFV). NFV plays the most important role in transition toward open software and network hardware. Given the promise of these technologies, micro-services can greatly benefit from the integration of SDN and NFV and execute in a suitable cloud platform. This paper describes a modular, and micro-service based SDN architecture that applies network programmability within the context of NFV and explores how it could benefit from the serverless computing paradigm. Serverless computing accompanies modular SDN in building cost-effective, energy-aware, and scalable networks, relieving the management burden of network maintenance.

Keywords: network management · open-source software · Software Defined Networks · serverless computing · Network Function Virtualization.

1 Introduction

Software-Defined Networking (SDN) and Network Functions Virtualization (NFV) are new paradigms in the move towards open software and network hardware [1]. SDN aims to accelerate the design and implementation of the next generation computer networks. It decouples vertical integration of the control plane and data plane and provides flexibility that allows software to program the data plane hardware directly according to a set of network policies. SDNs have the ability to facilitate the containerised applications and network traffic consolidation to optimise not only performance but energy consumption as well [2].

Serverless Computing [3] offers the illusion of infinite resources that are dynamically provisioned by cloud providers, allowing users to invest less effort and capital in infrastructure management. Moreover, a serverless computing system is an ideal solution to build and optimise any Internet of Things (IoT) operations with zero infrastructure and maintenance costs and little-to-no operating expense [4].

The SDN controllers are a great fit for the serverless computing paradigm as they are highly event-driven, modular, and parallel [5]. Moreover, serverless computing provides a resource-efficient, low overhead alternative to Virtual Machines (VMs) and containers, and can effectively support SDN and function virtualisation. Network function virtualisation (NFV) decouples networking software from the hardware that delivers it so that software can evolve independently [5].

The integrated SDN/NFV architecture deployed on a serverless platform can accelerate the innovation and deployment of network services. A serverless function is essentially a proxy for energy usage as a unit of (serverless) compute and therefore a cost, making network functions instantiation and orchestration significantly energy and resource efficient [2]. Therefore, this research aims at realizing the concept of modular SDN based on serverless functions with the goal to implement a novel platform to reduce the energy consumption of applications deployment and operation on the Internet [2]. The platform's new building blocks are made of 1) a methodology combining SDN, NFV and serverless architectures; 2) placement algorithms for serverless functions to minimise energy consumption; 3) the underlying software implementation.

This paper explores the technical issues and implementation of the proposed SDN/NFV architecture as a first step. The following summarizes its main contributions:

- We propose a micro-service based SDN/NFV architecture, where network services can be deployed as serverless functions. The proposed architecture leverages the virtualisation technology in deploying network functions on a serverless platform. Consequently, this architecture disaggregates the network functionalities so that the SDN controller only provides the minimum required functionalities, and the other network services can be deployed on zero infrastructure with a little-to-no operating expense.
- The proposed architecture is implemented using the well-known open network operating system (ONOS) [6] and function as service (OpenFaaS) serverless platform [7].

The rest of the paper is organized as follows. Section 2 provides a brief overview of the related literature. In Section 3, we briefly describe the proposed architecture used in this paper. Section 4 describes the details of implementation. Section 5 demonstrates and discusses the evaluation results, followed by a conclusion in Section 6.

2 Related Work

Related work snaps in the area of how to overcome the challenges that rise from deploying SDN and whether the underlying network services could be efficiently delivered, managed, and disseminated to the end users. This includes SDN integration with serverless architecture. Reference [8] provides insights into key factors such as the computational resources, the number of Virtual Network Functions (VNFs) running on a VM, and their resource demands affecting

the performance of VNFs that are hosted in virtualized system architectures. Reference [9] also provides an overview of NFV-based service development and requirements that are addressed in existing Service Development Kits (SDKs).

Some open-source implementations of SDN adopt a monolithic software approach besides utilizing the concept of VNFs in service delivery such as ONOS and OpenDaylight (ODL) [10]. For example, ODL can provide agile service delivery on OpenStack cloud infrastructures by implementing services using NFV. However, recent studies have started to move towards a microservices-based architecture. Reference [11] presents a distributed architecture for the design and implementation of SDN control plane systems that split the current monolithic controller software into a set of cooperating microservices, which can be implemented in different (and appropriate) programming languages. The architecture supports the distribution of events to external processes. In this regard, some studies focus on SDN-based strategies to manage the network efficiently, especially in the case of smart IoT applications, e.g., to leverage a SDN-based approach to satisfy the latency requirements of the services in multi-access edge computing applications [12]. However, these approaches do not aim to design a modular SDN architecture, but rather focus on utilizing SDN for managing distributed applications in an edge environment.

The adaptation of the serverless and Function as a Service (FaaS) paradigm in an edge environment was introduced in [13], where key issues and high-level directions were proposed, and different types of deployment and a serverless platform were discussed. This was supported by a prototype implemented with the open-source serverless solution OpenWhisk [14] and open source products.

In [15], a framework is proposed for efficient dispatching of stateless tasks with the goal of minimizing the response times and exhibiting short and long-term fairness. Their evaluation of the OpenWisk platform shows that the interaction with SDN controller can be useful in relieving network congestion. A high-performance serverless platform for NFV is presented in [16], in which the authors utilize three different mechanisms for minimizing the latency, including state management, efficient NF execution model, and avoiding packet latency. The work presented in [18] focuses on energy efficiency by decomposing the application into fine-grained functions.

3 A FaaS architecture for modular SDN

Network management strategies are undergoing a transition from using the proprietary technology of a vendor toward the open-source software modules with service automation. The first glimmer of this transformation is SDN that increases the velocity of network evolutions by delivering new network capabilities. SDN aims to decouple the control plane and data plane for scalability and easier network management. The control plane consists of SDN core functionalities (e.g. topology service, flow service, inventory service, etc.) and management applications (e.g. firewall, load balancing, routing, monitoring, etc.), which communicate requirements via Northbound Application Program Interfaces (API).

The data plane consists of forwarding elements (i.e., switches and routers) and uses OpenFlow [19] as Southbound API.

SDN controller operates by serving the events from both the southbound and northbound APIs. These events can be defined as any changes in the network that lead to invoking one or more SDN's management applications. Therefore, the SDN controller can be designed as an event-driven and modular software that operates by responding to these events. The management applications can be deployed on a serverless platform and can be executed whenever required. However, the monolithic design of current SDN controllers aggregates all functions into a single and huge program. This approach restricts its ability to deploy a new service, independent of other services. As a result, a modular and micro-service based SDN architecture [11] deployed on a serverless platform is developed as illustrated in Fig. 1, which applies network programmability within the context of NFV. In this approach, SDN core services provide minimum required functionality, and the other services can be provided by external applications in the form of a set of cooperating software modules. These modules can be implemented as a set of independent functions (network functions) that leverage the benefits of the serverless computing paradigm in providing on-demand scalability and efficient resource management.

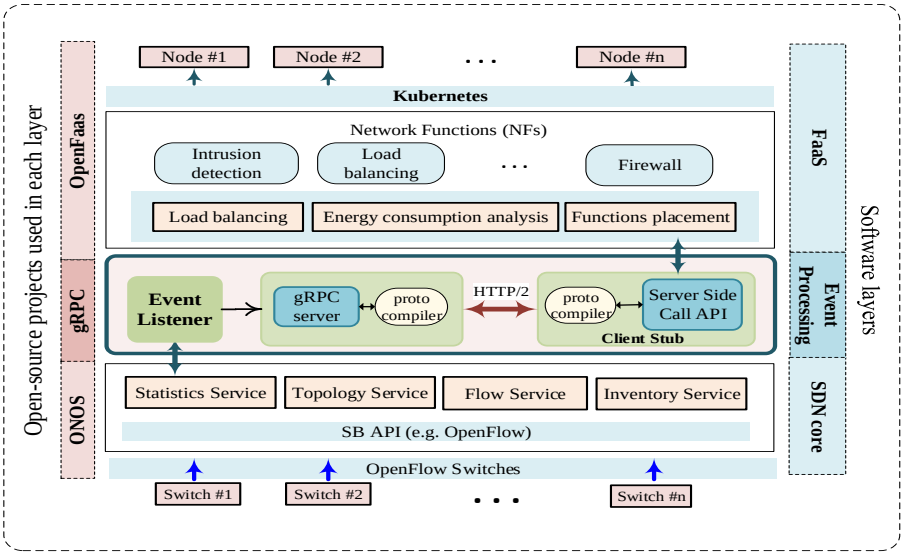

Fig. 1. A FaaS architecture for modular SDN.

The FaaS paradigm provides a platform to develop applications as a set of independent functions. It accelerates application deployment by eliminating the need for managing infrastructure, thus can be well suited for our SDN/NFV

architecture. Moreover, FaaS eliminates the management burden of resource allocation (i.e., choosing the right time as well as the right type of VMs, and containers) and further reduces costs. Accordingly, users only provide network functions (which are SDN microservices) to this platform. Upon receiving an event from Application APIs, the platform start executing the services automatically by deploying new instances. Network developers no longer need to consider function deployment, management, and scaling issues. Besides reducing the management burden, network service providers are charged according to the number of events in the NFV context and therefore only pay for what they use at a very fine granularity [16]. The next section gives details of the implementation of the architecture. The associated source code is available in the GitHub repository [1].

4 Implementation

This section presents the software components and technologies leveraged in implementing the serverless SDN/NFV architecture. We utilized three well-known open-source projects, i.e., Open Network Operating System(ONOS) [6], general-purpose Remote Procedure Call (gRPC) [20], and Functions as a Service (OpenFaaS) [7], for implementing our model's layer, respectively. As can be seen in Fig. 1, the architecture consists of three main components including:

- Event listening module in SDN layer which is responsible for catching an OpenFlow-based events coming from mininet and forwarding them toward OpenFaaS connector.
- Event processing module that uses a communication interface to notify ONOS events to the VNF deployed on OpenFaaS. This module provides the ability of interacting with the virtual functions via gRPC protocol.
- Microservice handling module in the serverless layer that manages the functions and invocations by receiving event notifications from underlying network.

ONOS is an open-source SDN controller developed by open networking foundation (ONF). It has a layered structure in its architectural design including the application, core and providers/protocols layers. Applications use a collection of Northbound Interfaces (NBI) to stay informed about the events and network states. ONOS services are implemented in the core part, where it manages the network states and notifies the applications by occurring the relevant changes in states. These services also provide an inventory of currently connected devices, hosts, links, and an overview of the current network topology, along with the rules installed in the devices. The lowest layer of ONOS stack consists of the Southbound Interface (SBI), where a collection of plugins resides including a provider interface with protocol-specific libraries and a service interface. It is responsible for interacting with the network environment using various control and configuration protocols. The mininet environment is used for emulating the underlying network such as switches and hosts as well as generating network packets coming from data plane [17].

[1] https://github.com/EDGNSS

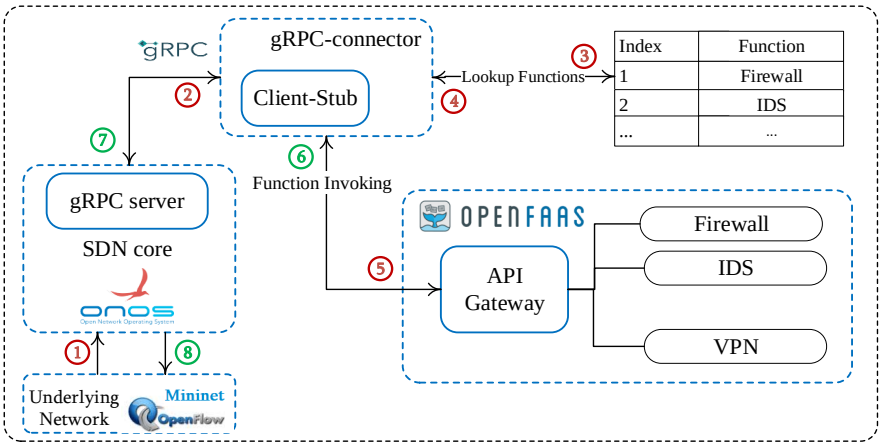

Fig. 2. ONOS and OpenFaaS Integration Sequense Diagram.

In ONOS, the events are used to notify the listening applications about the changes in network. According to the service structure in ONOS, we leverage the benefits of an event distribution system to externalize event processing in SDN [11]. This allows us to divide the control plane services into a set of cooperating microservices, where the minimum required services are provided by core and the extra functionalities can be served by microservices which are implemented on OpenFaaS (Fig. 2).

These microservices can be deployed as web services using VNFs in a distributed environment. The event listener and virtual functions are communicating (e.g. sending and receiving event notifications) through the communication interface. The communication interface used in this project is based on open-source communication interfaces. So, the next component of our event processing layer is the gRPC protocol that is used for connecting these microservices. gRPC is an open-source RPC system based on a protocol buffer that uses HTTP/2 as its underlying transport protocol. It has a lot of interesting features such as security, authentication mechanism, bidirectional streaming, etc. It also allows us to automatically generate the required code for both the server and client sides. Moreover, gRPC provides faster information exchange due to the compressed data packing of the Protocol Buffers and the use of HTTP/2. We have implemented a gRPC server as the ONOS application, where it makes a call to ONOS services and turned the returned value into protobuf form, serializes, and sends it to the client.

The client stub is implemented as an external application, named gRPC-connector to OpenFaas, that connects ONOS services to OpenFaaS Functions. OpenFaaS is a popular open-source serverless platform that makes it easy for developers to deploy event-driven functions with docker and Kubernetes. OpenFaas functions and microservices can be easily invoked by any kind of event, for

example, HTTP is one of the use cases that can connect systems. They are accessible over HTTP endpoints via the Gateway service. Upon receiving an event from ONOS, the connector queries the list of functions in Gateway and then builds a map between the event's topic and the existing functions, which leads to triggering the respective function on a given topic.

5 Evaluation

5.1 Results and Discussion

To evaluate the performance of SDN/NFV architecture, we deploy the architecture on two servers with 4 VCPUs, 16 GB memory running on Microsoft Azure. The first server runs onos with grpc-server installed on it, along with mininet which emulates a network with 8 openflow-enabled switches and hosts. The second server runs OpenFaaS which is used as a resource pool and network functions host. These two servers are connected via 1Gbps LAN network and run Ubuntu 20.04 LTS. At first, the feasibility of the modular and distributed architecture is investigated by deploying a simple serverless forwarding application (used for packet processing for every new packet arriving at the controller). In this scenario, once a host sends a ping request for which there is not any established forwarding rule, the switch directs the incoming packet to the SDN controller. The event listening module of the controller serves the packet by sending it to the external application deployed on a serverless platform via a gRPC channel. The deployed function can process the packet to find the source and destination addresses of the packet.

Furthermore, the performance of the proposed architecture is reported through some preliminary results considering various scenarios. The objective of the evaluation is to compare the performance of the proposed distributed architecture implementation with a monolithic one that hosts local applications. A host sends a continuous flow of IP packets, which generates the event notification in the controller, and accordingly leads a function to be invoked on OpenFaaS. In this experiment, ONOS communication with VNFs is handled via gRPC protocol and the OpenFaaS orchestrates the function requests by an automatic scaling process. We have conducted the experimental evaluation of our model through a diverse set of serverless workloads. To do so, the traffic load is increased during experiments to observe the behavior of both models in higher traffic arrivals. Fig. 3(a) illustrates the effect of the number of packets per second on the mean response time recorded during the experiment. Each observation shows the latency in both the control plane and the VNFs layer.

As can be seen, the mean response times of the local functions are lower than serverless-based functions at the start of the experiment. The reason for this is the latency imposed by grpc channel to invoke the remote functions on OpenFaas. Furthermore, both applications (i.e., local and remote functions) experience a higher amount of response times with increasing the arrival rate of packets on the data plane, because it imposes a higher processing load in the network. However, it is observed that the serverless-based approach outperforms

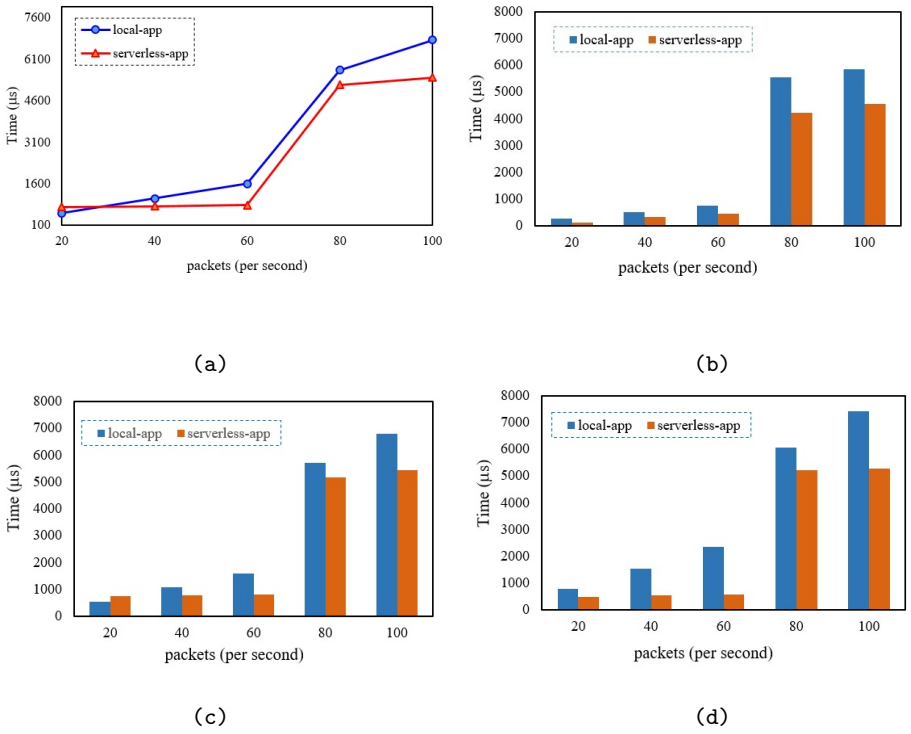

Fig. 3. The Average delay in: (a) different arrival rates. (b) service rate $= \frac{1}{b} = 25\mu s$. (c)service rate $\frac{1}{b} = 34\mu s$. (d)service rate $\frac{1}{b} = 42\mu s$.

the local method in higher packet arrivals, as it can automatically scale and manage the resources according to the workload.

In the second scenario, we have considered the latency in different service rates of the functions, which is due to the processing of the different packet sizes. Fig. 3(b) compares the results in mean processing time of $\frac{1}{b} = 25\mu s$ for both local and serverless-based applications. As observed, the mean delay of both applications shows an up trend. However, it is lower for serverless-based application than the local one with the results for local functions showing a steep slope. Furthermore, with increasing the processing time of the function in Fig. 3(c) to $\frac{1}{b} = 34\mu s$, local functions win at low packet arrivals, but experience higher amounts of delay with increasing the arrival rate. As discussed previously, the reason is due to the scalability of the serverless platform, which can also be seen from Fig. 3(d) in $\frac{1}{b} = 42\mu s$. Intuitively, the higher the workload in each experiment is, the more important the ability of scaling becomes. These results can clearly show the benefits of the serverless-based method in latency handling and service scaling.

5.2 Extensions

In this paper, a modular and distributed SDN framework that paves the way to manage a novel paradigm of microservices-based applications in designing the next generation IoT networks is presented. As such, the use of microservices represents a big step ahead in the vision of serverless edge architectures. However, the serverless paradigm still requires addressing special challenges to enhance its applicability across all phases of the general application landscape, including network security services. The following improvements can be considered to support the proposed architecture:

- As *energy consumption* is a vital issue in distributed systems, employing edge computing with low-powered computing resources contributes to more energy-efficient system development. Moreover, an energy aware automatic serverless functions instantiation and orchestration framework for edge computing environments is currently being implemented. In this context, a resource mapping algorithm is necessary to address the energy efficiency in the serverless platform by appropriately selecting the server nodes (i.e., the right CPU cores to run the functions) and the placement of functions. Functions taking up the most execution time can be identified, which equals *cost*.

- The modules in an SDN controller are implemented and deployed as separate, stand-alone serverless functions. Accordingly, a suitable task scheduling method can enable on-demand *scaling up/down* of the functions with regards to their workload in an energy-aware environment [21].

- To avoid start-up latency, the underlying virtualisation technology must be optimised to decrease the start-up latency of *cold* start launching in a selverless platform. This can be tackled by suitable managing of the function instances in this platform, e.g., reusing launched instances by keeping them *warm* for a period of time [21], or reduction of the container image size.

- Network functions are usually short-lived and in most scenarios, they are chained together to form a Service Function Chain (SFC). The development and performance tuning of SCFs are difficult and should be considered in designing the more complex application scenarios.

- An effective *load balancing* algorithm can also manage the resource utilisation by distributing the function executions to the available resources, which can further improve the performance of the system in terms of energy expenditure and service delivery latency. However, load balancing approaches may not necessarily enhance the performance of group functions in SFC due to the loss of locality [5]. In other word, the locality requirements must be considered to group functions as a single application to provide faster data sharing among interacting functions.

6 Conclusion

This paper presents a microservices-based distributed architecture for implementing SDN controller in order to improve agility, scalability, and performance in the network. The SDN features such as the possibility of network programmability and reconfiguration according to the application requirements make it a key enabler for the upcoming next-generation IoT networks. However, the limitations of the existing monolithic SDN architecture restrict it to fulfil the needs of IoT applications in terms of scalability and performance. In particular, a good system design needs modularity, so, a modular and microservice-based SDN architecture is required to tackle these limitations in distributed system environments such as IoT applications.

The experimental results show that the serverless paradigm can decrease service latency for disaggregated architectures, and also provide on-demand and scalable resource management. The reduction in the execution time and the average resource usage of microservices allows for many optimizations from the resource management point of view.

Acknowledgements The authors would like to thank the European Next Generation Internet Program for Open INTErnet Renovation (NGI-Pointer 2) for supporting this work under contract 871528 (EDGENESS Project).

References

1. Bonfim, M. S., Dias, K. L., Fernandes, F. L.: Integrated nfv/sdn architectures: A systematic literature review. ACM Computing Surveys, **51**(6), (2019)
2. Djemame, K.: Energy efficiency in edge environments:a serverless computing approach. Proceedings of the 18th International Conference on the Economics of Grids, Clouds, Systems and Services (GECON), LNCS Vol 13072. Springer, pp. 181–184, (2021)
3. Kiritikos, K. ,and Skrzypek, P.: A Review of Serverless Frameworks. in 2018 IEEE/ACM International Conference on Utility and Cloud Computing, pp. 161–168 (2018)
4. Gorbmann, M. , Ioannidis, C., Le, D.: Applicability of Serverless Computing in Fog Computing Environments for IoT Scenarios. in Proc. of the 12th IEEE/ACM International Conference on Utility and Cloud Computing, Auckland, NZ:ACM, pp. 29–34 (2019)
5. Aditya, P.,Akkus, I. E., Beck, A., Chen, R., Hilt, V., Rimac, I., Satzke, K., and Stein, M.: Will Serverless Computing Revolutionized NFV?. Proceeding of the IEEE, **107**(4), pp. 667–678 (2019)
6. Berde, P. , and et.al: ONOS: Towards an Open, Distributed SDN OS, HotSDN '14: Proceedings of the third workshop on Hot topics in software defined networking, ACM, pp. 1-6 (2014)
7. OpenFaas. Serverless functions made simple. [Online]. Available: https://github.com/openfaas/faas (2019)

8. Falkner, M., Leivadeas, A., Lambadaris, I., and Kesidis, G.: Performance Analysis of Virtualise Network Functions on Virtualized Systems Architectures. in 2016 IEEE 21th International Workshop on Computer Aided Modelling and Design of Communication Links and Networks (CAMAD), pp. 71-76 (2016)
9. R.F. Ustok and et.al., Service Development Kit for Media-Type Virtualized Network Services in 5G Networks, IEEE Communications Magazine, **58**(7), pp. 51–57 (2020)
10. OpenDaylight: OpenDaylight and Open Networking ecosystem, https://www.opendaylight.org.
11. Cormer, D., and Rastegarnia, A.: Toward Dissagregating the SDN Control Plane. IEEE Communication Magazine, **57**(10), pp. 70–75 (2019)
12. P. Fondo-Ferreiro and et al., A software-defined networking solution for transparent session and service continuity in dynamic multi-access edge computing. IEEE Transactions on Network and Service Management, **18**(2), pp. 1401–1414 (2020)
13. Baresi, L., Mendonca, D. F., Garriga, M., Guinea, S., Quattrocchi, G.: A Unified Model for the Mobile-edge-cloud Continuum. ACM Trans. Internet Technol., **19**(2), (2019)
14. Djemame, K., Parker, M., Datsev, D.: Open-source Serverless Architecture: an evaluation of apache openwhisk. in 2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC), pp. 329–335 (2020)
15. Cicconetti, C., Conti, M., Passarella, A.: A Decentralized Framework for Serverless Edge Computing in the Internet of Things, IEEE Transactions on Network and Service Management, **18**(2), pp. 2166–2180 (2021)
16. Shen, J., Yu, H., Zheng, Z., Sun, C., Xu, M., Wang, J.: Serpens: A high-performance Serverless Platform for NFV. in 2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS), pp. 1–10 (2020)
17. Download/Get Started with Mininet, http://mininet.org/download (2017)
18. Tzenetopoulos, A., Marantos, C., Gavrielides, G., Xydis, S., Soudris, D.: FADE: FAAS-Inspired Application Decomposition and Energy Aware Function Placement on the Edge. NY, USA, Association for Computing Machinery, pp. 7–10 (2021)
19. Singhvi, A., Khalid, J., Akella, A., Banerjee, S.: SNF: Serverless Network Functions, ACM Symposium on Cloud Computing, (2020)
20. grpc. (2016) A high performance, open source, general-purpose RPC framework. [online]. Available: https://github.com/grpc
21. Z. Li, L. Guo, J. Cheng, Q. Chen, B. He, M. Guo, The Serverless Computing Survey: A Technical Primer for Design Architecture, ACM Computing Surveys (2022)