


A distributed computing model based on delegation of serverless microservices in a cloud-to-thing environment


Antonio Silva   [Institute of Informatics, Federal University of Rio Grande do Sul – Brazil, 91.501-970 | aassilva@inf.ufrgs.br], and [Hamm-Lippstadt University of Applied Sciences, 59063 Hamm, Germany | antonio.santosdasilva@hshl.de]


Paulo Mendes  [Airbus, Willy-Messerschmitt-Strasse 1, 82024 Taufkirchen, Germany | paulo.mendes@airbus.com]

Denis Rosário  [Federal University of Pará | denis@ufpa.br]

Eduardo Cerqueira  [Federal University of Pará | cerqueira@ufpa.br]

João Paulo J. da Costa  [Hamm-Lippstadt University of Applied Sciences, 59063 Hamm, Germany | joaopaulo.dacosta@hshl.de]

Edison P. de Freitas  [Federal University of Rio Grande do Sul – Brazil, 91.501-970 | epfreitas@inf.ufrgs.br]

 Institute of Informatics, Federal University of Rio Grande do Sul, Av. Bento Gonçalves 9500, Porto Alegre, Brazil

Received: 22 October 2023 • Accepted: 15 May 2024 • Published: 03 August 2024

Abstract This article explores serverless cloud-to-thing architecture and the virtualization of functions across the network as a solution for implementing services across the cloud-to-thing continuum. It addresses the challenges posed by the emergence of 5G and 6G networks, where they require high bandwidth, low latency, and real-time computing capabilities. The proposed architecture leverages edge computing principles, orchestrating computing functions across edge/fog infrastructure. A serverless approach with microservices offers flexibility and scalability for deploying services on heterogeneous devices. The proof-of-concept implementation demonstrates the architecture's suitability for cloud-to-thing solutions and highlights possible research directions for improving its efficiency and reliability in dynamic network environments. In Scenario A, simulation results indicated a 23% rise in throughput at the cloud level within the first 2 seconds. At the 12-second mark, the throughput became uniformly distributed, indicating a significant offloading of computational tasks. Scenario B showed an almost linear throughput distribution between the cloud and the satellite starting at 8 seconds, highlighting the framework's capacity for dynamic reallocation of computing functions in real-time.

Keywords: Cloud-to-thing, serverless, distributed computing.

1 Introduction

Services targeting 5G and 6G networks, such as Autonomous Vehicles (AVs), Internet of Drones, and others, will consume more bandwidth, need low latency rates, and require extensive real-time computing services [Mahmud and Toosi, 2021; Król and Psaras, 2017]. Edge computing technologies envision an open horizontal architecture for distributed computing that promises to overcome the limitations of cloud-centric execution for different latency-sensitive services by providing computing resources closer to data sources and consumers [Laghari *et al.*, 2021]. However, the location of the servers is pre-set and cannot be changed according to the needs of different services and dynamic configurations of the environment.

In line with the above, researchers are moving towards a new paradigm of network computing continuum, where computing functions can be deployed anywhere along the cloud-edge-fog infrastructure [Dogani *et al.*, 2023]. In this scenario, cloud servers may become control nodes for intelligent edge/fog devices, in which case there is a need for comprehensive orchestration techniques that can coordinate and schedule network services across the cloud-to-thing contin-

uum [Rosário *et al.*, 2018]. In this context, serverless computing [Li *et al.*, 2022] emerges as an attractive technology in which small services run without dealing with the operational problems of server provisioning and resource management. In a world of serverless computing, microservices [Cerny *et al.*, 2022] also play a crucial role in promoting faster, more manageable deployments of services in a cloud-to-thing continuum in which devices (cloud, satellite, and drones) have heterogeneous capabilities [Čilić *et al.*, 2021].

There has been much progress in developing serverless implementation models Sarkar *et al.* [2019]; Goniwada [2022]. However, research in this field is still in the early stages. For example, existing solutions do not support dynamic executions on disparate devices, thus compromising the deployment of microservices solutions [Laghari *et al.*, 2021]. To tackle the properties of a cloud-to-thing continuum, in this article, the serverless architecture is leveraged, allowing computing functions to run on a set of nearby devices (satellites and drones) and not only on servers. In this sense, this work presents serverless-cloud-to-thing as the architectural approach in which programming is isolated from the specification of infrastructure requirements, allowing the combination of networking, storage, and computational resources to

deploy, execute, and adapt the operation of services defined as a set of microservices in the cloud-to-thing continuum.

This proposed serverless-cloud-to-thing model defines services as a set of microservices to allow the exploitation of devices with heterogeneous and potentially limited capabilities. With this serverless [Goniwada, 2022] model, microservices run on demand in response to triggers that service developers can configure in advance. The proposal uses a Software-Defined Networking (SDN) approach to coordinate deploying a series of microservices [Landmark *et al.*, 2018], increasing its flexibility. In addition, SDN enables the underlying network to respond quickly to changes with the proper placement or replacement of microservices in different devices.

Regarding the execution of services, the proposed serverless-cloud-to-thing model considers a data plane based on the concept of Named Data Networking (NDN) [Al-Omaisi *et al.*, 2021], which brings advantages for connecting small services [Król and Psaras, 2017] used to support the execution of services in a distributed manner. NDN enables the development of host-independent network structures better suited for the dynamic behavior of the cloud-to-thing continuum. With an NDN-based data plane, users can directly request the network to execute a specific service without discovering the location of all the previously needed microservices.

The main contributions of this work are the following:

1. Present an overview of serverless architectures for the cloud-to-thing continuum and an approach for implementing services in heterogeneous network environments.
2. Propose a serverless model for the cloud-to-thing continuum, allowing computing functions to be performed on a variety of devices close to consumers, such as satellites and drones, in addition to traditional servers.
3. Providing flexibility and scalability schemes for deploying services on devices with heterogeneous capabilities by using microservices and serverless approaches together.
4. Develop a proof-of-concept for demonstrating the viability of the proposed architecture for cloud-to-thing solutions. In addition, it identifies possible research directions to improve efficiency and reliability in dynamic network environments.

Two scenarios were evaluated, one with a normal load and the other with a high traffic load. The simulations in Scenario A revealed a 23% increase in throughput at the cloud level in the first 2 seconds, suggesting that most of the computational functions were initially processed in the cloud. However, at approximately the 12th second, the throughput exhibited a linear distribution, suggesting that the functions were redistributed to other points in the framework, highlighting the system's ability to reconfigure itself dynamically to optimize the use of resources. In Scenario B, the load was more intense, and From the 8th second onward, the simulation depicted an almost linear distribution of throughput between the cloud and the satellite, showcasing the framework's capability to dynamically reallocate computing functions in real-time to sustain efficiency, even in challenging circumstances. These findings highlight the framework's adaptability to fluctu-

uations in load and connectivity, suggesting its suitability for fog computing applications requiring adaptability and fast response.

The remainder of this paper discusses the design principles and challenges to be addressed so that serverless solutions can effectively contribute to the efficient operation of a cloud-to-thing continuum, leading to the proposal of a new serverless-cloud-to-thing model. This paper describes and evaluates a proof-of-concept that focuses on implementing a cloud-to-thing continuum over a satellite network connecting remote users to cloud providers' Point-of-Presence (PoP). The paper also discusses the current state of the computing continuum and its main principles, challenges, and opportunities related to the operations and management of dynamic serverless computing functions.

2 Main Existing Approaches to Cloud-to-Thing Computing

There has been extensive discussion in the literature about cloud-to-thing computing [Cheng *et al.*, 2019; Verginadis *et al.*, 2021], motivated by the need to manage complex and heterogeneous computing environments while being aware of all available resources to perform dynamic resource allocations and enable short time deployment of services [Gusev, 2021]. The need to consider resources located in the edge and beyond is derived from the fact that cloud resources reside at a distance of several hops from end users, leading to a degradation in the Quality of Services (QoS) [Afrin *et al.*, 2015].

This requires the deployment and coordination of computing functions on devices that are typically closer to users and data sources [Laghari *et al.*, 2021], meaning that small Single-Board Computers (SBC) will be widely used as computing nodes [Mahmud and Toosi, 2021]. On the one hand, these small devices are limited to facilitating multi-location and resource sharing, and on the other hand, managing computing resources through centralized entities further degrades the service performance [Mahmud and Toosi, 2021].

Several solutions have been proposed in the computing continuum space. Con-Pi [Mahmud and Toosi, 2021] relies on virtualization for edge and fog computing services running on SBC for IoT services. Other proposals [Cheng *et al.*, 2019; Verginadis *et al.*, 2021] offer Virtual Machines (VM) or container services on heterogeneous devices from data centers to nano servers. Another direction explores routers, switches, and gateways that are part of the network infrastructure to run services [Sarkar *et al.*, 2019]. While conceptually, routers, switches, and gateways can run VMs, these devices must perform continuous network operations and are often considered unsuitable for running complex services.

To increase the capability of routers, switches, and gateways to run services, solutions are needed to use computational resources efficiently, such as by exploring λ functions [Kaur and Mittal, 2021; Mekbungwan *et al.*, 2022] that are lighter. However, they have several operating limitations and are more complex for developers. In particular, multiple services are inaccessible in a single λ function, so recursive

calling of multiple λ functions leads to the earlier deployment challenges.

The described proposal reveals the power of the computing continuum [Cheng *et al.*, 2019; Verginadis *et al.*, 2021; Gusev, 2021; Afrin *et al.*, 2015; Laghari *et al.*, 2021; Mahmud and Toosi, 2021; Sarkar *et al.*, 2019; Kaur and Mittal, 2021; Mekbungwan *et al.*, 2022]. However, they are all based on a server architecture, meaning that developers need to know the application to be developed and the resources needed for the execution. Finally, deploying server-based solutions is subject to cost and power issues [Patros *et al.*, 2021], and CPU power and memory limitations often characterize the increased latencies of edge devices.

Table 1 summarizes the characteristics and approaches of the different proposed architectures for services in the context of cloud-to-thing. In general, existing frameworks that could be applied to the cloud-to-thing continuum have limited flexibility to support dynamic service composition with a data-driven approach. In addition, existing solutions are dominated by proprietary silos and incompatible technologies built around dedicated devices and runtime stacks.

3 Design Principles

This section presents the fundamental principles that guided the development of the serverless-cloud-to-thing proposal. An SDN-based approach is used to divide the data and control planes. In the data plane, an NDN-based approach is adopted to enable the execution of computational functions on demand. In the control plane, network infrastructure management and monitoring functions are deployed.

The proposal is based on microservices for application development. Figure 1 shows the logical separation of the proposed service. As shown, the service is divided into two parts: the first logical part is called the “Service Manager,” which is responsible for managing a service that contains various computer modules to serve it. The second part is the computing modules; in this case, the service is divided into three computing functions in the example shown, forming a chain of functions with the “Service Manager.” These functions can be deployed on one or more network devices.

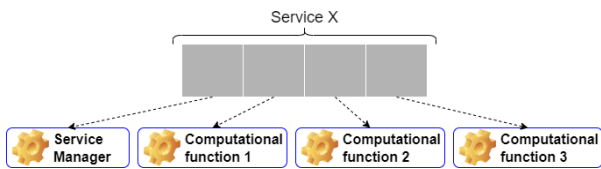


Figure 1. Division of Service X into four processing modules, each representing a part of a chain of computing functions, with the Service Manager as the head of the chain.

Computational functions can be implemented using various technologies, such as Unikernels, Docker Containers, and WebAssembly. As discussed in section 2, Unikernels and Docker Containers have significant mobility and service activation limitations. The proposed architecture uses WebAssembly to implement the computational functions. WebAssembly functions are built to run without a server.

WebAssembly VMs are smaller than other approaches, which allows their binaries to be transported over the network

more efficiently and quickly. In addition, they can be run on any device with computing power. This means that computing functions can be allocated to devices with idle computing resources. When a chain of serverless computing functions for a given service is allocated to a set of devices, these devices define a computing cluster, as all the serverless computing functions are interconnected by the NDN data plane.

Figure 2 illustrates the distribution of Service X’s computing functions, as shown in Figure 1, on various devices in the data plane. The network infrastructure management and monitoring functions are deployed in the control plane, as shown in Figure 2. Each function that makes up the proposed architecture is detailed in Section 4.

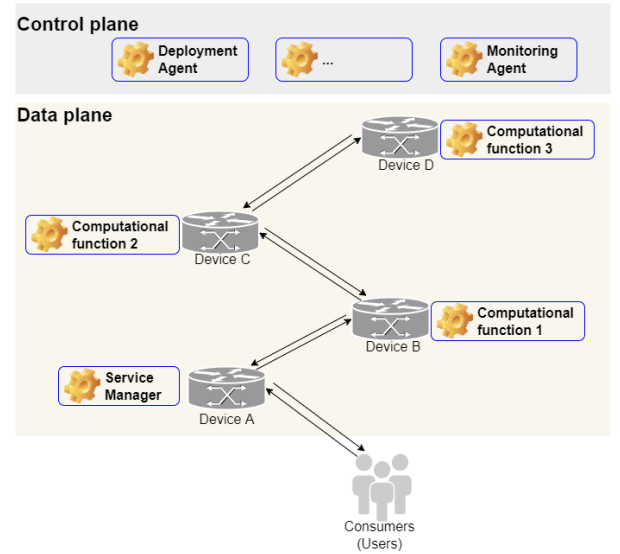


Figure 2. Deployment of computing functions in the data plane and control instances in the control plane. Four devices in the data plane each host a distinct computing function, while the control and monitoring functions are in the control plane.

This way, the proposed serverless-cloud-to-thing model can allow devices, such as cloud servers, to harvest resources from other nonexclusive and sporadically available devices, such as satellites and terminals, and expose these resources to computational services. Computational services are submitted to the network itself, which then can schedule different microservices to other devices within the network. Each device has the capability of monitoring resources and providing state information to other elements of the network. Monitoring network, process, and storage resources are important to allow the creation of service chaining: a set of connected devices that offer a particular service by executing interdependent microservices.

The serverless paradigm used in the proposed serverless-cloud-to-thing model allows system engineers to incorporate more decentralization. In this way, services can be handled in a more independent and isolated way. Furthermore, serverless defines an event-driven programming model, and its advantages include scalability, focus on business logic, and high cohesion. However, taking computing out of its conventional places and integrating it into the user environment poses new challenges due to the heterogeneous capability of different devices. Hence, the proposed model complements a serverless architecture with the idea of microservices, which can loosely couple the modules that are part of a service so

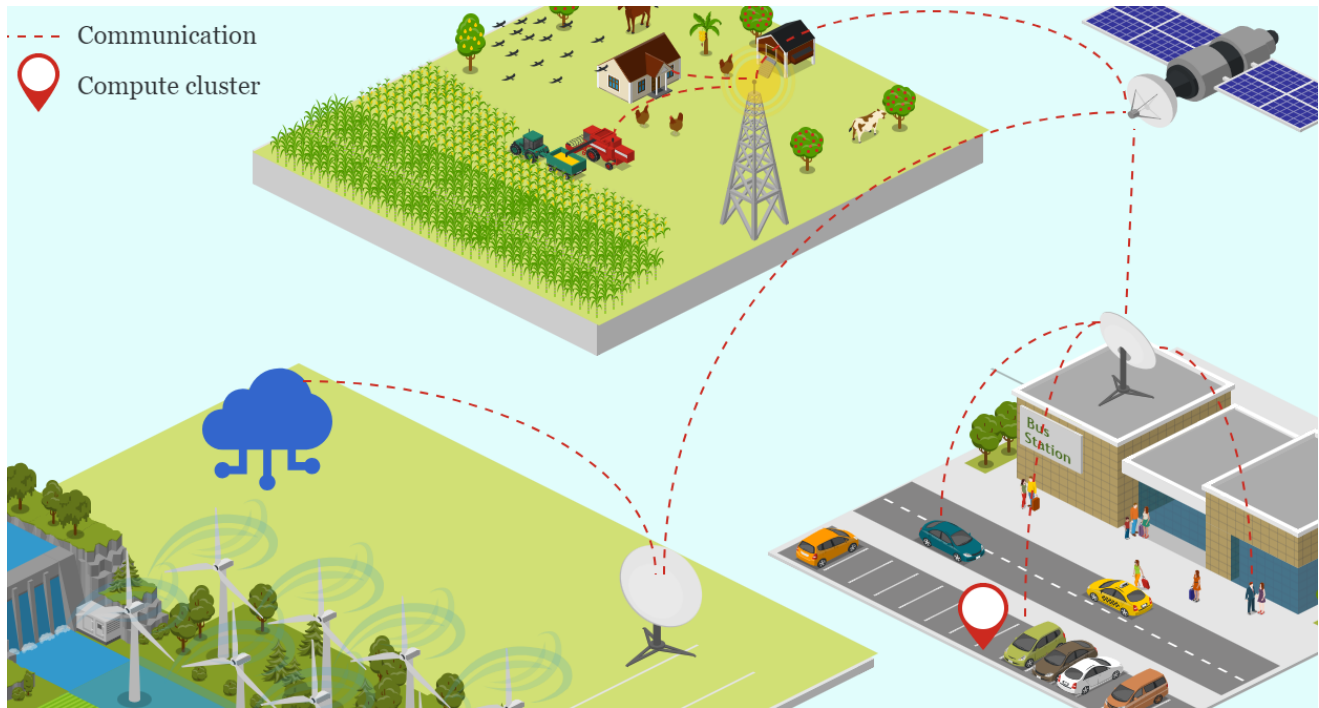
Table 1. Summary of the main cloud-to-thing computing proposals

Proposal	Network approach	Function mobility	Service management	Dedication of resources	Deployment technology	Service allocation
Mahmud and Toosi [2021]	IP*	Partial	No	Dedicated	Docker	Dynamic
Cheng <i>et al.</i> [2019]	IDs [#] GeoHash	Partial	No	Dedicated	Docker	Dynamic
Verginadis <i>et al.</i> [2021]	IP*	Partial	No	Dedicated	Docker	Dynamic
Sarkar <i>et al.</i> [2019]	IDs [#]	Partial	No	Dedicated	Docker	Dynamic
Kaur and Mittal [2021]	IP*	Partial	No	Generic	Docker	Dynamic
Mekbungwan <i>et al.</i> [2022]	ICN [§]	No	No	Dedicated	λ	Static
Gusev [2021]	IP*	No	No	Generic	No	Static
Król and Psaras [2017]	ICN [§]	No	No	Generic	Unikernels	Static
Kjorveziroski and Filiposka [2023]	IP*	Yes	No	Dedicated	WebAssembly	Dynamic
This	ICN[§]	Yes	Yes	Generic	WebAssembly	Dynamic

* Internet Protocol (IP).

Identifiers (IDs).

§ Information-Centric Networking (ICN).

**Figure 3.** An application scenario with urban and rural environments communicating with the cloud using PoPs and base stations. The red dotted line indicates the communication between the devices, and a compute cluster of four parked cars is also observed.

that independent serverless computing can be redefined in real time based on replacing different microservices.

The deployment of microservices in the cloud-to-thing continuum is done based on an SDN-based approach. In this approach, an SDN controller implements a Deployment Agent responsible for the serverless microservice delegation to different devices. In addition, the Deployment Agent is responsible for maintaining the required service quality, defined by intents submitted by the network operator, based on network monitoring information and on the definition of services that dictate the dependencies between different microservices.

4 System Design

The conceptual overview of the serverless-cloud-to-thing model is illustrated in Figure 3, which shows a satellite-based cloud-to-thing continuum, interconnecting the cloud with its PoP, the edge encompassing satellites and the far edge including computer clusters, of vehicles in this case. This scenario can be used to deploy front-hauling, backhauling, or hybrid scenarios. In the front-hauling case, users can access the satellite directly, for instance, based on the presence of a 5G eNB onboard the satellite. In the backhauling case, users interface with the satellite system based on a satellite terminal.

In this scenario, satellites can provide extensive coverage with significantly large bandwidth, as expected in 5G and

6G ecosystems. But in this example, the PoP represents a communication bottleneck, limiting bandwidth since the use of converters (optic-to-digital, radio-to-digital, and others) limits the processing load and, consequently, the bandwidth. This problem can be mitigated by applying the proposed serverless-cloud-to-thing model, in which resources available from the cloud, PoP, satellite, and terminal are considered to distribute service logic across the network, considering available resources and the requirements of the different microservices.

As illustrated in Figure 3, users can be autonomous vehicles, vehicles with Advanced Driver-Assistance System (ADAS), people using smartphones, or IoT devices. These users connect to the network using an ICN approach and request services or data from it, stating only their interests. This procedure is detailed in Subsection 4.1. The services are available throughout the infrastructure and implemented as computational functions deployed on demand, as detailed in Subsection 4.2. After deploying a service, it manages itself using the Service Manager function, detailed in Subsection 4.3. In general, a service can be reallocated to meet predefined requirements. This reallocation of services occurs dynamically and without interrupting the service; this process is detailed in Subsection 4.4.

4.1 Chain of computational functions

The serverless-cloud-to-thing model operates on named services. When users do not implement an NDN-based protocol stack, requests are directed to the nearest device. These requests are then mapped to a name reflecting the service within the NDN on the infrastructure. This design aspect is crucial, as location-agnostic services simplify management for the Deployment Agent installed in the SDN controller. It enables real-time redefinition of service behavior, including replacing microservices within the network infrastructure.

The service discovery process is illustrated in the diagram depicted in Figure 4. This process closely resembles the data discovery process in the NDN approach. However, instead of querying the Content Store (CS), the search is conducted within the Function Store (FS). The functions stored in the FS, developed in WebAssembly, are stored in the controller and transmitted to the devices as data packets, where they are subsequently stored in the FS.

As exemplified by the block diagram in Figure 4, a service is instantiated by a consumer, a user. The routing process is similar to that of NDN: on locating the service in its FS, the device starts executing the request or adds it to a queue if the service is already running. The computational function returns using the same concept as NDN. If a service is not available on the device, routing to the next device is carried out according to the Forwarding Information Base (FIB), and an entry is added or updated in the Pending Interest Table (PIT), in a similar way to NDN.

In this context, an extension to the current WebAssembly VM model is proposed. This change allows independent serverless execution and the chaining of functions that form a service, as presented in Figure 5. In the proposed serverless-cloud-to-thing model, WebAssembly does not include “Runtime” operations, and a service manager for each

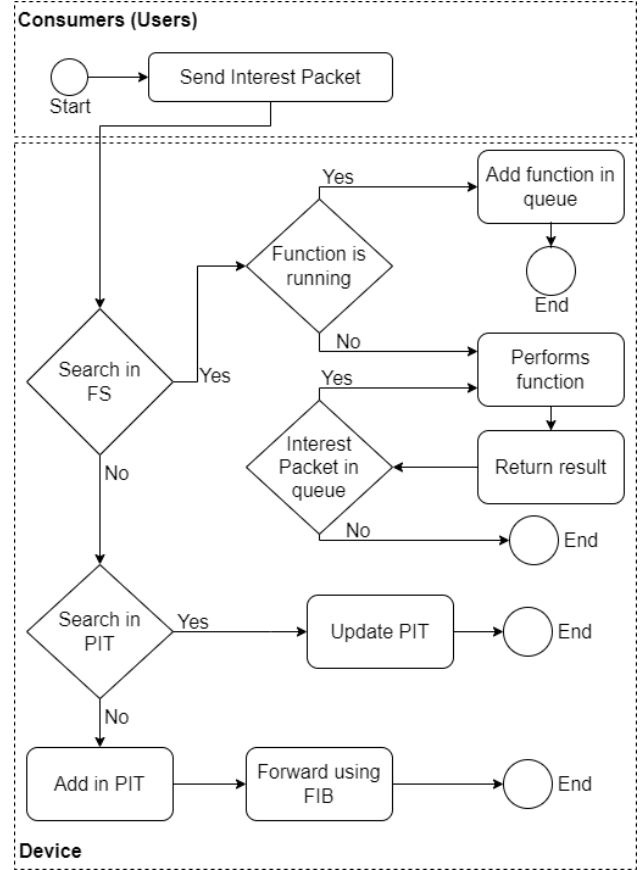


Figure 4. Block diagram illustrating the process of discovering and running a computational function.

service and a service agent for each device are added. For the “Wasm Module,” the “Imports” mechanism is removed, and the “Exports” becomes referential, thus guaranteeing the isolation and independence of each computational function. Finally, “Chain sequence” is added to reference the position of the computation function in the chaining formed for the service.

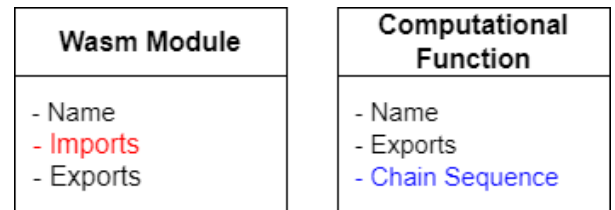


Figure 5. On the left, the components of a WebAssembly module (Wasm Module) are shown, while on the right are the components of a Computational Function, an adaptation of a WebAssembly module to support the chaining of computational functions of a microservice. The component highlighted in red represents the removal, while the component in blue indicates the addition to the implementation.

This way, it can be executed in any device with computational capacity through a Service Agents that communicates directly with the device’s hardware, meaning that computing functions can be allocated to devices with idle computational resources. When a chain of serverless computing functions of a certain service are allocated to a set of devices, those devices define a compute cluster since all serverless computing functions are interconnected by the NDN data plane (see Figure 3).

4.2 Deployment

During the development of a service, a developer should not be concerned with managing the computational and memory resources required to run the service. During runtime, a service must meet the application and user requirements.

Hence, serverless microservice functionality is incorporated to address the above aspects and extend them to computational functions. In addition, the proposal is a new implementation model of computational functions expressed in WebAssembly, to ensure the isolation of each computational function and deployed by the Deployment Agent interdependently of other computational functions that make a service.

The Deployment Agent onboard the SDN controller is responsible for knowing all the chain compute functions part of the serverless service, as well as the requirements of each service, so that after deployment of the service, it can identify whether its requirements are being met by monitoring all microservices. So, when a requirement becomes unmet, the Service Manager can react by analyzing the best options to reallocate the underperforming microservices.

Each computational function implemented in WebAssembly traverses the network as a data packet named `"/binary/<computational function name>."` Packets with the prefix `"/binary/..."` are recognized in networked devices by Service Agents implementing NDN-based serverless-cloud-to-thing model as binary VMs. Such Service Agents can put into execution this VM on their hardware. In this way, any device (thing) with computing power can run the VMs and become part of a cloud-to-thing continuum.

As mentioned before, the Deployment Agent is performed using an SDN-based approach. The choice of the centralized management approach is linked to the objective of the proposed work, where each function computing behavior can be redefined in real-time and independently. Thus, the proposed architecture presents a Deployment Agent in each infrastructure: a Deployment Agent for the cloud and a Deployment Agent for the satellite. The Deployment Agents communicate among themselves, but the infrastructure's computational functions respond only to the corresponding infrastructure's Deployment Agent. For instance, the computational functions present in the cloud respond only to the cloud's Deployment Agent. In short, the Deployment Agent ensures QoS and Quality of Experience (QoE) in its network infrastructure. Note that the proposal aims at 5G and 6G networks, so the proof-of-concept aims at applications with the characteristics defined in the proposed standards for 5G and 6G networks.

Regarding the cloud Deployment Agent, the SDN architecture allows it to be composed of one or several applications, for example, applications based on Artificial Intelligence (AI) or static policies. A single static policy-based application is used for the proof-of-concept, the bandwidth. The proof-of-concept application uses variables for each type of data stream. This variable stores the average data traffic per millisecond for a given stream (one application type is responsible for one stream). Thus, if the bandwidth consumption is close to the maximum, the service with the highest data consumption is sent to devices near the data source by using packets with the prefix `"/binary/..."`, as described

in the previous subsection. Since our proof-of-concept data sources are satellite users, such as vehicles, the microservices of the service to be replaced are sent to the Deployment Agent in the satellite infrastructure.

As far as the satellite Deployment Agent is concerned, just like the cloud Deployment Agent, it can also be composed of one or more applications. For the proof-of-concept, an application that monitors the link of the PoP of the service provider was implemented. In the case of the satellite infrastructure, the allocation of the computational functions occurs differently than in the cloud. The satellite's Deployment Agent is responsible for allocating and managing the computational functions and microservices received from the cloud to install them in devices of the satellite infrastructure with idle computational resources, aiming to fulfill the QoS required for the overall service while achieving a fair allocation of resources in the satellite infrastructure. In this context, the Deployment Agent creates a cluster for each service, representing all idle devices selected to receive and install the binaries of the microservices. Next, the Deployment Agent forwards the cluster information to the DNS servers to update the IP addresses of the local Service Agents that can receive local requests for that service.

4.3 Service Manager

To ensure QoE, the service manager, the first computational function in the chain of functions that describe the service, performs traffic monitoring, for example, the delay between the microservices that are part of the chain. If any monitoring metric is unsatisfied, the service manager sends this information to the responsible Deployment Agent. Upon receiving a message from the service manager, the responsible Deployment Agent can reallocate the chaining of the microservices to guarantee that the service requirements are met. In the proof-of-concept, the criterion used for reallocating the computational functions related to microservices is the proximity to the user or the data source, depending on the service.

In this way, the Service Manager plays the role of a local manager who ensures the quality of the service provided. In the proposed architecture, all the business logic of a service must be implemented in the Service Manager function and the processing in the other functions that will be linked to the Service Manager. Thus, a user or the data plane of an infrastructure has no knowledge of the computational functions that are part of the service. The only information available to them and necessary is the Service Manager of the service.

4.4 Reallocation of services

When bottlenecks are detected by the monitor installed in the Deployment Agent, computational functions may be reallocated inside the satellite infrastructure, followed by the update of DNS servers in order to allow local users to find the closest Service Agent to request the execution of a service. Similarly, the cloud Deployment Agent can request the satellite Deployment Agent to drop some services installed on the satellite infrastructure, after which the service may be stopped or transferred back to the cloud.

During reallocating computational functions, the Deployment Agent forwards the binaries to the devices determined by the selection model. During this process, the running functions remain active, the forwarded binaries are copies of the original functions. During the activation of the functions on the new devices, the head functions, Service Manager, perform the context synchronization, and only after that the previous functions are deactivated.

5 Evaluation

The serverless-cloud-to-thing was implemented using the NS3 network simulator, and the computation functions using Wasm. The aim of this proof of concept is to evaluate the redefinition of the service's behavior in real-time and independently, to analyze the chaining of computational functions and their reallocation at runtime. For this, the bandwidth of all PoPs was reduced to force the reallocation of the computational functions of the services at various times during the simulation. The source code is available on GitHub⁵. The NS3 runs on Ubuntu 20.04, kernel 5.4, Intel Core i5-7300U, and 8 GB RAM.

For the proof-of-concept, three services were implemented using the Wasm Module adaptation. With the first service application, APP1, for each packet the user sends, one packet is returned; this service comprises three computational functions. With the second service application, APP2, for each packet the user sends, two packets are returned; this application comprises two computational functions. Finally, with the third service application, APP3, four packets are returned for each packet sent by the user; this application comprises five computational functions.

The used topology is presented in Figure 6, composed of four users (users in this proof-of-concept can be smartphones, drones and/or IoT devices.), three PoPs, a satellite, a cloud, and a cluster structure encompassing devices from a third-party entity connected to the satellite infrastructure.

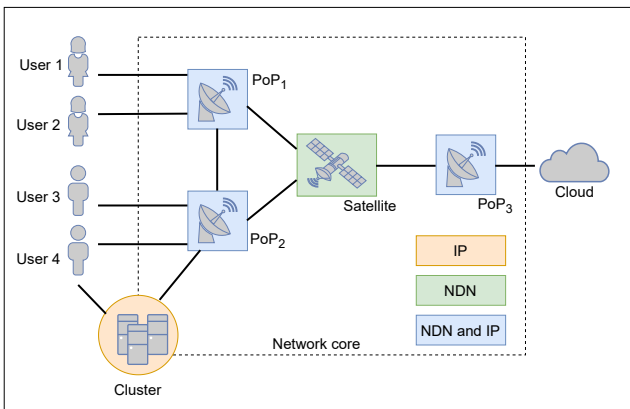


Figure 6. Evaluation network scenario.

Two simulation scenarios are configured, scenario A and scenario B, further described in the following sub-sections. In each scenario, the total transfer rate passing through the PoPs' interface at any given time is considered for the reallocation of computational functions. Figure 7 and Figure 8

show several different measurements related to: a) the sum of in/out flow on the interfaces connecting the satellite to PoP₁ and PoP₂; b) the sum of in/out flow on the interface connecting PoP₃ to the satellite. Finally, c) the sum of in/out flow on the interfaces connecting the cluster to User 4 and PoP₂.

5.1 Scenario A

For scenario A, 4 user nodes are considered, the defined configuration is user 1, and user 2 has APP1, user 3 has APP2, and user 4 has APP3 configured. The main objective of this scenario is to analyze the dynamic reconfiguration of computational functions on devices with ossified computing resources that are not normally part of the network infrastructure. To do this, applications that require greater bandwidth are installed on users 3 and 4 to overload the connection interface with PoP₂ and PoP₃.

Figure 7 shows each user's throughput. It is important to observe that in the initial moment of the simulation, the computational functions are allocated almost completely in the cloud. This observation can be noticed by the throughput of little more than 23% in b) at instant 2s. It should also be observed that no function is installed in the cluster (throughput at 0% for all users). During the simulation, the computational functions start to be deployed in real-time as expected. This can be concluded by observing the linear throughput distribution starting at time 12s in graphs a), b), and c).

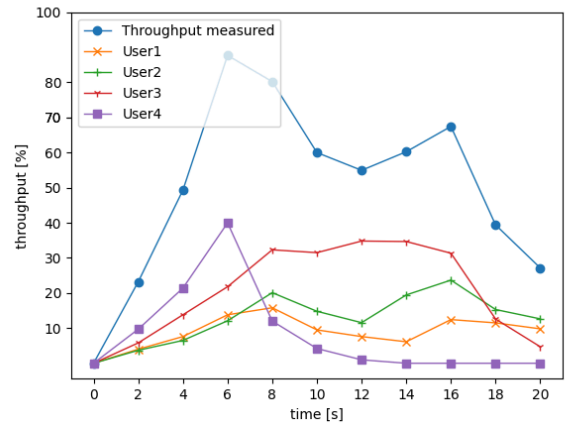
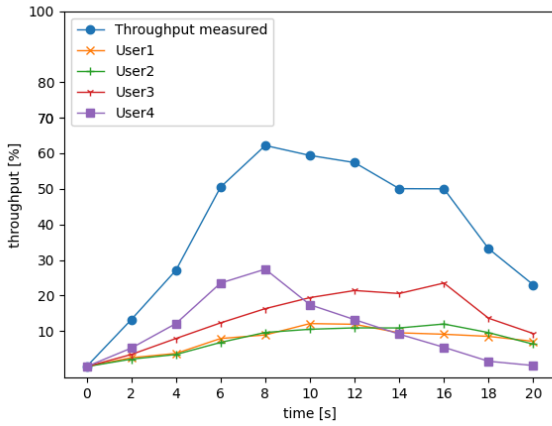
The throughput results generated by the simulations confirm that the computational functions are dynamically allocated in real time according to the throughput measured until that moment. The results indicate a dynamic behavior and a real-time configuration of the computational functions. It is possible to state that the QoE and QoS required by the user and network operator, respectively, may be optimized using a manager with a more robust algorithm based on Artificial Intelligence, such as recursive neural networks. Furthermore, the results obtained by the proof-of-concept show that the computing functions have been moved to locations where the bandwidth parameter (used by the control application) can be optimized in the network infrastructure. This indicates that the continuum computing model based on the chaining of computational functions and with mobility could be an alternative solution for next-generation network infrastructure.

5.2 Scenario B

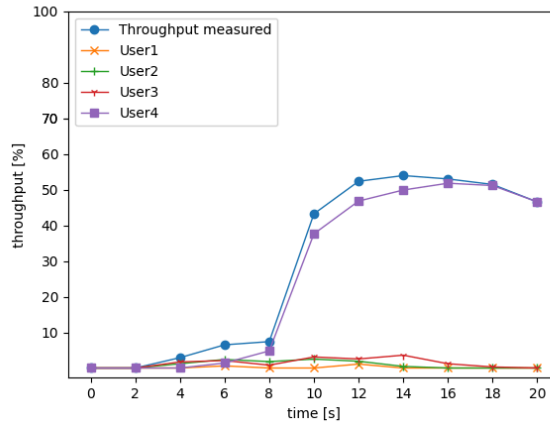
In scenario B, the following configuration is defined: User 1 and User 2 has APP3, User 3 has APP1 and User 4 has APP2 configured. The main objective of this scenario is to explore the behavior of the proposed project with load stress on PoP₁, in which users do not communicate with a cluster without going through PoP₁ itself.

Figure 8 shows the throughput for each user. Note that up to time 2s, a similar behavior to the one observed in Scenario A is obtained. However, from time 4s on, the behavior starts to diverge from the one observed in Scenario A, specifically concerning the cluster's behavior when PoP₁ is stressed. It is

⁵<https://github.com/aassilva/cloud-to-thing>



a) in/out flow on the satellite's interfaces to PoP₁ and PoP₂. b) in/out flow on the interface connecting PoP₃ to the satellite.



c) in/out flow on the cluster interfaces connecting User 4 and PoP₂.

Figure 7. Throughput measured in Scenario A.

possible to notice an almost linear throughput distribution between the cloud and the satellite a) and b), respectively, from instant 8s on, which shows that the deployment agent tried to balance the levels of operation between the cloud and the satellite by distributing the computational functions among themselves.

Curiously, the application deployed all the computational functions that attend to User 3 and User 4 in the cluster. With this, there was a flow deviation, as it is possible to see in a) and b) from instant 12s without the presence of flows for User 3 and User 4. With this, and with the data in c), it is possible to state that all the computational functions of APP3 were installed in the cluster. Another curious fact is that it is possible to observe flows of User 1 and User 2 in the cluster (see in c) at the instant between 8s and 16s. This can be explained by observing Figure 9, which shows a higher packet loss, possibly due to the overload in PoP₁. This observation suggests that the Deployment Agent observed the service degradation and proactively tried to reallocate the functions.

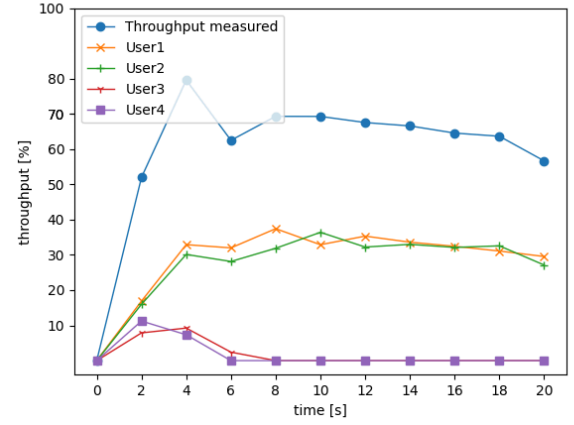
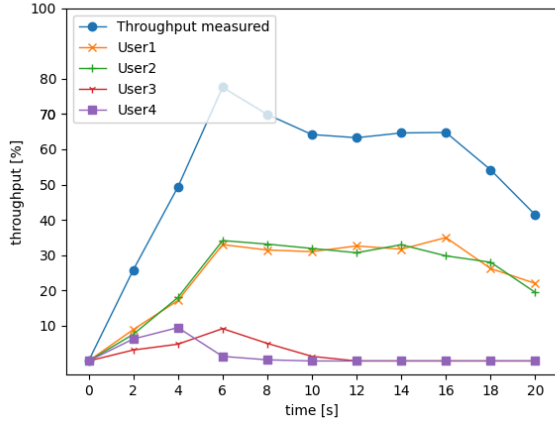
6 Discussion

With the increasing adoption of applications in 5G and 6G, the realization of the computing continuum concept on a

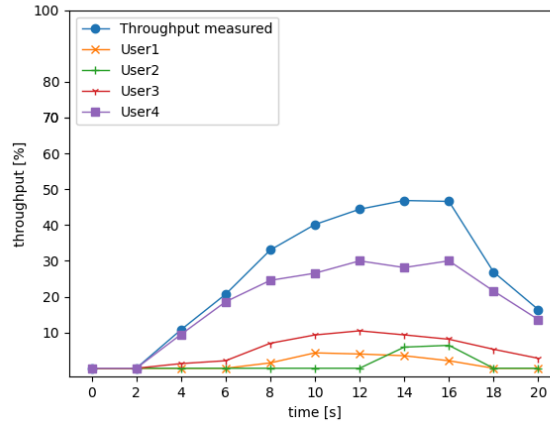
large scale may soon become a reality. Nevertheless, there is a long road ahead to decouple computing from the cloud and dedicated devices to computing on any device (cloud-to-thing). In this context, serverless-cloud-to-thing provides valuable insights into an architecture that can support cloud-to-thing in a model based on the delegation of serverless microservice.

6.1 Redefinition of service behavior in real-time

The performance of a service is considered correct if there is a signature of all computational functions in the payload of a package and the packet has the expected size. Figure 9 shows scenarios A and B's packet loss and return service error rates. The service error rate is relatively low and was almost stable in both scenarios, which can be explained by the service load distribution in the infrastructure. Possibly, this rate could be lower if the application had a more robust service manager, which will be explored in future implementations. Additionally, it is possible to observe that scenario B had a much higher packet drop than scenario A. As this packet drop had no real impact on the QoS performed by the application, it is possible to conclude that the packet losses



a) in/out flow on the satellite's interfaces to PoP₁ and PoP₂. b) in/out flow on the interface connecting PoP₃ to the satellite.



c) in/out flow on the cluster interfaces connecting User 4 and PoP₂.

Figure 8. Throughput measured in Scenario B.

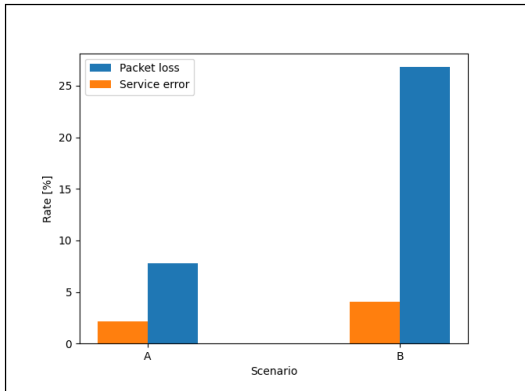


Figure 9. Packet loss and service error rate in Scenarios A and B.

happened on the links connecting User 1 and User 2 to PoP₁.

It is possible to state that the proof-of-concept presented an expected performance. As mentioned, future implementations may improve the performance and make the solutions more robust, particularly, AI and Graph Neural Network (GNN) algorithms can be explored in the service manager implementation, making the service more resilient to real-time dynamic changes. Furthermore, future research aims to study a more intelligent data plane with the support of GNN algorithms. Moreover, scalability can be enhanced through controller replication techniques. Additionally, alternative

models for controller implementations could be explored in future research, such as deploying the controller as a chain of computational functions distributed across the network.

6.2 Independence of computational functions

This work anticipates that future serverless microservice solutions will need to consider design options for the direction of independent computational functions and execution in a simple interface. Take containers, for example. One can use containers for more generic execution implementations at the operating system level. As a side effect, it introduces some overhead that may not be suitable for applications that require low-latency response or hardware platforms with limited resources, such as those served by edge computing environments. One can also look at 'heavier' solutions such as hypervisors, excluding the possibility of using embedded, domain-limited devices, etc. Similarly, different devices must be able to perform different functions because they have different storage, caching, aggregation, and data processing capabilities.

Depending on the services and heterogeneity of the network devices, multiple applications can perform complex services in the network core, with resource allocation and ser-

vice chain priorities being defined. In addition, the large images result in fewer RAM and disk functions, higher network usage (for downloading the images), and longer cold-start time, all of which affect overall network performance.

7 Closing Remarks

This article explored the serverless, cloud-to-thing continuum, addressing the growing demand for low-latency, high-bandwidth computing services in 5G and 6G networks. The proposed model takes advantage of serverless architecture to deploy computing functions on a variety of devices, including satellites and drones, in addition to traditional servers, allowing for a more flexible and distributed computing environment. Furthermore, by adopting an SDN approach and taking advantage of NDN at the data plane, the proposed model facilitates the dynamic deployment and management of computing functions across the network infrastructure. WebAssembly technology is used to implement computing functions, ensuring efficiency and compatibility between devices.

The results obtained in simulation by the serverless-cloud-to-thing proof-of-concept indicate that the deployment of microservices in the cloud-to-thing continuum orchestrated by a Deployment Agent dynamically allocates resources based on service requirements and network conditions. Real-time monitoring and reconfiguration of services ensure adaptability to changing network conditions. Moreover, the reported experience with serverless-cloud-to-thing allowed the identification of other promising research directions. One is to develop models for maintaining Forward Information Base (FIB) entries, which can be done using a name-based routing protocol. Another involves proposing an improvement of SDN reliability in the presence of intermittent connectivity that can degrade the deployment agent. One possible solution may involve developing a distributed system built as chains of microservices, each responsible for a particular control task. Interacting with the community using serverless-cloud-to-thing, the intention is to discuss directions to provide insights into an efficient computing continuum model suitable for the new 5G and 6G networks.

Declarations

Acknowledgements

This study is partially supported by CNPq, Project 309505/2020-8 and FINEP, Project ref. 2904/20.

Authors' Contributions

AS contributed to the conception of this study and also performed the experiments. PM, DR, EC, JC, and EF are this manuscript's main contributors and writers. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

References

- Afrin, M., Mahmud, M. R., and Razzaque, M. A. (2015). Real time detection of speed breakers and warning system for on-road drivers. In *proceedings of the IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE 15)*. IEEE. DOI: 10.1109/WIECON-ECE.2015.7443976.
- Al-Omaisi, H., Sundararajan, E. A., Alsaqour, R., Abdullah, N. F., and Abdelhaq, M. (2021). A survey of data dissemination schemes in vehicular named data networking. *Elsevier Vehicular Communications*. DOI: 10.1016/j.vehcom.2021.100353.
- Cerny, T., Abdelfattah, A. S., Bushong, V., Al Maruf, A., and Taibi, D. (2022). Microservice architecture reconstruction and visualization techniques: A review. In *proceedings of the IEEE International Conference on Service-Oriented System Engineering (SOSE 22)*, pages 39–48. IEEE. DOI: 10.1109/SOSE55356.2022.00011.
- Cheng, B., Fuerst, J., Solmaz, G., and Sanada, T. (2019). Fog function: Serverless fog computing for data intensive iot services. In *proceedings of the IEEE International Conference on Services Computing (SCC 19)*. IEEE. DOI: 10.1109/SCC.2019.00018.
- Čilić, I., Žarko, I. P., and Kušek, M. (2021). Towards service orchestration for the cloud-to-thing continuum. In *proceedings of the 6th International Conference on Smart and Sustainable Technologies (SpliTech 21)*, pages 01–07. IEEE. DOI: 10.23919/SpliTech52315.2021.9566410.
- Dogani, J., Namvar, R., and Khunjush, F. (2023). Auto-scaling techniques in container-based cloud and edge/fog computing: Taxonomy and survey. *Computer Communications*. DOI: 10.1016/j.comcom.2023.06.010.
- Goniwada, S. R. (2022). Serverless architecture. In *Springer Cloud Native Architecture and Design*. Springer. DOI: 10.1007/978-1-4842-7226-8_7.
- Gusev, M. (2021). Serverless and deviceless dew computing: Founding an infrastructureless computing. In *proceedings of the IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC 21)*. IEEE. DOI: 10.1109/COMPSAC51774.2021.00273.
- Kaur, N. and Mittal, A. (2021). Fog computing serverless architecture for real time unpredictable traffic. In *IOP Conference Series: Materials Science and Engineering*. IOP Publishing. DOI: 10.1088/1757-899X/1022/1/012026.
- Kjorveziroski, V. and Filiposka, S. (2023). Webassembly orchestration in the context of serverless computing. *Journal of Network and Systems Management*, 31(3):62. DOI: 10.1007/s10922-023-09753-0.
- Król, M. and Psaras, I. (2017). Nfaas: named function as a service. In *ACM Conference on Information-Centric Networking*. DOI: 10.1145/3125719.312572.
- Laghari, A. A., Jumani, A. K., and Laghari, R. A. (2021). Review and state of art of fog computing. *Springer Archives of Computational Methods in Engineering*. DOI: 10.1007/s11831-020-09517-y.
- Landmark, L., Larsen, E., and Kure, O. (2018). Traffic control in a heterogeneous mobile tactical network with autonomous platforms. *Tech-*

- nical report, Norwegian Defence Research Establishment, Kjeller. Available at: <https://ffi-publikasjoner.archive.knowledgearc.net/handle/20.500.12242/2485>.
- Li, Z., Guo, L., Cheng, J., Chen, Q., He, B., and Guo, M. (2022). The serverless computing survey: A technical primer for design architecture. *ACM Computing Surveys (CSUR)*, 54(10s):1–34. DOI: 10.1145/3508360.
- Mahmud, R. and Toosi, A. N. (2021). Con-pi: A distributed container-based edge and fog computing framework. *IEEE Internet of Things Journal*. DOI: 10.1109/JIOT.2021.3103053.
- Mekbungwan, P., Pau, G., and Kanchanasut, K. (2022). In-network computation for iot data processing with activendn in wireless sensor networks. In *proceedings of the 5th Conference on Cloud and Internet of Things (CIoT 22)*. IEEE. DOI: 10.1109/CIoT53061.2022.9766613.
- Patros, P., Spillner, J., Papadopoulos, A. V., Varghese, B., Rana, O., and Dustdar, S. (2021). Toward sustainable serverless computing. *IEEE Internet Computing*. DOI: 10.1109/MIC.2021.3093105.
- Rosário, D., Schimunek, M., Camargo, J., Nobre, J., Both, C., Rochol, J., and Gerla, M. (2018). Service migration from cloud to multi-tier fog nodes for multimedia dissemination with qoe support. *Sensors*, 18(2):329. DOI: 10.3390/s18020329.
- Sarkar, S., Wankar, R., Srirama, S. N., and Suryadevara, N. K. (2019). Serverless management of sensing systems for fog computing framework. *IEEE Sensors Journal*. DOI: 10.1109/JSEN.2019.2939182.
- Verginadis, Y., Apostolou, D., Taherizadeh, S., Ledakis, I., Mentzas, G., Tsagkaropoulos, A., Papageorgiou, N., and Paraskevopoulos, F. (2021). Prestocloud: a novel framework for data-intensive multi-cloud, fog, and edge function-as-a-service applications. *Information Resources Management Journal (IRMJ)*. DOI: 10.4018/IRMJ.2021010104.