Received: 12 April 2019 | Revised: 26 April 2019 | Accepted: 30 April 2019

DOI: 10.1002/itl2.106

SPECIAL ISSUE ARTICLE

WILEY

End-to-end elasticity control of cloud-network slices

Alisson Medeiros[1] | **Augusto Neto**[1,2] | **Silvio Sampaio**[1] | **Rafael Pasquini**[3] | **Javier Baliosian**[4,5]

[1] Department of Informatics and Applied Mathematics (DIMAp), Federal University of Rio Grande do Norte (UFRN), Natal, Brazil

[2] Instituto de Telecomunicações (IT), Aveiro, Portugal

[3] Federal University of Uberlândia (UFU), Uberlândia, Brazil

[4] Universidad de la República de Uruguay (UdelaR), Monte Video, Uruguay

[5] Universitat Politècnica de Catalunya (UPC), Barcelona, Spain

Correspondence
Alisson Medeiros, Federal University of Rio Grande do Norte (UFRN), Natal, RN, Brazil.
Email: alisson@ppgsc.ufrn.br

The design of efficient elasticity control mechanisms for dynamic resource allocation is crucial to increase the efficiency of future cloud-network slice-defined systems. Current elasticity control mechanisms proposed for cloud- or network-slicing, only consider cloud- or network-type resources respectively. In this paper, we introduce the elaSticity in cLOud-neTwork Slices (*SLOTS*) which aims to extend the horizontal elasticity control to multi-providers scenarios in an end-to-end fashion, as well as to provide a novel vertical elasticity mechanism to deal with critical insufficiency of resources by harvesting underused resources on other slices. Finally, we present a preliminary assessment of the SLOTS prototype in a real testbed, revealing outcomes that suggest the viability of the proposal.

KEYWORDS
5G, cloud-network slicing, end-to-end slicing, resource elasticity

1 | INTRODUCTION

Our previous research[1] highlights the benefits and challenges of orchestrating the integration of softwarization- and cloudification-like resources, considering the requirements for achieving quality-assured 5G services in Ultra-Dense Networking scenarios. Our proposed approach leverages slicing techniques to become capable of orchestrating end-to-end multi-dimensional virtual partitions over shared physical infrastructure, which are isolated and independent by definition, at runtime. The novelty of our approach is the ability to provide orchestration for both network- and cloud-type resources to afford end-to-end cloud-network slices, which have been proven to be viable, reasonable and efficient. In cloud-network slice-defined systems, intelligent mechanisms capable of performing automatic and autonomous elasticity, for both cloud and network resources, are extremely important, since they are needed to enable a common information model that explicitly provides elasticity policies to accomplish the scheduling of resources in both environments.

Although our previous research establishes end-to-end cloud-network slices in a flexible way, elasticity control mechanisms were not available. Beyond state of the art, in this work we elicit requirements we believe must be met to enable efficient elasticity in cloud-network slice-defined systems: (a) to carry out elasticity, taking into account both network- and cloud-type resource conditions in an integrated way and at runtime; (b) to implement an end-to-end fine-grained view per cloud-network slice instance, with enclosing services (eg, Virtual Network Functions [VNFs] and software applications); and (c) to deploy statistic-based elasticity control of resources, that necessarily considers usage levels in each cloud-network slice instance and enclosing service.

Part of this research was supported by the H2020 4th EU-BR Collaborative Call, under the grant agreement no. 777067 (NECOS—Novel Enablers for Cloud Slicing), funded by the European Commission and the Brazilian Ministry of Science, Technology, Innovation, and Communication (MCTIC) through RNP and CTIC. This study was also financed in part by both the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior*—Brazil (CAPES)—Finance Code 001, and the *Conselho Nacional de Desenvolvimento Científico e Tecnológico* (CNPq). This research was carried out under the Research Group on Future Internet Service and Applications (REGINA) Laboratory.

In this paper, we propose the ela**S**ticity in c**LO**ud-ne**T**work **S**lices (SLOTS), which suits all the aforementioned requirements towards orchestration of cloud-network slice-defined systems and, thereby, foster an adaptable control plane that leverages emerging technologies and concepts to be able: (a) to provide vertical elasticity as a service in a per cloud-network slice instance control basis, while (b) keeping both cloud- and network-type resources towards assuring service quality over the cloud-network instance slice lifetime. SLOTS is implemented in a prototype approach for real evaluations and reveals outcomes that suggest SLOTS to be a viable approach to orchestrate end-to-end cloud-network slices at runtime.

The rest of this paper is organized as follows: Section 2 provides an analysis of the most relevant related works. Section 3 presents SLOTS. Section 4 describes implementation and experimental evaluation. Finally, Section 5 provides conclusions and remarks for future research.

2 | RELATED WORK

Resource elasticity has been widely explored in cloud computing and, recently, in the context of 5G networks. Some research efforts explore new elasticity features for network slicing in 5G networks.[2] Nevertheless, research on how network elasticity can be integrated into cloud elasticity is still open. Also, most of the cloud computing elasticity solutions are based on fixed thresholds (a.k.a, reactive elasticity approach), whereas the majority of authors claim threshold-based technique performance is highly dependent on the selected parameters.[3] Despite the fact of reacting after events, the reactive elasticity approach is still widely explored by many commercial and academic initiatives. One reason is its ease of implementation, as it adopts a stochastic resource computing approach (eg, 2-fold, 3-fold and the like, or fixed percentages rates, 20%, 50%, etc.) to scale up and down targeted resources. Another reason is the fact that large-scale cloud providers such as Google, AWS, and others have no urgency on avoiding an eventual overprovision of resources situation on the due to their abundance of resources.

A number of academic proposals have exploited the reactive elasticity approach. For example, the work in Reference 4 presents the *ACCRS* framework, which adopts static thresholds to trigger reactive elasticity. Several other proposals present similar restrictions under different scenarios, such as Reference 5: which proposes *ElasticDocker* and Reference 6 which proposes (*DoCloud*), both proposal to scale up and down docker containers, considering unlimited resources[7]; works with QoS metrics to support unlimited elasticity[8]; which proposes *Helpar*, an hybrid model for elasticity, but also based on static thresholds and unlimited resources. As can be seen, none of these proposals address critical scenarios in terms of the amount of resources available to perform elasticity, and all follow stochastic models for resource-computing.

In the commercial side, different elasticity mechanisms are employed. The Amazon EC2 AWS platform offers a auto-scaling technique called *Target Tracking Scaling*, which is able to dynamically provide horizontal elasticity for a specific resource (eg, CPU, RAM, bandwidth).[9] The *Google Cloud Platform* supports a auto-scaling mechanism called *Multiple Policies* (MP), which uses multiple elasticity policies at different levels.[10] The *MP* mechanism is also restricted to horizontal elasticity. In the same way, container management systems such as Kubernetes[11] and Docker Swarm[12] also adopt horizontal elasticity for containerized applications. A common characteristic of all these elasticity mechanisms is the fact they are restricted to horizontal elasticity, assuming that the infrastructure provider is always capable to offer new resources to meet the elasticity demand, following stochastic models.

Finally, the elasticity mechanisms hereinabove described lack to address specific scenarios imposed by cloud-network slices, in which the underneath multi-provider/multi-domain infrastructure implies an end-to-end elasticity control. Regarding this, SLOTS aims to extend the elasticity control over resources from multiple providers by applying vertical and horizontal elasticity at the same time and also using elasticity as a service, seeking to provide different options to be used by cloud-network slices. In addition, stochastic resource-computing scheme impact in several issues, varying from waste of resources, poor response time, and multiple attempts to meet stability. In this sense, SLOTS introduces an elasticity mechanism that can be used to address critical situations of resource insufficiency where the stochastic approaches fail. SLOTS proposes a solidarity approach to deal with this issue in which slices having unused resources can act as donner in the benefice of others that need more resources. Finally, SLOTS is intended to small- and medium-sized cloud and network providers that require a fine-grained control of the resources.

3 | ELASTICITY CONTROL FOR CLOUD-NETWORK SLICES

By combining softwarization and cloudification technologies, slicing has emerged as a viable ecosystem to leverage new business models imposed by the 5G scenario. At the same time, slices independently provided a partitioned view of physical

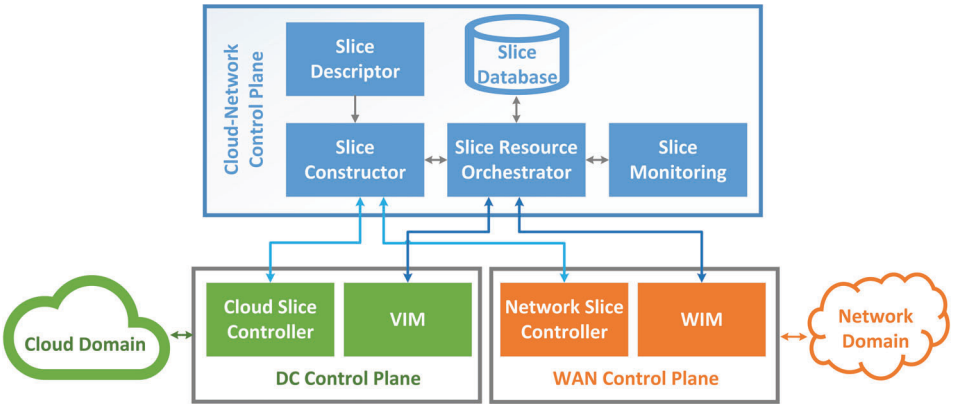

FIGURE 1 Functional architecture of the SLOTS approach

and virtual resources, thereby allowing instances of virtualized networks to run on top of a partitioned and dedicated infrastructure. Based on the new requirements, use cases and challenges of the 5G scenario, the SLOTS proposal was designed towards the native integration of cloud and network resources. SLOTS extended the orchestration of network slicing to the cloud, thereby allowing both scenarios to perform the slicing in a collaborative and coordinated way.

In this context, the cloud-network slicing approach implemented by SLOTS provides an end-to-end view, involving a set of physical (cloud servers, network nodes), and virtual (Virtual Machines (VMs) containers, VNFs, virtual links) components. Such a view enables mechanisms to be developed to support the management of multiple cloud infrastructures and networks in a collaborative way. Figure 1 depicts the functional architecture of the proposed SLOTS approach. As shown in Figure 1, the SLOTS proposal encompasses the design of a functional architecture by presenting different functional blocks and internal subsystems that interact through specific interfaces to perform different functions. The main blocks of the SLOTS architecture are: the Cloud-Network Control Plane, DC, and WAN Control Plane. Next, we give a brief description of each block and its components, introducing the proposed mechanism for cloud-network vertical elasticity.

Cloud-Network Control Plane is a key building block of SLOTS architecture, and contains a set of functional components that interact to orchestrate the lifecycle of cloud-network slices in different technology domains. The elements of Cloud-Network Control Plane are summarized as follows: The Slice Constructor assumes the task of creating a whole end-to-end cloud-network slice across multiple technology domains from the relevant constituent slice parts, taking into account the slice template information to check available resources in the participating providers; The Slice Resource Orchestrator (SRO) component is responsible for combining the slice parts that make up an end-to-end cloud-network slice, orchestrating slices and service elements over slices parts. The main functions of the SRO are: (a) Orchestrating the end-to-end Slice Lifecycle, (b) Decommissioning the end-to-end Slice, (c) Instantiating virtual resources for the services, (d) Performing the Elasticity. The SRO performs the Elasticity, which in turn directly deals with the Virtual Infrastructure Manager (VIM) and WAN Infrastructure Manager (WIM) so that modifying the allocation of the virtual resources of each selected slice. The Slice Monitoring component provides a uniform abstraction layer over the monitoring subsystems in heterogeneous VIMs and WIMs that are part of an end-to-end cloud-network slice. Finally, the Slice Database maintains full knowledge of all activated cloud-network slices in the SLOTS ecosystem (eg, DC slice part or WAN slice part, and on the services running on each slice).

In order to provide advanced connectivity and resource control capabilities for cloud-network slices, the Data Center (DC) Control Plane contains the necessary components to handle all of the cloud infrastructure needed by SLOTS. Similarly, the WAN Control Plane provides the capabilities of network infrastructures to enable connectivity between different slice parts. These two control planes reside specifically in the cloud and network providers and the components are described in the sequence.

The Cloud Slice Controller component participates in each Data Center Provider and is responsible for dynamically creating a data center slice. To this end, the Cloud Slice Controller supports the following operations: (a) Resource Management, (b) Cloud Slice Creation, (c) VIM Deployment, and (d) Cloud to Network Connectivity. The Network Slice Controller component participates in each network provider to dynamically creating a network slice. A network slice is a virtual network infrastructure (eg, nodes, links, VNFs, etc.) properly connecting two cloud slice parts. To create a network slice, the Network Slice Controller carries out the following operations: (a) Network Management, (b) Network Slice Creation, (c) Resource Provision, and (d) WIM Deployment.

The primary goal of the elasticity architecture proposed in this paper is to guarantee the performance of the cloud-network slices as much as possible while maintaining committed levels of service level agreement (SLA) and application performance. A reactive model triggers the decision-making process, whereby the elasticity control mechanism follows a unique statistic-based

decision scheme (in contrasts to the stochastic model, as raised in Section 2). Moreover, this proposal distinguishes itself from the related work through its distributed architecture that can simultaneously act in the different cloud and network providers so that applying vertical elasticity control functions for readjusting (scale up and down) resources at CPU, RAM, and network levels to each of the activated cloud-network slices.

On the basis of page-limit restrictions, the proposed elasticity control workflow encompasses the following steps: (a) Monitoring: responsible for collecting metrics from the infrastructure providers composing a slice and delivering such metrics to the elasticity module; (b) Reasoning: in charge of computing which slice resources will be subjected for re-sizing, as well as the new amount patterns each selected resources will be scaled up (augmented) or down (shrank) to suit respective SLA commit; and finally, (c) Enforcement: the process responsible for enforcing the changes in the slice, but also for the intelligence about the proper moment to enforce the required adaptations. Initially, a monitoring stack is used to gather slice metrics. When the resource use of a slice is marked as critical, for example, on matching a predefined maximum usage rate of the current assigned amount, the SLOTS elasticity control function is triggered. Firstly, the resource selection function is invoked for determining the required resources (CPU, memory, and/or network) to suit the slice's SLA. Afterwards, the resources selection function checks if resources are available in the system and, if so, the resource discovery function starts the resource-computing task as follows. Slices are classified as either *beneficiaries* or *donors*. The former (Beneficiary) are those subjected to obtain additional resources from other slices, the donors, which in turn are able to cede a fraction of their residuals (ie, underused resources, meaning the difference between assigned, and under instant use). SLOTS maintain a list of potential donor slices, entailing only those featuring residual resources for optimization perspectives. In order to obtain the amount of the required resource for scaling up the beneficiary cloud network slice, SLOTS breaks forcibly this amount into a number of parts matching the units of listed donor slices, equally. To avoid slice starvation, each cloud-network slice instance is set with both minimum (cannot shrink below) and maximum (cannot exceed) limits of cloud- and network-type resources. On summing all of the resource parts shrank from the donor slices, SLOTS augments the beneficiary slice by the resulting amount accordingly. This statistic-based resource-computing model of SLOTS raises as unique and distinguishes itself from the related work significantly.

4 | IMPLEMENTATION AND EVALUATION OF THE SLOTS PROPOSAL

This section presents aspects concerning both SLOTS prototyping and evaluation tests. The prime goal of the prototyping stand to carry a proof-of-concept in the SLOTS architecture, whereas the evaluation tests mainly addresses to assessing the resulting impact that both stochastic-based and statistic-based resource-computing models take in affording cloud-network slicing level elasticity control functions under high resource-constrained conditions (when all DCs become fully saturated). A real testbed was set for accurate-enhanced assessments. The methodology used considers two set of experiments participating in the tests. The Regular Elasticity Experiment implements a stochastic-based resource-computing approach (as raised in Section 2), which scales up and down in a fraction of 20% (augmenting) and 10% (shrinking) of current resource patterns respectively. The SLOTS Elasticity Experiment adopts the same resource-computing approach of the Regular Experiment, but on detecting resource saturate condition, it then automatically switches to the statistic-based approach shown in Section 3.

Three cloud-network slices are set in each of the experiment: the Slice #1 represents a FIWARE-supported IoT Machine-To-Machine (M2M) application serving a maximum rate of 5 Mbps, through which a Mosquitto server provisions publish-subscribe operations. The Iperf tool is used to provision UDP flows to slices #2 and #3; both slices #2 and #3 consume UDP-like streaming flows at a maximum rate of 7 and 3 Mbps respectively, representing Over the top multimedia content. In order to denote per-slice resource use dynamic behavior, both computing (CPU and RAM) and network (bandwidth) parameters variations are set along the course of the experiment (10 minutes), which can be summarized in the following. At the beginning of the experiment (instant time 0 second), the UDP-like flows start within slices #2 and #3 and gradually scale up until matching their respective maximum transmission rates (ie, 7 and 3 Mbps respectively, at instant time 30 seconds). In regards of the TCP-like flow (served by slice #1), the number of clients increase progressively until reaching a total of 500 clients (instant time of 3 minutes). This behavior simulates a situation in which several IoT devices are associated with a single MQTT Broker.

In light to meet the main goal of testbed evaluation, in terms of carrying out both prototyping and evaluation tests, key performance indicators (namely bandwidth, CPU, and RAM usage patterns) are fetched all over the experiment time. In order to allow the DC rack servers running through resource-saturated condition, and thus enable our assessments, the virtual machines are set to operate in limited computing and networking conditions.

On the basis of the prototyping tests, the proof-of-concept confirms the SLOTS architecture fully comply with the Figure 1, as well as with the Section 3 definitions. In the evaluation tests, both experiments contain three slices with the following

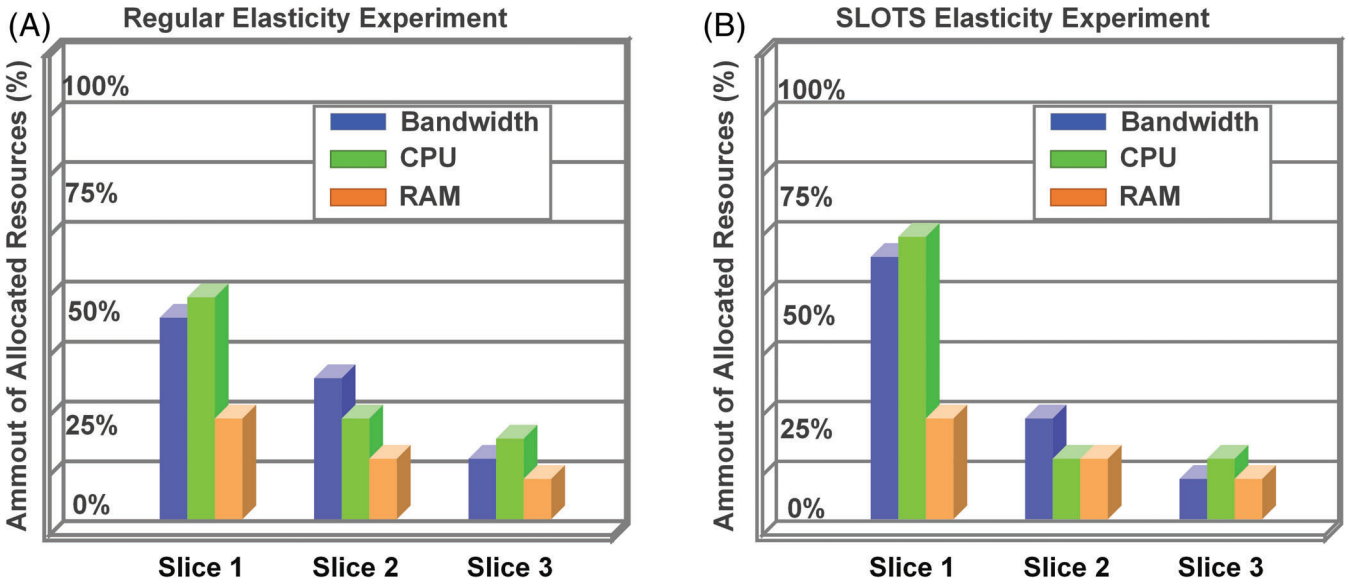

FIGURE 2 Prototyping outcomes of regular and SLOTS elasticity experiments

resource allocations in terms of bandwidth, CPU and RAM: 25%, 45%, and 25% for the slice #1; 35%, 25%, and 15% for the slice #2; and 15%, 20%, and 10% for the slice #3. In addition, the resources available in the system for elasticity to be performed are: 5 Mbps (25% of total bandwidth), 10% vCPU and 1GB RAM (50% equivalent). The total resources available for both experiments are: 20 Mbps bandwidth, 1vCPU (0% to 100% usage scale) and 2GB RAM. When the 500 MQTT clients perform the publish-subscribe messages, the MQTT Broker (slice #1) matches the need for elasticity control. Figure 2A shows that the Regular elasticity solution scales up to the bandwidth of slice #1 from 5 Mbps (currently allocated resources) to 10 Mbps (still insufficient to suit the SLA). Through the SLOTS approach, impact shown in Figure 2B, the bandwidth of slice #1 is allowed to scale up 65%, achieving 13 Mbps (30% more than the Regular solution) through leveraging 5 Mbps of the system, together with 5 Mbps of donor slices #2 (2 Mbps) and slice #3 (1 Mbps). The Regular Elasticity Experiment solution allows each cloud-network slice to leverage, at the end of the tests, average resources amounts of bandwidth, CPU and RAM in the following patterns respectively: 25%, 45%, and 25% for the slice #1; 35%, 25%, and 15% for the slice #2; and 15%, 20%, and 10% for the slice #3. On the other side, the SLOTS Elasticity Experiment outcomes reveals that the respective averaging patterns with regard to bandwidth, CPU and RAM are of: 75%, 75%, and 50% for slice #1; 20%, 30%, and 30% for slice #2; and 10%, 20%, and 20% for slice #3. The numerical analysis on the temporal behavior of virtual CPU resources, reveals that the Regular Elasticity solution has been provisioned from 45% to 55%, since only 10% of the vCPU residual resources are available. In contrast, SLOTS allows to re-provision resources patterns from 45% to 70%, through leveraging a total 25% of residual resources (ie, 10% from system residual resources, 10% of donor slice #2 and 5% of donor slice #3 fulfilling their respective committed limits). Due to the nature of the experiments, the RAM resources could not be provisioned.

The outstanding ability that the SLOTS statistic-based elasticity control approach allows in the evaluation tests outperforms the stochastic scheme of the Regular Elasticity Experiment, by means of successfully accomplishing an amount of 20% more elasticity control. It is worth to highlight that, from the 7.5 minutes of the experiment time, the Regular Elasticity solution becomes unable to carry out elasticity control functions due to DC's fully resource-saturated conditions.

5 | CONCLUSIONS AND FUTURE RESEARCH

In this paper, the SLOTS approach is proposed to carry out resource elasticity control functions tailored to cloud-network slicing-defined systems. The statistic resource-computing model that SLOTS applies for elasticity control is unique in orchestrating resources among donors and beneficiaries cloud-network slices, according to their current patterns along with committed rates. The SLOTS solution was assessed through real testbed prototyping for accurate insights. The proof-of-concept analysis confirms that the SLOTS elasticity control scheme complies the proposed architecture and deals with end to end cloud-network slices. Moreover, the numerical outcome analysis suggests that SLOTS provides resource management and orchestration under highly resource-saturated DC conditions, through adjusting resources among donors and beneficiaries cloud-network slices

cooperatively, taking a unique statistic resource-computing approach. The evaluation outcomes prove that SLOTS outperforms the Regular Elasticity solution through enabling to accomplish a rate of 20% more elasticity functions under the critical DC's resource conditions.

As SLOTS is an ongoing project, we will continue to focus on studying and developing additional assessments, enforcement, and control mechanisms, and tailor them to improve the re-orchestration of all the resources that support the cloud-network slices. Moreover, we need to introduce new functions to deal with horizontal elasticity.

ORCID

Augusto Neto https://orcid.org/0000-0002-9936-3770

REFERENCES

1. Carmo M, Dantas Silva FS, Neto AV, Corujo D, Aguiar R. Network-cloud slicing definitions for Wi-Fi sharing systems to enhance 5G ultra dense network capabilities. *Wirel Commun Mob Comput*. 2019;*2019*:1-17.
2. Logota E, Corujo D, Jeon S, Rodriguez J, Aguiar RL. *The 5g internet*. *Fundamentals of 5G Mobile Networks*. Chichester, West Sussex, UK: John Wiley & Sons; 2015:29-62.
3. Farokhi S, Jamshidi P, Brandic I, Elmroth E. Self-Adaptation Challenges for Cloud-Based Applications: A Control Theoretic Perspective; In: *10th international workshop on feedback computing*. Seattle, 2015.
4. Al-Sharif ZA, Jararweh Y, Al-Dahoud A, Alawneh LM. ACCRS: autonomic based cloud computing resource scaling. *Cluster Comput*. 2017;*20*(3):2479-2488. https://doi.org/10.1007/s10586-016-0682-6.
5. Al-Dhuraibi Y, Paraiso F, Djarallah N, Merle F. Autonomic Vertical Elasticity of Docker Containers with ELASTICDOCKER; In: *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*. Onolulu, Hawaii, IEEE; 2017: pp. 472–479.
6. Kan C. DoCloud: An Elastic Cloud Platform for Web Applications Based on Docker; In: *2016 18th International Conference on Advanced Communication Technology (ICACT)*. Pyeongchang, South Korea, IEEE; 2016: pp. 478–483.
7. Kumar D, Gondhi NK. A QoS-Based Reactive Auto Scaler for Cloud Environment; In: *2017 International Conference on Next Generation Computing and Information Systems (ICNGCIS)*. Jammu, India, IEEE; 2017: pp. 19–23.
8. Rosa Righi dR, Rodrigues VF, Rostirolla G, Costa AC, Roloff E, Navaux POA. A lightweight plug-and-play elasticity service for self-organizing resource provisioning on parallel applications. *Future Gener Comput Syst*. 2018;*78*(P1):176-190. https://doi.org/10.1016/j.future.2017.02.023.
9. AWS A. Amazon target tracking scaling. http://docs.aws.amazon.com/autoscaling/latest/userguide/as-scaling-target-tracking.html; 2018. Online; Accessed February 26, 2019.
10. Google. Google multiple policies scaling. https://cloud.google.com/compute/docs/autoscaler/multiple-policies; 2018. Online; Accessed February 26, 2019.
11. Google. Kubernetes horizontal pod auto-scaling. https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/; 2019. Online; Accessed February 24, 2019.
12. Docker. Docker service scale. https://docs.docker.com/engine/reference/commandline/service_scale/; 2019. Online; Accessed March 28, 2019.

How to cite this article: Medeiros A, Neto A, Sampaio S, Pasquini R, Baliosian J. End-to-end elasticity control of cloud-network slices. *Internet Technology Letters*. 2019;2:e106. https://doi.org/10.1002/itl2.106