

Enabling Elasticity Control Functions for Cloud-Network Slice-Defined Domains

Alisson Medeiros¹ Augusto Neto^{1,2} Silvio Sampaio¹ Rafael Pasquini³ Javier Baliosian⁴

¹Federal University of Rio Grande do Norte (UFRN), Natal-RN, Brazil

²Instituto de Telecomunicações, Aveiro, Portugal

³Federal University of Uberlândia (UFU), Uberlândia-MG, Brazil

⁴Universitat Politècnica de Catalunya (UPC), Barcelona, Spain

alisson@ppgsc.ufrn.br, augusto@dimap.ufrn.br, silviocs@imd.ufrn.br, rafael.pasquini@ufu.br, javier.baliosian@upc.edu

Abstract—The NECOS (Novel Enablers for Cloud Slicing) platform controls the entire lifecycle of end-to-end cloud-network slice instances, so that new services and applications can run in a broader perspective for quality-guaranteed and highly isolated multi-tenancy. Elasticity plays a vital role in ensuring the efficiency of the NECOS platform. According to the state of the art, mainstream reactive elasticity solutions are not suitable for dealing with cloud-network slicing-defined systems by the fact that they cannot trigger time-burdensome vertical elasticity on deriving resource-depleting conditions. This gap can be filled by introducing the elasticity in cLOUD-neTwork Slices (SLOTS) solution, which participates in the NECOS platform in the form of new building blocks that interwork with others to add new patterns of cloud- and networking-type resources under resource-depleting conditions. The unique contributions made by SLOTS are as follow, and go beyond the state-of-the-art by: (i) fully supporting cloud-network slice-defined systems; (ii) offering a hybrid elasticity approach; (iii) provide a sophisticated mechanism to enable a slice solidarity approach, and (iv) providing a real lab-premised large-scale testbed for a SLOTS prototyping assessment. Analysis on the evaluation results suggest that SLOTS performs well by accomplishing a higher number of elasticity events under resource-critical conditions, while slightly adding computing overhead, when compared with a widely-used stochastic-based representing solution.

Index Terms—Elasticity Control, Cloud-Network Slicing, Virtualization, Cloud Computing.

I. INTRODUCTION

The aim of the NECOS (Novel Enablers for Cloud Slicing) project [1] is to overcome the limitations of cloud computing and network-softwarized infrastructures with regard to responding effectively to the new service demands of futuristic 5G use cases. The NECOS platform features a set of building blocks that interwork to partition a set of computational (cloud data centers servers) and networking (software-defined network links) resources into constituent cloud-slice parts connected by network-slices, all of which span a shared physical infrastructure formed of several administrative domains.

From a huge list of innovative functionalities, this paper focuses on elasticity capabilities that the NECOS platform needs for addressing the new requirements imposed by cloud-network slice-defined systems. In general terms, elasticity can be defined as the ability to increase and shrink selected resources in a systematic and autonomous manner, so as to adapt to workload changes [2]. Within the scope of NECOS-enabled

cloud-network slice definitions, a new elasticity control plane plays a crucial role in allowing an efficient use of both cloud- and networking-type resources to keep cloud-network slice instances best connected and best served over time and end-to-end. In light of this, a paradigm shift is needed as a means of providing novel features that are suited to the strict requirements inherent in innovative 5G use cases.

On the basis of our previous research [3], this paper sets out a new vertical elasticity control-plane for the SLOTS approach, that can handle the NECOS-platform elasticity needed for keeping cloud-network slices active and ensuring their quality over time. SLOTS solidarity approach can improve vertical elasticity mechanisms by providing a 2-layer hybrid solution using stochastic and statistical approaches whereas mainstream reactive vertical elasticity solutions from industry consider only the stochastic approach. It is worth to mention that SLOTS vertical elasticity scheme acts to reduce the negative effects of slice service while horizontal elasticity is deployed. Horizontal elasticity is not covered in this paper.

The contributions that this work seeks to make are four-fold: (i) supporting vertical elasticity operations in cloud-network slicing-defined systems, capable of autonomously scaling up and down both computing- and networking-type resources in a coupled approach; (ii) provisioning a hybrid vertical elasticity control-plane to deal with the different resource conditions of the underlying physical infrastructures; (iii) leveraging residual resources from selected slices (designated as *Donors*) proportionally to their resource utilization ratio, and to donate them to a given slice (i.e. the *Recipient*) close to resource saturation without any violations of either the SLA or isolation; and, (iv) providing a real lab-premised cloud-network slice-defined testbed, which is used to accurately assess the feasibility and performative effects of the SLOTS vertical elasticity approach when compared with a traditional configuration (called *Regular*) that represents widely-used stochastic-based solutions.

The remainder of this paper is structured as follows. Section II provides an analysis of the most significant related work. Section III describes the SLOTS solution in details. Assessment of feasibility and performance impacts on real cloud-network slice-defined testbed experiments are discussed in Section IV. Finally, Section V provides some concluding remarks and makes suggestions for future research.

II. RELATED WORK

The literature regards elasticity control as one of the most challenging fields in cloud computing research. Cloud elasticity solutions based on reactive principles, generally adopt a stochastic-based resource computing strategy which involves taking a fixed percentage rate above the current amount (e.g., 50%, 100%, and the like) to scale selected resources up or down. Another reason is the fact that large-scale cloud providers (e.g., Google, AWS, and others) have no urgent need to avoid the risk of resource overprovisioning since they possess them in abundance.

According to Farokhi et al. [4], reactive elasticity solutions highly depend on selected parameters. Nevertheless, reactive elasticity is widely used in industry. Leading cloud providers (AWS, Azure, Google Cloud Platform - GCP) use reactive mechanisms to deploy their elasticity strategies. Amazon EC2 AWS developed a *Target Tracking Scaling* (TTS) elasticity mechanism to control VM and container scale-up and scale-down operations [5]. Similarly, GCP supports *Multiple Polices*, which is a reactive elasticity mechanism to calculate the number of instances required to be added or removed to a cluster [6]. This is due to the ease deployment of reactive elasticity solutions. Following the same trend, leading container management platforms (Kubernetes, Docker Swarm, Apache Mesos) use reactive elasticity mechanisms to enable container auto-scaling in virtualized environments. Kubernetes deploys *Horizontal Pod Autoscaler* (HPA), a mechanism to increase or decrease containers instances in specific pods taking into account CPU resources [7]. The elasticity schemes presented are unaware of resource-limited environments and also cannot act on cloud and network resources simultaneously. Besides that, the cost to maintain an overprovisioned infrastructure is paid by the customers, as when resources tend to run out no strategy is used, causing the customer to pay for more resources. Private and smaller cloud-network providers can be severely harmed by the lack of resource management mechanisms in resource-constrained scenarios.

Other schemes include similar restrictions in different scenarios, such as [8] that puts forward *ElasticDocker*, and [9] which employs (*DoCloud*), both of which are strategies to scale-up/down docker containers in a situation of unlimited resources. Authors of [10] recommend the *Helpar* hybrid elasticity mode but also based on static thresholds by assuming unlimited resources. The authors in [11] adopt an elasticity approach employing a dynamic multilevel auto-scaling method, which acts by setting dynamic thresholds at an infrastructure and application level. All these elasticity mechanisms deploy stochastic models to support either vertical elasticity, assuming that the underlying infrastructure provider is always under optimal resource conditions. Thus, none of these schemes addresses the problem of resource-depleting scenarios.

To the best of our knowledge, there is still a lack of solutions for coupling elasticity control that are tailored to scale cloud-network slices, in which the underlying multi-provider/multi-domain infrastructure involves a full end-to-end elasticity con-

trol take in to account resource-depletion scenarios. Moreover, the mainstream elasticity solutions deploy a stochastic-based resource-computing system, which affects several issues that range from the waste of resources, a reduced response time and multiple attempts to meet the demands for elasticity. We argue that stochastic elasticity-based resource-computing systems are not suitable for the cloud-network slicing scenario. This is due to the need to use federating Mobile Edge Computing (MEC) and Telco Cloud environments for multi-domain slice provisioning. In this scenario, cloud or network resource providers participating in a federation may not have unlimited resources. The need then arises for mechanisms to resource management in resource-depletion scenarios. Taking into account the aforementioned points, there is a need for elasticity mechanisms to add operations for critical resource-depleting situations in end-to-end cloud-network slice-defined scenarios to avoid the risk of time-consuming horizontal elasticity.

III. DESCRIPTION OF THE SLOTS PROPOSAL

According to the architectural specifications of the NECOS system [12], end-to-end cloud-network slice instances can be composed of cloud slice parts (i.e. containers or virtual machines running in data center servers) leveraging a customized connectivity service that is provided in the form of network slice parts. The SLOTS proposal is an answer to the required vertical elasticity control plane of the NECOS platform, which seeks to keep appropriate quality of service levels of activated cloud-network slice instances in the underlying infrastructures over time. Thus, SLOTS is designed on the basis of integrating both network-type and cloud-type resources, with the aim of systematically calculating new resource patterns to provide the best possible cloud-network slice performance in response to vertical elasticity demands. Fig. 1 depicts the SLOTS functional blocks that extend the NECOS system architecture.

The components that form a part of the DC Control Plane Tier are only found in the cloud together with the network provider infrastructures. This allows the orchestration of the resources (in their respective roles) to be carried out by the DC Elasticity Controller with the aid of the Cloud-Slice and Net-Slice Elasticity Agents, the Elasticity Agent component, and with added support from the Cloud-Slice and Net-Slice Elasticity Monitors. An architectural outline of the SLOTS building blocks, which extend to the NECOS functional architecture, is provided as follows:

- the DC Elasticity Controller resides in every infrastructure design for DC providers of the shared NECOS-enabled domains, and represents the core building block of the SLOTS approach. The DC Elasticity Controller implements the whole approach with regard to vertical elasticity decisions in terms of cloud- and networking-type resource-computing, and enforcement resources in the request task.
- the Cloud-Slice Elasticity Agent is a part of the DC provider infrastructure, and takes on the role of interfacing with the local DC Slice Controller so that it can

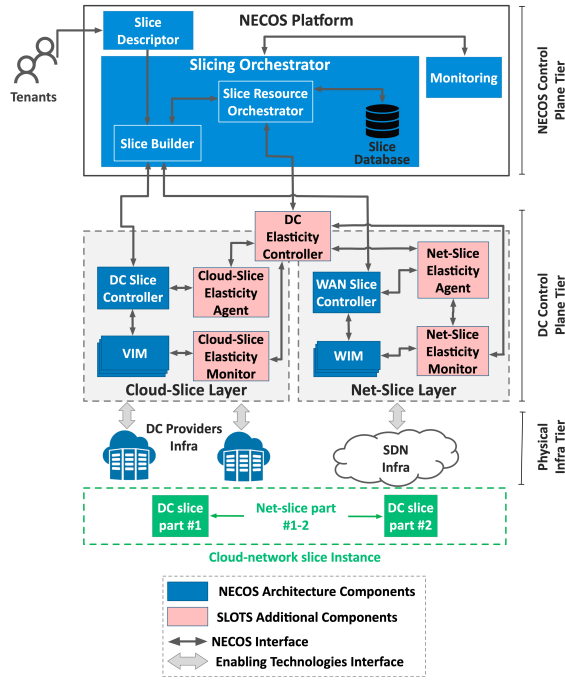


Fig. 1: SLOTS building blocks extensions to the functional architecture of the NECOS platform

provide descriptions of the cloud-slice new resource patterns (on the basis of the received template specifications) that need to be enforced in the running VIM.

- the Net-Slice Elasticity Agent, in turn, runs in the infrastructure of the network providers. On receiving the TOSCA template that describes the new network-slice resource patterns from the DC Elasticity Controller, the Net-Slice Elasticity Agent interworks with the *WAN Slice Controller*, which provides the necessary capabilities to interact with the running WIM of the local network infrastructures to setup the new network-slice definitions.
- the Cloud-Slice and Net-Slice Elasticity Monitors are both responsible for collecting the monitoring data regarding the resources allocated to the cloud-slice and network-slice parts at the local DC and network provider infrastructures respectively, and constantly checks the utilization rates of cloud-slice and network-slice active instances in real-time.

The SLOTS approach is designed as a part of the broader NECOS system architecture, whereby participating components interwork to activate the vertical elasticity workflow. The elementary architectural artifacts that compose the NECOS system architecture, provide the essential technologies that activate the SLOTS vertical elasticity workflows, as in the following example. The *Slice Descriptor* is responsible for interpreting the high-level abstraction required by the tenants. When requesting a slice, the tenant specifies the required data, and the slice descriptor generates a TOSCA template that will serve as input for the slice builder. The *Slice Resource Orchestrator* (SRO) component has the task of being

responsible for combining the slice parts that make up an end-to-end cloud-network slice instance. Moreover, the SRO is capable of managing the real-time life-cycle of slices, as well as orchestrating the service elements for the slice parts.

The *Slice Builder* adaptive orchestrated support (for leveraging the Cloudify [http://cloudify.co] tool) ensures the propagation of slice-specific attributes at the service hosting data centers, so that a whole cloud-network slice can be created for multiple technology domains from the constituent slice parts. The *DC Slice Controller* component participates in each shared DC infrastructure provider, which contains all the necessary components to create the cloud-slice parts, while updating and decommissioning tasks. The *WAN Slice Controller*, in turn, participates with each network provider to dynamically set up network-slice parts instances. This means the virtual links that ensure advanced connectivity between pairs of cloud-slice parts that are provided with the assistance of underlying WAN Infrastructure Managers (WIMs).

The *cloud-* and *net-slice Elasticity Monitoring* components, provide uniform abstraction layers in the monitoring subsystems in heterogeneous VIMs and WIMs to enable access to a set of specific slice-elasticity monitoring data in a per cloud-network slice part granularity, including resource utilization patterns. Finally, the *Slice Database* maintains a complete understanding of the cloud-network slices, including current resource patterns, along with services running in a per cloud-network slice part granularity. The next section outlines in detail the vertical elasticity mechanisms that SLOTS provides to deal with the NECOS-enabled domains.

A. SLOTS Vertical Elasticity mechanism

The main goal of the SLOTS vertical elasticity approach is to acquire the capacity to compute new resource patterns for re-sizing cloud-network slice parts (i.e., scale-up and/or scale-down the compute resources – CPU, RAM, cache, etc. These are allocated for a particular VM/container on a data center server) and/or network-slice parts (i.e., bandwidth of a tunnel in between two cloud-slices) to adapt to changes resulting from workloads when there is an increase in the service demands of a given slice. The *DC Elasticity Controller* component of the SLOTS approach embeds all the functional support of the Vertical Elasticity workflow. The SLOTS approach is designed to offer a hybrid approach that deploys two vertical elasticity modes: Stochastic and Statistical.

During the period that the slice-accommodating infrastructure is aware that the whole local infrastructure provider features a considerable amount of resources available (i.e., more than 20% of the residual resources available), the SLOTS Stochastic mode deals with the elasticity demands of active cloud-network slice instances by scaling-up and leveraging a part of them. SLOTS operates in the Stochastic mode, in the same way as for the mainstream vertical elasticity solutions (described in Section I) by either scaling up current resource patterns by 100% or reducing the additional amount by rates of 50% when aware of the impossibility of scaling-up current slice resources completely.

On receiving a trigger from the Cloud-Slice Elasticity Monitor to indicate that the DC server infrastructure matches the saturation threshold (80% by default), the DC Elasticity Controller switches to the Statistical mode. This allows it to make a final attempt before rejecting the elasticity event and thus indicating the SRO (which will trigger the horizontal elasticity). The SLOTS Statistical mode is designed on the principle of *Recipients* and *Donors* slices, which is designated as the Solidarity Elasticity mechanism. On the one hand, *Recipients* slices facing bottlenecks (i.e., the current utilization rate of a given resource type matches the saturation threshold) which might have an impact on SLAs and, as such, are eligible to receive more resources from the infrastructure provider. These resources, which are designed to scale-up the *Recipients* slices, come from the residual resources of other activated slice(s) (i.e., the difference between the current rates of the amount reserved for a given resource and the corresponding usage rate).

On the other hand, *Donors* are active slices that, for some reason, have more resources allocated to them than their actual usage rate (or need). An efficient elasticity mechanism should always provide over-provisioned resources for slices, although these residual resources cannot be either high or low. In situations where, over-provisioned resources are high, there might be an excessive expenditure. Conversely, in circumstances where over-provisioned resources are low, the elasticity mechanism may be triggered several times. During the period that the slices display the residual resource rates that are available within their respective over-provisioned amounts, SLOTS remains able to calculate the new resource patterns that must be allocated from the residual resources of *Donors* slices to a given *Recipient* slice, without infringing SLA regulations.

Both the *Recipient* and *Donor* slices set a Maximum Resource Threshold ($MRth$, which represents the maximum amount of resources that can be assigned to every active slice) along with a Committed Resource Threshold ($CRth$, the SLA-defined minimum amount of resources that can ensure "overtime" for every active slice). The premise of the Statistical elasticity mode resource-computing scheme is to enable the *Donor* slices to donate resources proportionally to their current utilization ratio. The Donating Rate (DR) comprises only a fraction of residual resources and its purpose is to shrink the size of a *Donor* slice so as to scale-up the *Recipient* slice, where $CRth \leq DR \leq MRth$. SLOTS keeps a list of potential *Donor* slices, which is only composed of the slices that display current resource utilization rates $\geq CRth$. As foreseen, this meant that the larger the *Donor* slice's utilization rate, the lower the amount of DR that is calculated to enable the residual resources to be shrunk in the SLOTS Statistical elasticity mode. The *Recipient* slice, in turn, will be scaled-up by the sum of all the DRs calculated for the selected *Donor* slices.

The mathematical model behind the SLOTS Statistical elasticity mode encompasses a set of equations for sequential processing which are given below. Equation 1 is designed for an idealized operation to calculate the total donating amount

available in the k *Donor* slice (where k is within all d listed *Donor* slices).

$$\forall k \in d, TRDR = \sum_{k=1}^n MRth(k) - CRth(k) \quad (1)$$

Where:

- *TRDR (Total Residual Donating Resources)* – The sum of all the donating residual resources that SLOTS calculates for all the selected *Donor* slices;
- $MRth(k)$ – *Maximum Resource Threshold* of *Donor* slice(k);
- $CRth(k)$ – *Committed Resource Threshold* of *Donor* slice(k);
- n – Number of listed *Donor* slices.

On the basis of Equation 1, the Equation 2 is used to calculate the fraction of residual resources that each listed *Donor* slice k will donate to the *Recipient* slice r .

$$DR = \sum_{k=1}^n \frac{(MRth(k) - CRth(k)) * 100}{TRDR} \quad (2)$$

Where:

- *DR (Donating Rate)* – represents the fraction of residual resources for donating made by each listed *Donor* slice;

Lastly, Equation 3 is used to calculate the new definition of a resource-type for each listed *Donor* slice k . Equation 3 is expressed in the range between the slice in the first position of the *Donor* slice list d , n represents the total number of *Donor* slices list members.

$$\forall k \in d, MRth_{k=1}^n = (MRth(k) - CRth(k)) - \frac{((TRDR - Nr) * DR(k))}{100} \quad (3)$$

Where:

- *Nr (Necessary Resources)*: means the resource required by a given *Recipient* slice.

On the basis of the sequence of procedures illustrated in Fig. 2, from the beginning SLOTS monitors every active cloud-network slice instance. After determining that a given cloud-network slice instance matches its resource saturation threshold, the corresponding elasticity monitor component invokes the DC Elasticity Controller (event 1) to carry out vertical elasticity based on the *Recipient* slice template. In the first place, the DC Slice Controller retrieves the type of resource required (CPU, RAM and/or bandwidth) from the *Recipient* slice template (sent by the Cloud-/Net-Slice Elasticity Monitor), and after accomplishing this, it searches the listed *Donor* slices. On calculating the DR amount for each *Donor* slice (by calculating Equations 1) and 2), the DC Elasticity Controller scales-down each listed *Donor* slice. Afterwards, the DC Elasticity Controller adds up all the DR amounts obtained from the *Donor* slices and scales-up the *Recipient* slice through Equation 3). When it has made the

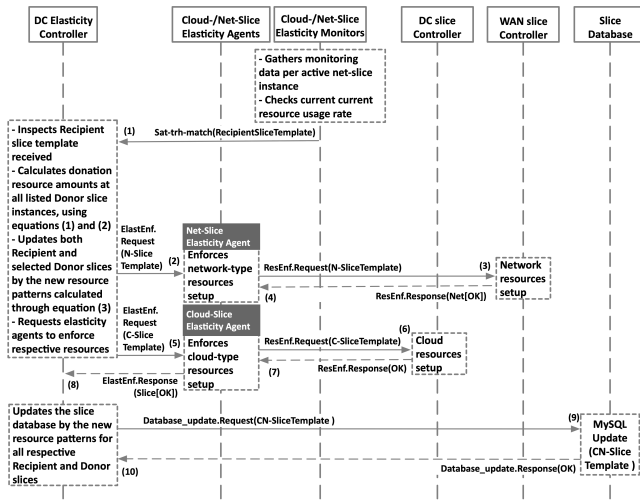


Fig. 2: SLOTS vertical elasticity workflow

new resource definitions of the *Recipient* and *Donor* slices, the DC Elasticity Controller propagates the new templates to the Net-Slice and/or Cloud-Slice Elasticity Agent(s) so that it can set up the new resource definitions accordingly (Stages 2 and 5). Each Elasticity Agent carries out its respective resource enforcement tasks by interfacing with the WAN (Stages 3 and 4) and/or WAN (Stages 6 and 7) Slice Controller(s). On receiving confirmation of the resource enforcement from the Elasticity Agent(s), the DC Slice Controller then updates the local slice database and synchronizes it with the Slice Database of the NECOS platform, thus completing the vertical elasticity workflow for the *Recipient* slice.

IV. SLOTS TESTBED EVALUATION

This section examines the evaluation results collected from our lab-premised testbed prototype with the aid of two vertical elasticity mechanisms: a traditional solution referred to as Regular (simulates the behavior of the Kubernetes algorithm [7]), and our SLOTS scheme. With the goal of assessing their efficiency for handling elasticity events, it was found that both Regular and SLOTS mechanisms adopt the same stochastic-based elasticity control approach to scale-up the current resource definitions of an elasticity-demanding slice, by adding an elasticity factor rate. The elasticity factor keeps 100% of its current resource definition until vertical elasticity matches the resource utilization threshold of 80% of the amount reserved, being reduced by 50% until it achieves success. If the vertical elasticity function cannot be achieved successfully even after consecutive elasticity factor reductions, Regular experiments will reject the elasticity event, whereas SLOTS triggers the Statistical elasticity mode.

The testbed is built on top of an OpenStack-empowered cluster enclosing two Dell PowerEdge R740 servers interconnected by 8 OpenFlow-enabled wire-meshed switches, which allow virtual machines to be connected by SDN paths. Each server features 40 vCPUs along with RAM of 64 GB, thus forming a total computing infrastructure of 80 vCPUs and

128 GB RAM that is available for cloud slicing. In the SDN infrastructure, a bandwidth rate of 5 Gbps is available for network slicing (5 Gbps remains for in-bound OpenFlow control-plane).

The methodology employed for both Regular and SLOTS testbeds, consists of the following different scaling stages with regard to active cloud-network slice instances (i.e., 100, 250, 500, 750 and 1,000) from the beginning of the experiments, since slices must be active before they can become the subject of elasticity operations.

A. Result assessments

The main aim of the evaluation tests is to assess to what extent the SLOTS elasticity mechanism (solidarity approach) is more beneficial than the stochastic-based scheme which the Regular mechanism deploys. Our analysis sets out by examining the number of "reject" and "accept" elasticity events or loads that both SLOTS and Regular experiments provide for the different scales of the tests that are carried out. This measure is important to give an idea of the level of efficiency obtained by the resource-computing approaches that each benchmark solution provisions. Figure 3 displays the average outcomes of the tests that were conducted.

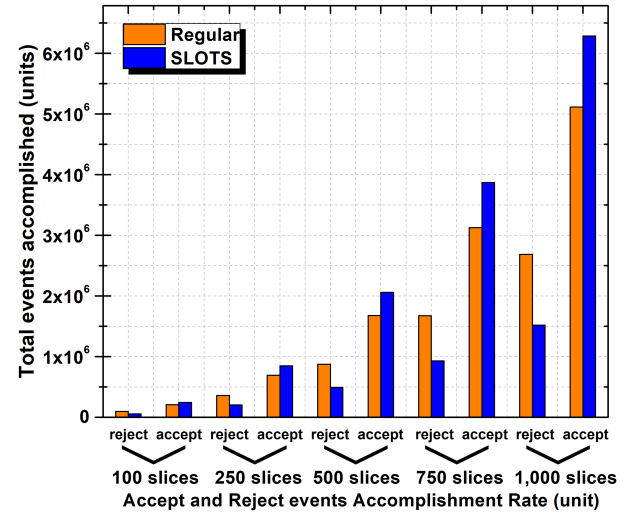


Fig. 3: Figure 3. A comparative analysis of the 'reject' and 'accept' elasticity loads.

The outcomes of Fig. 3 yielded very good results and provides evidence to confirm, that the SLOTS hybrid elasticity control-plane outperforms the Regular solution in obtaining enhanced "accept" loads of vertical elasticity while allowing lower "reject" loads to be provided at the same time. Although the literature claims that over-provisioning resources in cloud-computing auto-scaling are undesirable and unnecessary, the results show that when in Statistical elasticity mode, SLOTS benefits from this. Basically, the presence of over-provisioned resources enables SLOTS to share these residuals between the slices, and thus lead to an improved 'accept' elasticity events load. The numerical results that Figure 3 sketches are on the

basis of Table I, and I complement with resulting numerical analysis.

TABLE I: Results of SLOTS and Regular experiments.

	100	250	500	750	1,000
SLOTS Accept	244,693	847,734	2,059,062	3,871,360	6,283,448
SLOTS Reject	55,307	202,266	490,938	928,640	1,516,552
Regular Accept	204,960	691,465	1,676,289	3,125,397	5,115,137
Regular Reject	95,040	358,535	873,711	1,674,603	2,684,863

TABLE II: Percentual loads of accept and reject events accomplished in the experiment tests

	100	250	500	750	1,000
SLOTS Accept	81.56%	80.74%	80.75%	80.65%	80.56%
SLOTS Reject	18.44%	19.26%	19.25%	19.35%	19.44%
Regular Accept	68.32%	65.85%	65.74%	65.11%	65.58%
Regular Reject	31.68%	34.15%	34.26%	34.89%	34.42%

In the case of the SLOTS and Regular tested solutions, the percentage obtained from the loads of both 'accept' and 'reject' vertical elasticity events (displayed in Table II) reveals a slight variation in their behavior after all the experimental tests have been conducted. The numerical analysis revealed that the average performance of SLOTS (compared with the Regular mechanism) is 18.23% with regard to improving the 'accept' vertical elasticity events load, while allowing a reduction of 43.45% in the average number of reject loads during all the assessments that were carried out.

The averaging cloud- and network-type resource patterns resulting from the vertical elasticity workloads of both SLOTS and Regular mechanisms after accomplishing the experimental tests, along with indications of the initial resource-patterns and the committed resource threshold, are sketched in the Fig. 4. These results can provide insights that expose the capabilities that both SLOTS and Regular mechanisms carry out to compute the new resources patterns along each of the experimental test scales.

The outcomes that Fig. 4 sketches manifest the enhanced ability of the SLOTS approach over the Regular testbed configuration in computing resources to cloud-network slices. This feature is desirable since it means SLOTS is able to avoid including new slice parts in the slice composition, by preventing the time-consuming problem of the migration of the components that belong to the services running inside the slices. Indeed, Fig. 4 outcomes can give the wrong idea that SLOTS provides slight resource allocation performance over the Regular solution. However, the y-axis shows the amount of resource enforcement in the significant scale of thousands and millions, far not a slight difference. When comparing with Fig. 3, the evaluation results suggest the SLOTS approach distinguishes itself from mainstream elasticity solutions by the ability to accomplish a higher density of elasticity events, while allocating more resources amounts to cloud-network slice at the same time, under resource-depleting conditions.

It is worth highlighting that whereas SLOTS improves the vertical elasticity ability of the Regular experiments in the

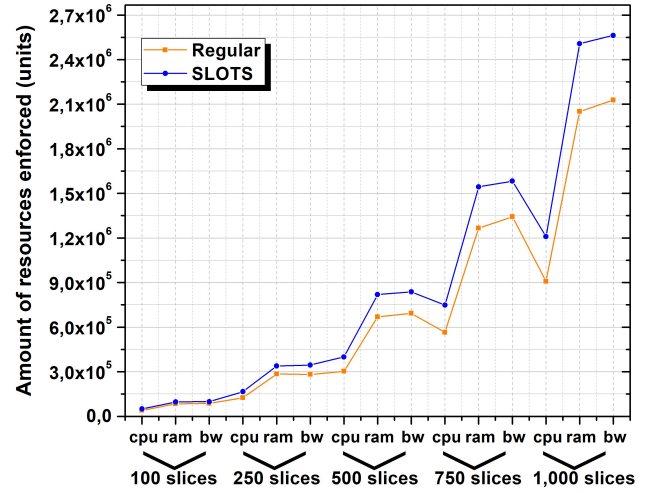


Fig. 4: Averaging resources amount accomplished by SLOTS and Regular in all the experimental tests.

scale of thousands of event units, the scale for the processing time spent is in milliseconds. Moreover, the number of *Donor* slices that SLOTS searches so that it can calculate donating rates for scaling-up *Recipient* slices, increases exponentially with the scale of the different experimental tests, whereas the Regular approach always deals with individual slots.

V. CONCLUDING REMARKS AND FUTURE WORK

This paper introduces the new vertical elasticity hybrid control-plane addition to the SLOTS approach, which provides the ability for the NECOS-Platform to deal with vertical elasticity of cloud-network slices with an awareness of resource-depletion conditions. The outstanding capabilities of leveraging residual resources from selected slices (designated as *Donors*) to donate portions to a given slice (*Recipient*), makes SLOTS unique over related works. A large-scale lab-premised prototyping evaluation is used to achieve more in-depth knowledge of different performance factors under different slice densities. The evaluation tests provide proof of the value of the SLOTS conceptual design and functional architecture. The outcomes suggest that the SLOTS outperforms a traditional stochastic-based elasticity control approach (Regular), by reaching enhanced performance to accomplish both 'accept' and 'reject' elasticity loads (80.52% and 19.14% on average, respectively) over the Regular testbed (averaging 66.12% and 33.88% respectively) in the slice scale of thousands, and that there was an additional processing cost ratio of 54.83% measured in the milliseconds scale.

ACKNOWLEDGMENT

This research was partially supported by the H2020 4th EU-BR Collaborative Call (Grant Agreement no. 777067 – Novel Enablers for Cloud Slicing). This study was financed in part by both the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior-Brasil* (CAPES), Finance Code 001, and the National Council for Scientific and Technological Development (CNPq).

REFERENCES

- [1] F. S. D. Silva, M. O. O. Lemos, A. Medeiros, A. V. Neto, R. Pasquini, D. Moura, C. Rothenberg, L. Mamatas, S. L. Correa, K. V. Cardoso, C. Marcondes, A. ABelem, M. Nascimento, A. Galis, L. Contreras, J. Serrat, and P. Papadimitriou, "Necos project: Towards lightweight slicing of cloud federated infrastructures," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, June 2018, pp. 406–414.
- [2] E. F. Coutinho, F. R. de Carvalho Sousa, P. A. L. Rego, D. G. Gomes, and J. N. de Souza, "Elasticity in cloud computing: a survey," *annals of telecommunications-Annales des télécommunications*, vol. 70, no. 7-8, pp. 289–309, 2015.
- [3] A. Medeiros, A. Neto, S. Sampaio, R. Pasquini, and J. Baliosian, "End-to-end elasticity control of cloud-network slices," *Internet Technology Letters*, vol. 2, no. 4, p. e106, 2019. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/itl2.106>
- [4] S. Farokhi, P. Jamshidi, I. Brandic, and E. Elmroth, "Self-adaptation challenges for cloud-based applications: A control theoretic perspective," in *10th international workshop on feedback computing*, vol. 2015, 2015.
- [5] "Amazon target tracking scaling," Amazon AWS, 2018. [Online]. Available: <http://docs.aws.amazon.com/autoscaling/latest/userguide/as-scaling-target-tracking.html>
- [6] "Google multiple policies scaling," Google, 2018. [Online]. Available: <https://cloud.google.com/compute/docs/autoscaler/multiple-policies>
- [7] "Kubernetes horizontal pod auto-scaling," Google, 2019. [Online]. Available: <https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/>
- [8] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah, and P. Merle, "Autonomic vertical elasticity of docker containers with elasticdocker," in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*. IEEE, 2017, pp. 472–479.
- [9] C. Kan, "Docloud: An elastic cloud platform for web applications based on docker," in *2016 18th International Conference on Advanced Communication Technology (ICACT)*. IEEE, 2016, pp. 478–483.
- [10] R. da Rosa Righi, V. F. Rodrigues, G. Rostirolla, C. A. da Costa, E. Roloff, and P. O. A. Navaux, "A lightweight plug-and-play elasticity service for self-organizing resource provisioning on parallel applications," *Future Generation Computer Systems*, vol. 78, pp. 176–190, 2018.
- [11] S. Taherizadeh and V. Stankovski, "Dynamic multi-level auto-scaling rules for containerized applications," *The Computer Journal*, vol. 62, no. 2, pp. 174–197, 2018.
- [12] "D3.1: Necos system architecture and platform specification. v1," Stuart Clayman, 2018. [Online]. Available: <http://www.h2020-necos.eu/documents/deliverables/>