

Exploiting Case Based Reasoning to Automate Management of Network Slices

Pedro Martinez-Julia*, Jose Ordonez-Lucena[†], Ved P. Kafle*, Hitoshi Asaeda*, and Diego Lopez[†]

*National Institute of Information and Communications Technology (NICT), Tokyo, Japan

[†]Telefonica I+D, Madrid, Spain

Email: {pedro,kafle,asaeda}@nict.go.jp* {joseantonio.ordonezlucena,diego.r.lopez}@telefonica.com[†]

Abstract—Network softwarization is a transformation trend for networks to converge in programmable service platforms able to host different requirements from multiple tenants. The concept of network slicing plays a key role in this trend. It offers isolated logical networks to different tenants over the same underlying infrastructure. Network slices hosting mission-critical services require fast reactions to changes in environment, adapting their capacity as needed to avoid service disruption and performance degradation. This represents a key challenge for the management plane, traditionally under the direct operation of human administrators, unable to provide a nimble enough response rate. To support tenants in driving lifecycle management operations over their network slices in a timely manner, data-driven, closed-loop mechanisms must be designed. In this paper we introduce a solution that takes advantage of events occurring outside the boundaries of a tenant-managed network slice to guide those adaptations. We finally demonstrate the feasibility of the proposal and describe an experimentation scenario to exploit such results.

I. INTRODUCTION

Network slicing is a cutting-edge technology whereby a single physical network infrastructure is partitioned into a number of segregated logical network instances, each optimized to satisfy the particular service requirements of a given tenant (e.g. industry vertical) or application [1], [2]. The optimization of these Network Slice Instances (NSIs) brings management and control challenges, since resources from the underlying network infrastructure must be abstracted and delivered in such a manner that (i) each NSI is provided with enough resources to satisfy its performance needs, and (ii) all NSIs can run on top of the same infrastructure with isolation guarantees [3].

The time-varying traffic loads that need to be accommodated by NSIs translate into constant modifications in the capacity of these NSIs. This makes traditional resource provisioning operations to be much more dynamic, leading to an increase in Operational Expenditure (OPEX) that could compromise the cost-effectiveness of network slicing technology. Although the use of Software Defined Networking (SDN) and Network Functions Virtualization (NFV) [4] provides some means of automation to cope with such a dynamicity, today's SDN/NFV solutions still require too much supervision of human administrators, with time scales that are significantly far from what is expected in some slicing scenarios, especially those involving mission-critical NSIs. This type of NSIs have dedicated resources and are usually delivered in the form of Network

Slice as a Service (NSaaS) to tenants that wants to keep a lot of control over them (e.g. government, military, public safety).

Mission-critical NSIs require closed-loop resource optimization algorithms to cope with the fast reactions they demand, but few algorithms are well suited for them [5]. The Autonomic Resource Control Architecture (ARCA) [6] aims at dynamically modifying the capacity of any mission-critical NSI in response to changing situations in an agile and automated manner. ARCA does reason from both performance measurements collected from the NSI and events notified by external detectors [7]. External event notifications might be earthquake or terrorism alerts.

In this paper we present ARCA as the management system of a given tenant in NSaaS scenarios, achieving the elastic adaptation of mission-critical NSI in a timely and automated manner. We also discuss how ARCA decisions are improved by incorporating the intelligent reasoning following Case Based Reasoning (CBR). Such decisions rely on CPU load, network bandwidth, and external events, as they are the major determinants for the amount of resources required by the network. This allows the formulation of cases from the observations received by ARCA, in the form of in-NSI performance metrics and external event notifications. We formalize external events as *causes* and the ulterior performance measurements derived from them as *effects*.

The remainder of this paper is organized as follows. First, we contextualize the present work by introducing NSaaS implications and the ARCA framework in Section II. Then, in Section III we describe the proposed decision algorithm and in Section IV we evaluate it to justify our claims. Finally, in Section V we conclude the paper.

II. BACKGROUND

In this section we contextualize ARCA, introduce a system architecture for delivering NSI on demand by the management and orchestration of network slices (NS-MANO), and discuss the related work.

A. Autonomic Resource Control Architecture (ARCA)

As stated in the introduction, ARCA has been designed with the ability to manage NSIs [6], [7], especially mission-critical NSIs. It allows optimizing the capacity of any NSI, and modifying the amount of resources assigned to its constituent VNF instances in response or anticipation to different stimuli.

ARCA correlated performance measurements and external events, such as an earthquake or the detection of a heavy rainfall [8], to the damages it can make to the NSI.

ARCA workflow follows the MAPE-K closed control loop defined by the Autonomic Computing (AC) paradigm [9], as well as the MANA and ETSI-AFI reference models [10]. Thus, it includes the four main elements and activities defined by AC: the *collector*, the *analyzer*, the *decider*, and the *enforcer*. The *collector* is responsible of gathering and formatting the heterogeneous observations that will be used in the control cycle. Then, the *analyzer* finds the current load of the resources allocated to the managed system (i.e. the NSI), the rate of packet drops, and the occurrence of an event that can affect its normal operation. The outputs from the analyzer are therefore the *summarized* load, drops, and event. The *decider* determines the necessary actions to adjust the resources to the present or near future load of the managed system and, finally, the *enforcer* is in charge of requesting the underlying and overlying infrastructure to make the necessary changes to enforce decided actions. In this article, where NSaaS is assumed, ARCA takes the role of tenant management system.

B. System Architecture for NS-MANO

NS-MANO consists of those activities that guarantee the expected behavior and isolation of deployed NSIs, from their commissioning to their decommissioning [11]. These activities can be executed at:

- Different network domains, including Radio Access Network (RAN), Core Network (CN) and Transport Network (TN) domains. These domains require separate management, since they have different requirements in terms of functionality, capacity and scalability.
- Different layers, including application and resource layers. Application layer deals with the semantics and parameter settings of the NFs taking part in the NSI, while resource layer focuses on the deployment of these NFs.

As depicted in Figure 1, the system architecture for NS-MANO has NSIs running on top of a shared network infrastructure driven by upper-layer management functions. These management functions can be arranged into different domains, which vary depending on the considered scenario. In this work we assume NSIs are deployed in a 3GPP mobile network with up to five management domains: E2E, RAN, CN, TN and Virtualization Management Domains, including the 3GPP Network Slice Management Function (NSMF) [11], and providing both vertical and horizontal deployments.

C. NSaaS

NSaaS is a service delivery model whereby a network slice provider delivers an NSI to a given tenant (e.g. industry vertical) upon request, allowing him to operate the NSI according to his specific needs. From a management perspective, NSaaS means that the provider management system deploys the NSI and makes it available to the tenant management system, allowing this external management system to trigger lifecycle management operations over that NSI in a timely manner.

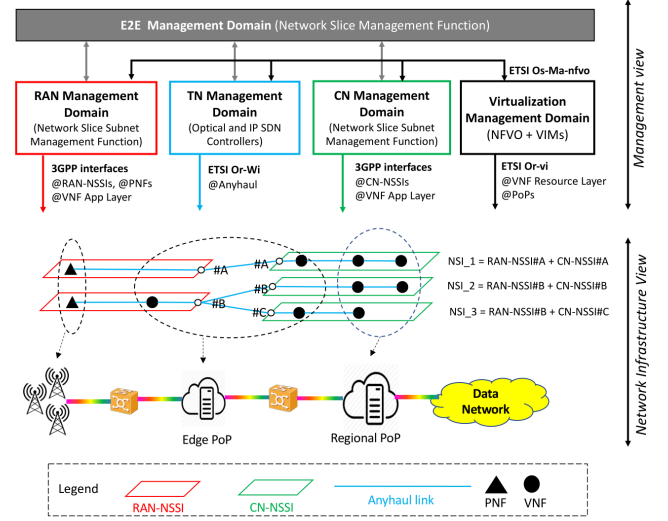


Figure 1. A system architecture for NS-MANO.

Thus, the tenant management system must be securely exposed by the management capabilities from the provider system in order to properly enable NSaaS.

Service exposure in network slicing scenarios raises important implications, particularly considering that different tenants could want to keep different levels of management and control over their provided NSIs. This fact makes necessary to define different levels of exposure. In this regard, some approaches have been proposed so far to address this problem [12].

D. Related Work

Current solutions for automated control and management rely on the estimation of requirements to determine the amount of resources or number of network element instances that would be assigned to the controlled virtual network system. They estimate the requirements by analyzing some performance metrics obtained from the resources already assigned to the controlled system (e.g. their CPU load) [5]. Then, they match the estimated requirements to the needed resource amount and adjust the controlled system accordingly. The most usual objective of these operations is to reduce the amount of resources to be assigned to the controlled system as a function of some performance metric, such as the CPU load itself.

The analysis function can be as simple as a mapping from certain level of requirement and certain amount of resources [13], or a more complex calculation such as a mathematical induction [14] or even the result from some machine learning operation [15]. The objective in all of them is to minimize the amount of resources assigned to the controlled system but, even when using some ML method, they only consider the performance metrics to determine the amount of resources to be enforced to the system [16].

As discussed throughout this paper, relying management operations just on performance metrics ends in inefficient allocations, which are especially delicate in network slicing environments. Therefore, decision mechanisms need to be

empowered with better AI mechanisms and rely on more parameters, such as external events.

III. APPLICATION OF CASE BASED REASONING TO NETWORK SLICING AS A SERVICE

In this section we describe how to enhance ARCA operation by applying Case Based Reasoning (CBR), adapting the jCOL-IBRI2 [17] framework. Current ARCA decision operation already exploits AI methods to find the resources required by the NSI under its management. By following the CBR methodology, ARCA decision operations will be strengthened by exploiting the powerful reasoning abilities provided by CBR.

The CBR methodology is quite old but it is overlooked by current network management solutions and latest research work, especially considering the potentially new applications to virtual networks and network slicing. However, CBR can benefit to this kind of management, especially in the context of NSaaS, as we discuss throughout this paper.

While most of current management solutions work only on internal alarms bound to the managed system, we propose to build the CBR from external events (causes/conditions), performance measurements (effects/telemetry), and countermeasures (actions). In this sense, telemetry is processed by a threshold-based rules mechanism to provide allocations for new situations and assess the decisions taken by the CBR. On the one hand, telemetry is obtained from the controller of the underlying platform (e.g. OpenStack). On the other hand, external events are provided by external detectors (e.g. seismometers and weather stations) and are then uniformed by using a common ontology provided by ARCA. The main caveat is how to formulate cases so that their distance can be measurable, which is the major technical contribution of the paper, as shown in the algorithm discussed below.

Particularly, the integration of the CBR methodology enables ARCA to exploit the knowledge acquired from resolving past undesired and/or failure situations. While resolving such situations, ARCA records the problems found in the NSI and the actions (countermeasures) required to resolve them. Those actions are then requested to the provider management system for their enforcement. Subsequently, if similar situations are detected, ARCA is able to avoid the effects by anticipating actions to correct the undesired situations.

In summary, the key objectives for the incorporation of CBR to ARCA operation are as follows:

- Detect potential, trivial, and/or frequent failures from assessing the working environment and context of the managed network slice.
- Anticipate, as much as possible, as many actions as possible to avoid disruption of the managed network slice.
- Learn from previous hits and failures, reinforcing "believes" of most important and correcting wrong assumptions, applying fuzzy logic and fuzzy reasoning to recorded cases.

To achieve such objectives and thus maximize the benefits of CBR, we first define how cases will be formulated from the external events and performance measurements. As introduced

above, while performance measurements are required by all management solutions, especially those that implement some automation, the external event notifications are required mainly for reasoning and deriving the resources required by the NSI. Thus, both sources of information form the basement of the autonomic management solution described in this paper.

During the formulation of cases, the external events that have occurred at some moment of time are linked with performance measurements obtained in later moments. For this property to be useful for the CBR, the cases must be mapped to an euclidean space and then ensure their distance is subject to the following properties:

- Cases formed from similar external events are close to each other, so their euclidean distance is minimized.
- Cases that include similar performance measurements and similar external events are close to each other, so their euclidean distance is minimized.
- Cases with similar performance measurements but different external events are far away to each other, so their euclidean distance is maximized.

In order to ensure those properties, in this paper we present an algorithm (see Algorithm 1) to formulate the cases by linking external events and ulterior performance measurements before cases are actually formulated. Before going into detail on the algorithm, in the next subsection we formalize the problem, objectives, and constraints to the algorithm.

A. Problem Statement

As discussed throughout this paper, to apply the benefits of CBR to the context of NSaaS, ARCA must be able to *sense* the environment of the NSI and then formulate the required cases to represent the situations ARCA is able to detect. Case formulation becomes a key operation of the whole CBR process, so this paper is focused on the problem of defining a method for case formulation that accomplishes the requirements mentioned above. Thus, the proper extraction of knowledge from previous cases (causes, effects, situations, actions) depends on case formulation and the application of obtained *wisdom* to resolve potential cases found in ulterior moments, even before they actually affect the NSI (anticipation). Therefore, the problem relies on:

- T : Set of all time instants.
- E_q : Set of event notifications for $t \leq q$; $t, q \in T$.
- M_q : Set of perf. measurements for $t \leq q$; $t, q \in T$.

Although the function works dynamically for each time tick ($t \in T$), we specify the operation (case formulation) by considering all time ticks together (in T). Moreover, we need to properly identify the set of external events (E_q) and the set of performance measurements (M_q) obtained before certain moment of time.

From those simple definitions we can state the problem as obtaining the association of performance measurements on time $t = i + \delta$ to events from time $t = i$. This will form the premise of a case, in which the external events are the *cause* and the measurements are the *effect*. For resolving this problem

we provide E_q and M_q as input and we get all associations of events and measurements (target output) as follows:

$$\{e_i, m_{i+\delta}\} \forall e_i \in E_q \quad (1)$$

$$: m_{i+\delta} \subset M_q; i, \delta \in T; i + \delta \leq q$$

While building the set of associations, the method is able to match a huge amount of measurements and events. From all of them and their potential associations, we are interested in those that are more representative of interesting situations of the NSI, which will be the best candidates for being chosen as representative cases for the CBR. Therefore, the case formulation method must enforce the following objectives:

$$\min |e_k, m_{k+\delta} - e_l, m_{l+\delta}| \forall k, l \in T \quad (2)$$

$$: |e_k - e_l| < \epsilon$$

$$\min |e_k, m_{k+\delta} - e_l, m_{l+\delta}| \forall k, l \in T \quad (3)$$

$$: |e_k - e_l| < \epsilon, |m_{k+\delta} - m_{l+\delta}| < \epsilon$$

$$\max |e_k, m_{k+\delta} - e_l, m_{l+\delta}| \forall k, l \in T \quad (4)$$

$$: |e_k - e_l| \gg 0, |m_{k+\delta} - m_{l+\delta}| < \epsilon$$

The first objective is specified in Equation 2. It aims to minimize the distance of associations between events and measurements related to similar (close) events. This minimization is separated from the next one to clearly ensure that similar events will form similar cases, as it is desirable for the proper application of the CBR. Moreover, in Equation 3 we enforce the minimization of the distance among cases that have similar events and similar measurements. This is a trivial and straightforward objective but it must be enforced as disparate measurements can conduce to wrong case formulations.

Although the first two objectives are overlapped, they must be considered separately as there can be cases that match with only one of them not the other. Finally, in Equation 4 we set the objective of maximizing the distance among cases that have close measurements but distant events. This objective is important to facilitate the ulterior extraction of knowledge because it ensures that similar cases are somewhat grouped in the vector space and different cases are somewhat distant. We do not specify any objective for other relations since these objectives are enough for a rich extraction of knowledge.

From all possible combinations of events and measurements we are interested in those that have actual meaning for the managed NSI. Thus the formulation method must choose those associations of events and measurements that accomplish the following constraints:

$$\forall e_i \in E \exists m_{i+\delta} \subset M; i, \delta \in T; \delta \geq 0 \quad (5)$$

$$: f(e_i) = m_{i+\delta}$$

$$\forall m_j \subset M \exists e_{j-\delta} \in E; j, \delta \in T; \delta \geq 0 \quad (6)$$

$$: f'(m_j) = e_{j-\delta}$$

Algorithm 1 Case formulation procedure.

```

1: procedure FORMULATEALLCASES( $e$  : operation_
   environment)
2:    $eset, mset \leftarrow \text{GETDATA}$ 
3:    $kv \leftarrow []$ 
4:   for all  $t \in T$  do
5:      $esel \leftarrow [e : e \in eset, e.t = t]$ 
6:      $mset \leftarrow [m : m \in mset, m.t = t + \delta]$ 
7:      $kvn \leftarrow \{e = esel, m = mset\}$ 
8:     for all  $kvi \in kv$  do
9:       if  $\text{VSDST}(kvn.e, kvi.e) < \epsilon$  then
10:         $kvi.m \leftarrow kvi.m + knv.m$ 
11:         $kvi.e \leftarrow kvi.e + knv.e$ 
12:       else if  $\text{VSDST}(kvn.m, kvi.m) < \epsilon$  then
13:         $kvn.e \leftarrow knv.e - kvi.e$ 
14:       end if
15:     end for
16:      $kv \leftarrow kv \cup [kvn]$ 
17:   end for
18: end procedure

```

We use $y = f(x)$ to represent that y is linked to x (but not the contrary) and $x = f'(y)$ to represent that x is linked to y (inverse relation). The constraints ensure that each set of events from time i can be linked to an ulterior set of measurements in time $i + \delta$ and that each set of measurements from time j can be back-traced to (and somehow derived from) a set of events occurred on time $j - \delta$. With these constraints and the equations introduced above the problem of case formulation is fully specified.

B. Algorithm

In this section we describe the procedure that allows resolving the problem described in above. This procedure takes the input data, namely the external events and performance measurements, and builds all cases, namely the associations between events (causes) and measurements (effects).

The procedure is formalized in Algorithm 1. It first gets the input data, which are the sets $eset$ ($= E_q$) and $mset$ ($= M_q$), and defines the output vector kv as an empty set. Then, the procedure goes through every time moment $t \in T$ and selects the events found at it and the measurements found in ulterior time $t + \delta$, as we specified in the objective equations above. The resulting vectors from such selections are used to build a new association kvn .

With the new vector kvn , the procedure loops throughout the previous vectors existing in kv and checks the vector-space distance (e.g. euclidean) of their events and measurements with those from the new one. If the events are close, the vector of measurements of the entry found in the set of previous vectors is completed with the measurements from the new vector ($kvn.m$). With this we ensure that the resulting vector meets the objective of Equation 2, since both events and measurements will be similar, meaning they are close in the euclidean space. At the same time, the vector of events of

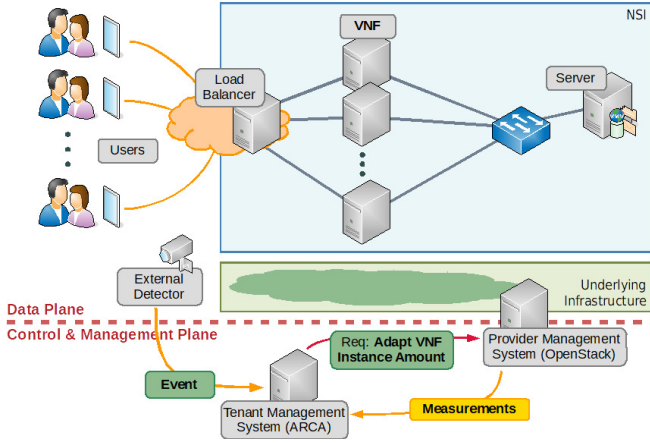


Figure 2. Evaluation topology: NSI and control/management elements.

the entry found in kv is completed with the events from the new association ($kvn.e$). With this we ensure that objective of Equation 3 is met.

If the set of events of the new item is not close to the set of the existing item but the set of measurements is, the events from the set of events of the existing item ($kvi.e$) are removed from the set of events of the new item ($kvn.e$). Finally, after all existing items in kv are processed, the remaining item (kvn) is added to kv . This represents how a new entry is added to the knowledge vector.

Although the formalization of the procedure assumes T has all moments of time, since it is built following the *dynamic programming* principle, it is also valid for incomplete T sets. The incomplete T sets are those including all moments of time from t_0 and the moment of time when the procedure is executed. Note that the output is idempotent and, furthermore, it can also continue working for new moments of time t newly incorporated in T as well as the events and measurements associated to them.

IV. EVALUATION AND RESULTS

In this section we describe an environment we use to evaluate the applicability of ARCA to NSaaS as well as some preliminary results of the execution of ARCA with the CBR.

A. Validation of the CBR

For the validation of the CBR we first build a simulation scenario based on the straight implementation of the proposed algorithm onto an ARCA instance. For the simulation, which is built with SimPy [18], we set some boundaries for the amount of resources assigned to an NSI (minimum and maximum) and initialize the case database to the vector $((0, 0, 0), 1)$, which means that for an event 0, having load 0 and drops 0, the amount of resources is set to 1. The cases used in the evaluation are formed by humans (network administrators) from a data-set generated from real network measurements.

The simulation is executed using the topology depicted in Figure 2 for a dataset we generate following the daily patterns from a real environment. Apart from ARCA/CBR, we also

run the *THR* algorithm, which is based on the widespread threshold-based rules control algorithm. We set the lower and upper thresholds to 25% and 75%, respectively. We then retrieve the necessary measurements to evaluate and compare the performance of both solutions.

First we measure the amount of servers that are allocated to the NSI over the optimum, which is directly derived from the amount of users served by the NSI and the portion of server that is required by each user, as well the amount of servers that is currently assigned to the NSI. Second, we measure the amount of requests that each solution sends to the infrastructure for each moment of time. Third, we measure the packets discarded (dropped) by the servers for each moment of time.

With all those measurements we not just demonstrate that the behavior of our proposal, *CBR*, is better than the behavior of *THR*, as we already did in previous work, but also demonstrates the relevance of matching the event notifications sent by external detectors with previous experience (stored cases) to improve the responsiveness of the system while optimizing the number of servers assigned to it. We summarize the measured data in Figure 3. It shows how THR clearly behaves behind CBR in terms of over-allocation, an average of 3 servers allocated by CBR against 6 servers allocated by THR for more than 90% of the time, even having a difference of 1 request in favor of CBR for more than 92% of the time. Finally we see that CBR keeps the drop rate below 100 for almost all the time, while THR doubles that quantity.

V. CONCLUSIONS AND FUTURE WORK

In this paper we have presented an algorithm that allows a tenant to optimize the capacity of an NSI in time-varying scenarios. This algorithm determines what scaling in/out VNF operations are needed within the NSI, in response to the performance measurements collected from the underlying network as well as events reported by external detectors. The execution of this algorithm in the tenant management system is one of the great benefits that NSaaS model brings, since it discharges the provider management system of the responsibility of taking scaling related decisions of each NSI, leaving them up to the corresponding tenant. This approach not only reduces the OPEX, but also allows provider management system to focus on other issues (i.e., cross-NSIs decisions).

We have incorporated the above algorithm to ARCA [6], [7], and evaluated it on an environment where the tenant users accessing the NSI exhibit different behaviors, as reflected in the different traffic patterns provided by the datasets used for the evaluation. We demonstrate our claims by showing the reduction of the resulting allocation in a network slicing scenario whose load follows a typical daily pattern, what has higher impact considering the self-assessment qualities of the presented algorithm. Moreover, we quantified the benefit of using the information provided by external event detectors. With all of it, we obtained 50 % improvement in the total amount of packets dropped during the execution of the evaluation scenario.

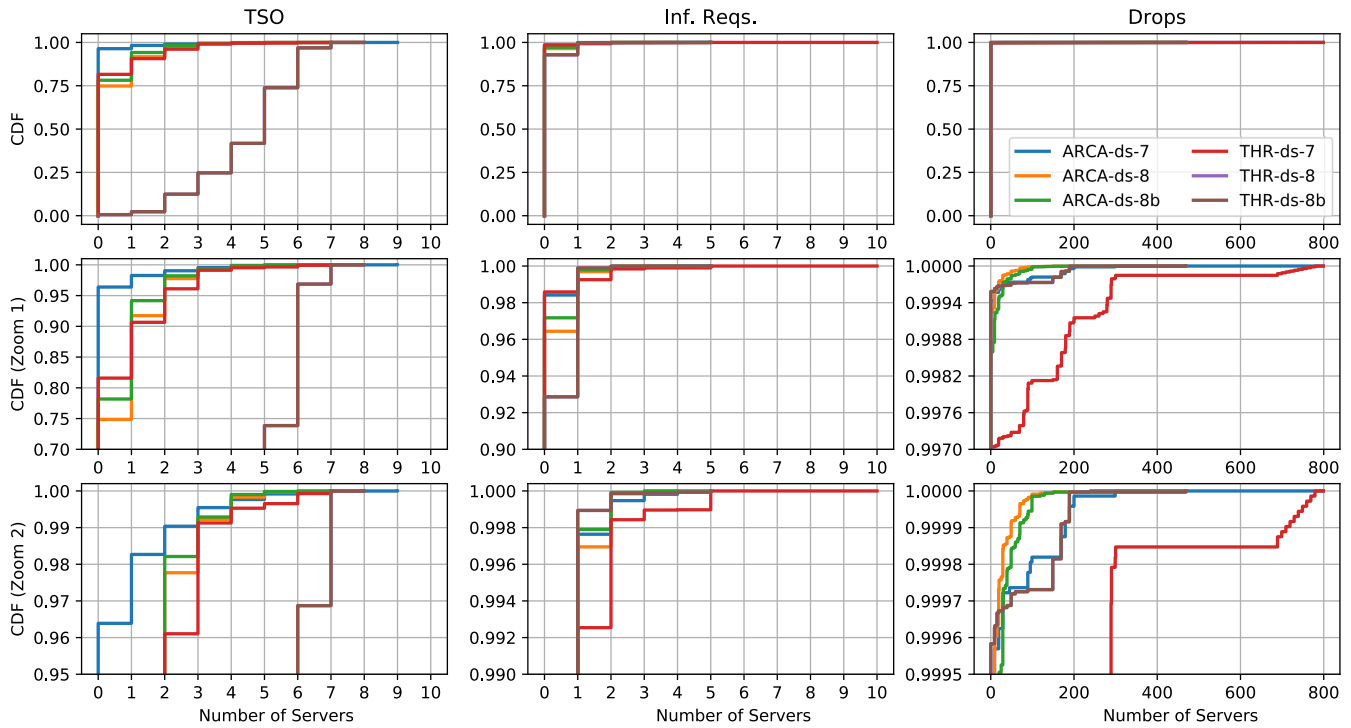


Figure 3. Results obtained from the evaluation.

Finally, in future work we will keep improving the decision mechanism in order to further reduce the computation time of ARCA, and we will evaluate the performance of the experimentation scenario described at the end of this paper, where ARCA interacts with OSM.

REFERENCES

- [1] A. Galis, S. Clayman, L. Mamatas, J. R. Loyola, A. Manzalini, S. Kulkinski, J. Serrat, and T. Zahariadis, "Softwarization of future networks and services-programmable enabled networks as next generation software defined networks," in *Proceedings of the IEEE SDN for Future Networks and Services (SDN4FNS)*. Washington, DC, USA: IEEE, 2013, pp. 1–7.
- [2] P. Martinez-Julia, A. F. Skarmeta, and A. Galis, "Towards a secure network virtualization architecture for the future internet," in *The Future Internet*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, vol. 7858, pp. 141–152.
- [3] J. Ordonez, P. Ameigeiras, D. Lopez, J. Ramos, J. Lorca, and J. Folgueira, "Network Slicing for 5G with SDN/NFV: Concepts, Architectures and Challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 80–87, 2017.
- [4] D. R. Lopez, "Network functions virtualization: Beyond carrier-grade clouds," in *Proceedings of the 2014 Optical Fiber Communications Conference and Exhibition (OFC)*. Washington, DC, USA: IEEE Computer Society, 2014, pp. 1–18.
- [5] T. Lorigo-Boltran, J. Miguel-Alonso, and J. A. Lozano, "A review of auto-scaling techniques for elastic applications in cloud environments," *Journal of Grid Computing*, vol. 12, no. 4, pp. 559–592, 2014.
- [6] P. Martinez-Julia, V. P. Kaffle, and H. Harai, "Exploiting external events for resource adaptation in virtual computer and network systems," *IEEE Transactions on Network and Service Management*, vol. 15, no. 2, pp. 555–566, 2018.
- [7] P. Martinez-Julia, V. P. Kaffle, and H. HAsaeda, "Using the total cost of ownership to decide resource adjustment in virtual networks," in *Proceedings of the 22th ICIN Conference (Innovations in Clouds, Internet and Networks, ICIN 2019)*. Washington, DC, USA: IEEE, 2019, pp. 1–7.
- [8] H. Saito, H. Honda, and R. Kawahara, "Disaster avoidance control against heavy rainfall," in *Proceedings of the IEEE INFOCOM 2017*. Washington, DC, USA: IEEE, 2017, pp. 1–8.
- [9] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *IEEE Computer*, vol. 36, no. 1, pp. 41–50, 2003.
- [10] A. Galis *et al.*, "Position Paper on Management and Service-aware Networking Architectures (MANA) for Future Internet," http://www.future-internet.eu/fileadmin/documents/prague_documents/MANA_PositionPaper-Final.pdf, 2010.
- [11] 3GPP TS 28.530, "Management and orchestration; Concepts, use cases and requirements," v16.2.0, 2019.
- [12] Luis M. Contreras and D. Lopez, "A Network Service Provider Perspective on Network Slicing," *IEEE Softwarization*, 2018.
- [13] L. Mamatas, S. Clayman, and A. Galis, "Information exchange management as a service for network function virtualization environments," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 564–577, 2016.
- [14] H. Moens and F. De Turck, "Vnf-p: A model for efficient placement of virtualized network functions," in *Proceedings of the 10th International Conference on Network and Service Management (CNSM) and Workshop*. Washington, DC, USA: IEEE, 2014, pp. 418–423.
- [15] J. Barron, M. Crotty, E. Elahi, R. Riggio, D. R. Lopez, and M. P. de Leon, "Towards self-adaptive network management for a recursive network architecture," in *Proceedings of the 2016 IEEE/IFIP Network Operations and Management Symposium*. Washington, DC, USA: IEEE, 2016, pp. 1143–1148.
- [16] J. Gil Herrera and J. F. Botero, "Resource allocation in nfv: A comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 518–532, 2016.
- [17] J. A. Recio-García, P. A. González-Calero, and B. Díaz-Agudo, "jcolibri2: A framework for building case-based reasoning systems," *Science of Computer Programming*, vol. 79, no. 1, pp. 126–145, 2014.
- [18] N. Matloff, "Introduction to discrete-event simulation and the simpy language," Dept. of Computer Science, University of California at Davis, Tech. Rep., 2008.