Using Real-World Event Notifications to Reduce Operational Cost in Virtual Networks

Pedro Martinez-Julia, Ved P. Kafle, Hitoshi Asaeda, and Hiroaki Harai
Network Science and Convergence Device Technology Laboratory
National Institute of Information and Communications Technology (NICT)
4-2-1, Nukui-Kitamachi, Koganei, Tokyo 184-8795, Japan
Email: {pedro,kafle,asaeda,harai}@nict.go.jp

***Abstract*—In this paper we propose a comprehensive technique to reduce the operational expenses (OPEX) of virtual network systems, which comprises the cost of using resources, the cost of requesting the underlying infrastructure to make changes to the allocated resources, and the cost of dropping network traffic and hence disrupting the service. We demonstrate that it is feasible to reduce OPEX by optimizing resource assignation without relying on thresholds, neither for determining the amount of resources to be assigned nor to assess the correct operation of the control system. Instead, our proposal relies solely on the context definition (day of week, time of day, current resource amount, etc.) and the events reported by external detectors. Finally, we quantify the benefit of considering such events when taking management decisions and how they are useful to step further towards the optimum assignation of Virtual Network Function (VNF) instances to accomplish the demands of the overall system.**

I. Introduction

Resource optimization in virtual computer and network systems is a well known area with always open research space and unresolved questions, as well as management and control challenges [1]. Lately, the proliferation of Network Function Virtualization (NFV) [2], which abstracts functions from specific hardware or locations as Virtual Network Functions (VNFs), has reinforced the need to keep improving the control and management operations of virtual network systems, not just to reduce the operational expenses (OPEX) in terms of resources consumed by the target systems but also to endeavor different situations without the constant supervision of human administrators.

To respond to such challenges, in previous work we designed the Autonomic Resource Control Architecture (ARCA) [3], which adjusts the amount of resources (mainly servers implementing VNFs) assigned to virtual network systems to changing situations, both elastically and automatically. Other management solutions are able to optimize resources [4] but their latency can lead to service disruption, which must be avoided for some systems, such as real-time stock market systems and emergency support systems during emergency situations.

While other solutions rely solely on monitoring information obtained from the controlled system, ARCA additionally exploits the information provided by external event detectors [5], while automating management tasks to provide a substantial reduction in the time required to estimate (and anticipate) the amount of resources that a system requires as its workload changes. Moreover, the final aim of current solutions is to optimize the amount of resources assigned to the system. However, there are other *sources of cost* that must be considered when dimensioning virtual network systems. First, the underlying infrastructure provider can charge for every operation issued to adapt the system. Such cost can be small but, if the management system makes frequent changes, it can be higher than the proper cost of utilizing the resources. Second, we also consider the penalty of the target service for dropping user requests (or packets). In general, current management solutions try to avoid it by any means, incurring in higher cost than strictly required.

In this paper we propose to enhance ARCA by incorporating a new decision algorithm that considers the costs detailed above and demonstrate that it is feasible to approach optimum resource assignation without relying on thresholds and, therefore, resource over- and under-assignation. Second, we show how the new decision algorithm can be used to reduce the total OPEX of a virtual network system based on NFV. Third, and not less important, we demonstrate that events reported by external detectors are useful to step further towards the optimum assignation of VNF instances, improving a threshold-based rules control algorithm (THR) by reducing over-allocations to less than the 20 % of the amount used by THR, the amount of infrastructure requests to less than 60 % of the amount used by THR, and the amount of packet drops to less than 70 % of the drops experienced by the target system when using THR.

The remainder of this paper is organized as follows. First, we contextualize the present work by introducing ARCA 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 and discuss our future work.

II. Background

It is well known that virtualizing and generally softwarizing a computer and network infrastructure provides enormous flexibility to the system and its different instantiations [6]. This led to whole virtualization architectures [7], which incorporate

978-3-903176-15-7

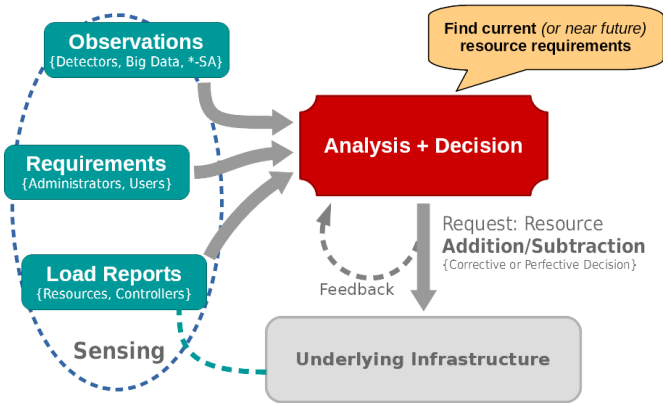

Figure 1. Abstract overview of ARCA workflow.

both SDN and NFV mechanisms from their inception. Such architectures therefore specify the components and stakeholders found in virtualization ecosystems, remarking several emergent use cases. From those use cases we chose the resource optimization [1] and deployment of on-premise vCPE (virtual customer premises equipment) [2] as the main targets of our work. The former is well known and recurring among most virtualization and softwarization technologies. The latter is the direct result of the benefits that such technologies bring to distributed network systems of small to medium size organizations that rely on their own *Cloud* services without outsourcing them to external providers.

To complement and improve the aforementioned architectures, we designed the Autonomic Resource Control Architecture (ARCA) [3]. It targets the optimization of virtual computer and network systems, such as vCPE deployments, especially optimizing the amount of resources (VNF instances) assigned to them. It operates by estimating the current and near future computing and network resource demands of the controlled system by analyzing the load measurements taken directly from the controlled system together with other information reported by external event detectors of different types that notify about the state of the environment of the system. For instance, ARCA is able to consider the measurements taken by a seismometer during an earthquake or correlate the detection of a heavy rainfall to the damages it can make to the controlled system [8].

In Figure 1 we show the components and overall workflow of ARCA. It 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], [11]. 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 system, 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 controlled system and, finally, the *enforcer* is in charge of requesting the underlying and overlying infrastructure to make the necessary changes to enforce decided actions.

The decision algorithm used by ARCA before the present work is based on a learning and anticipation method derived form Support Vector Regression (SVR), the regression application of a Support Vector Machine (SVM) [12], which is self-assessed and self-corrected by incorporating a simultaneous THR mechanism [4]. Although combining THR with SVR gives quite effective results and retains a high level of simplicity, it is difficult to get optimum allocations, especially considering different *sources of cost*. Moreover, the THR sub-algorithm somehow limits the decisions taken by the AI-based sub-algorithm because, as mentioned above, using thresholds imply that some amount of resources will be unused, which imposes a hard limit to the extent the overall control method can reach the optimum allocation and thus optimizing OPEX. However, its effectiveness and accuracy is clearly demonstrated [5], so *freeing* ARCA from the assessment of such classical approach could impact its performance or reliability. To advance its design, in this paper we work on a method to consider the operational costs that have higher impact on the overall OPEX of the system and its resource allocation. Moreover, we propose a new assessment sub-algorithm that replaces the previous based on THR with the intention of breaking the barriers and improving the efficiency of the target system.

III. Improved Decision Method

In this section we describe the method we propose to use with ARCA to adapt the amount of resources assigned to a virtual computer and network system in order to minimize its OPEX. As introduced at the beginning of this paper, we propose to consider a complex cost function that involves the amount of resources (servers) used by the virtual system, the amount of operations requested by the controller and executed by the underlying infrastructure, and the amount of packets that are discarded because of high system load.

The main reason to consider such composite function is that, in some situations, the controller is allowed to make no action and thus reduce the volatility of the system and therefore the forthcoming pressure on the total OPEX induced by underlying infrastructure provider, even at the risk of reducing the quality of the network service to some extent (e.g. discarding some packets). This behavior is controlled by internal parameters specified by the administrators/operators of the virtual system to indicate the sensitiveness of the system to packet drops and over-allocations.

For instance, when using the decision method proposed here, ARCA is enhanced to be able to endeavor three basic but different operational environments:

[1] https://www.opendaylight.org/use-cases-and-users/by-function/network-resource-optimization

[2] https://www.sdxcentral.com/sdn-nfv-use-cases/virtual-customer-edge/on-premise-vcpe

- First, the virtual network system implements a service that tolerates packet drops and any rate of resource adjustment requests to the infrastructure but resource usage is very expensive. In this scenario, the control system will be indicated to minimize the amount of resources, allowing the controller to somewhat increase the volatility of the resource allocation or suffering a higher packet drop rate.
- Second, the virtual network system implements a sensitive service (e.g. an emergency support service) that must attend the clients, even when it implies increasing the OPEX. In this case, the control system will be instructed to induce some amount of resource over-allocation and have any rate of adaptation to avoid any packet drops and the consequent disruption of the service.
- Third, the virtual network system is implemented over an infrastructure provider that charges for executing changes of the virtual system. In this situation, the control system is instructed to avoid making frequent resource adjustment requests as much as possible, so it will avoid some adaptation operations even when it induces some packet drops and/or over-allocations.

Considering these, as well as various combinations or more-complex cases, we design our decision method as defined by the highly abstract pseudocode shown in Algorithm 1. It works as follows. First, the decision subsystem retrieves the latest event description (severity of the event) from all external detectors that have sent notifications to the controller through its collector subsystem, as described in the previous section. All these events are summarized to just one and common event by correlating their contents and choosing the most frequent severity as the resulting one. Then, the amount of resources (servers) required to respond such event is obtained from the *anticipator* by providing both the current time in terms of seconds since epoch, day of week, etc. and the severity of the event.

The *anticipator* is derived from a learning and regression method based on K Nearest Neighbours (KNN), which is a well known method for mapping complex vectors of features. In our case, it maps the combination of the time features with the severity of the event notified by the external detectors to the amount of resources required by the controlled system (respectively *NOW*, *cevent*, and *nra* in lines 9 and 24). The main modification we applied to the learning method is the limitation of the number of vectors it keeps to reduce the amount of memory and CPU it requires to perform its task. When the amount of vectors, which are introduced by *set* (line 24) is higher than some value (particularly tuned to around 400 units), some vectors are discarded and the overall count is reduced to the half. The vectors that are kept are carefully chosen by checking their influence on the overall regression, a similar concept to those *support vectors* chosen by the SVM method. The age of the vectors is also taken into account, so the anticipator is able to adapt its beliefs to changes in its environment as fast as possible. Finally, the resulting vectors

Algorithm 1 Decision procedure.

```
 1: procedure DECIDE(e : operation_environment)
 2:     enforcements ← 0
 3:     while True do
 4:         events, loads, drops ← [], [], []
 5:         for all detector ∈ e.detectors do
 6:             events ← events ∪ [detector.event]
 7:         end for
 8:         cevent ← SUMARIZE(events)
 9:         nra ← ANTICIPATOR.GET(NOW, cevent)
10:         if nra ≠ cra and
                DROP_COST((nra − cra) ∗ DPS) >
                    INFRA_COST(enforcements + 1,
                        nra − cra) then
11:             E.INFRASTRUCTURE.ENFOCE(nra)
12:             enforcements ← enforcements + 1
13:         end if
14:         for all server ∈ e.servers do
15:             loads ← loads ∪ [server.load]
16:             drops ← drops ∪ [server.drops]
17:         end for
18:         cload ← SUMARIZE(loads)
19:         cdrops ← SUMARIZE(drops)
20:         if cdrops > 0 and DROP_COST(cdrops) >
                INFRA_COST(enforcements + 1,
                    nra − cra) then
21:             nra ← GET_RES_AMOUNT(cload, cdrops)
22:             E.INFRASTRUCTURE.ENFOCE(nra)
23:             enforcements ← enforcements + 1
24:             ANTICIPATOR.SET(NOW, cevent, nra)
25:         end if
26:     end while
27: end procedure
```

are re-introduced to the learning method together with the new input in order to be ready for the next anticipation request.

After obtaining the anticipated amount of servers (*nra* in line 9), the procedure checks if such amount is worth to be enforced by comparing the cost of the potential packet drops to the infrastructure cost of adding a new server to the system. The former is derived from the difference between *nra* and the current amount of resources assigned to the system (*cra*), multiplied by the rate of drops per server (*DPS*, which is directly related to the clients (users) of the system. The latter is derived from the execution of an additional operation, recorded as *enforcements*, and the incorporation of additional resources to the system (*nra* - *cra*). Both are subject to the weighting established by system administrators. When the adaptation cost is assumed, the *nra* is enforced to the infrastructure (line 11) and such enforcement is recorded (line 12).

In addition to the anticipation procedure described above, the decision algorithm we propose in this paper *assesses* and *corrects* the amount of resources anticipated by checking the status of the system after they have been enforced (lines 20 to 25). This makes our proposal a closed-loop control and

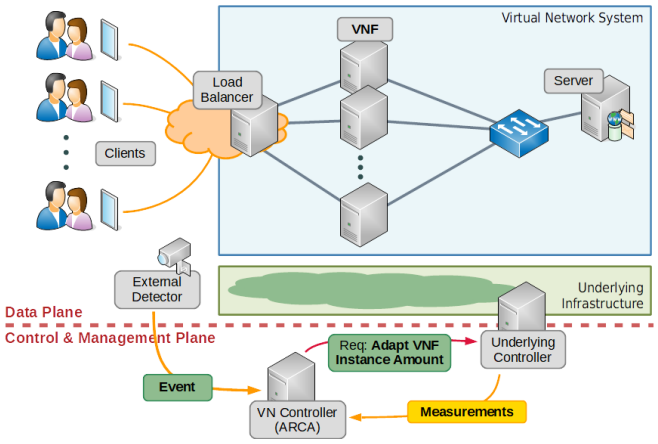

Figure 2. Evaluation topology: Virtual network and control/management elements.

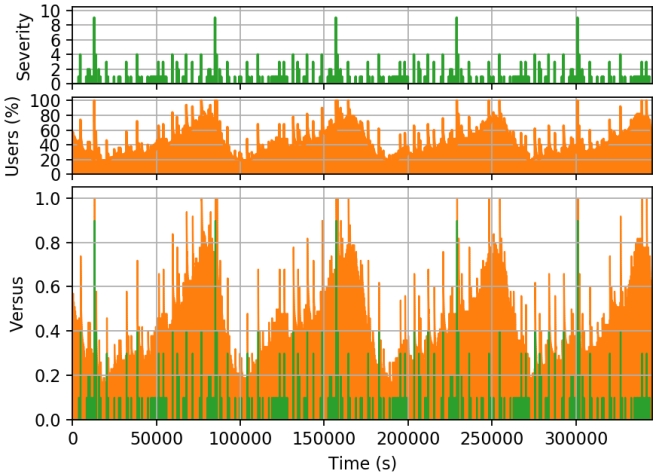

Figure 3. Model of the dataset used in the evaluation.

decision method. It is executed in parallel to the anticipation procedure described above. To perform the assessment and correction, this sub-procedure first obtains from the collector the latest readings of load and dropped packets for the servers that it is controlling. Then, it summarizes both sets of readings by calculating the average load and average drop rate respectively to get the common load (*cload*) and common drop rate (*cdrops*) that represent the current state of the system.

Once obtained the state of the system, this sub-algorithm checks if there are any current drops, so the drop rate is bigger than zero, and, if so, it checks if the cost of such amount of drops is bigger than the cost of contacting the underlying infrastructure provider to request the adaptation of the virtual network system. Both *drop_cost* and *infra_cost* represent functions configured by system administrators to specify the cost of each operation. The drop cost is calculated in the same way than it was calculated for the anticipation sub-algorithm. If the cost of the drop rate is bigger, the decision procedure calculates the required amount of resources to overcome such rate and resolve the corresponding problem of system load (line 21). Then, it enforces the new amount of resources onto the underlying infrastructure, by sending a request to the infrastructure provider, and accounts it (lines 22 and 23). Finally, the sub-algorithm teaches the *anticipator* about the association of the latest event and the *actual* resource amount required to attend them.

It is worth to mention that, in both sub-algorithms, *nra* can be bigger or lower than *cra*, because it is calculated towards optimizing the amount of resources assigned to the system. Anyway, even in the case that *nra* is lower than *cra*, the algorithm we propose checks the potential drop rate and compares it with the operational cost (enforcement + usage). In such situations, the control system can even induce some drops to reduce the overall cost. This is achieved when *nra* - *cra* is a negative value (lines 10 and 20).

IV. Evaluation and Results

We can easily find out that the complexity of the algorithms presented above has an order of $O(n)$, where n is bounded to the sum of the number of external detectors and the number of servers assigned to the system. For most systems, the order is very low. Otherwise, they can be split in separate sub-systems and assign hierarchical controllers to them. Anyway, in order to demonstrate its feasibility and overall qualities, we evaluate its performance in terms of the interaction with the controlled system. We use the amount of servers assigned to the virtual network, the amount of infrastructure requests, and the amount of packets discarded by those servers.

A. Setup: Virtual Network and Traffic

To carry out the evaluation we first build a simulation model based on the implementation of ARCA with the decision algorithm discussed above. We set the general administrative policies to avoid drops as much as possible but without incurring high amount of infrastructure cost, although leaving some liberty to the amount of over-allocations to obtain results that can be comparable to those from other solutions. This is then enclosed in a simulation environment that provides an operational environment, the virtual network system, for the controller to demonstrate its capabilities. The simulation is built with SimPy [13], which is a powerful and highly customizable discrete event simulation framework for the Python programming language.

As shown in Figure 2, the operation environment we use in our evaluation is mainly built around the adaptation of VNFs to the changing requirements set by the behavior and amount of clients using the network services implemented by those VNFs. To facilitate the adaptation, the system incorporates a transparent load balancer that is reconfigured after each adaptation of the system. Furthermore, the environment separates the data plane and the control and management plane, so it incorporates all required elements to implement the target scenario. A special element, the external detector,

although residing in the data plane, it sends notifications to the control plane. On it, the virtual network controller based on ARCA receives such notifications, as well as the measurements obtained from the controller of the underlying infrastructure, makes a decision and enforces it back to the underlying infrastructure to adapt the VNF in response to changes in the environment.

The evaluation is then executed for different datasets we generate, as detailed below, following patterns found in real environments. Each dataset includes, for each moment of time (tick), the amount of clients that are accessing the system (users) and the notification from the external detector, which includes a determination of the *severity* of the event. In this manner we use three datasets: DS-7, DS-8, and DS-8b.

DS-7 is generated by considering that users access the system mainly after an external event occurs, as in their response to emergency situations [14]. Thus, unless an event occurs, this dataset is mainly flat, meaning that there is little or no activity from the users to the network. DS-8, as depicted in Figure 3, is derived from the daily pattern that occurs in real networks (increasing during the day, lowering during the night). It includes the percentage of users accessing the system for every second during four days. However, it has been altered by the generator to include the reaction from users to external events, as done with DS-7. Finally, DS-8b is the same dataset than DS-8 but without the notification of events. This basically means that for DS-8b the external detector will not have any role and the decision approach must rely solely on metrics obtained from the network traffic.

B. Results

To get the evaluation results we run the simulation with the aforementioned datasets using two different approaches for managing the system: ARCA running the decision proposal described in this paper and represented in Algorithm 1, and THR, as defined above, set to 25% and 75% for the lower and higher thresholds respectively. From these executions we retrieve several measurements to evaluate the performance of each solution.

First we measure the server over-allocation as the difference of the optimum amount of servers assigned to the system, which is directly derived from the amount of users accessing the system and the portion of server that is required by each user, and the amount of servers that is currently assigned to the system. 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 keeps improving the behavior of the threshold-based rules solution, as we already did in previous work, but also demonstrates the relevance of the event notifications sent by external detectors in the improvement of the cost of the system. First, we summarize the measured data in Figure 4. All plots in it show the cumulative distribution function (CDF) of each metric, i.e. server over-allocations (TSO), amount of infrastructure requests (Inf. Reqs.), and packet drops (Drops). The first row shows the whole range CDF for each metric but, since plots are too tight, especially for the drops plot, we included two levels of zoom. Each level of zoom shows a particular section of the CDF that reveals specific information.

Table I
RESULTS: TOTAL AMOUNTS OF TSO, INF. REQS., AND DROPS.

Dataset	Metric	THR	ARCA
DS-7	TSO	24540	5134
	Inf. Reqs.	1827	1367
	Drops	41647	4248
DS-8	TSO	1546308	126546
	Inf. Reqs.	25146	13392
	Drops	17859	6839
DS-8b	TSO	1546308	104558
	Inf. Reqs.	25146	10524
	Drops	17859	12620

For the TSO, we can clearly identify that the THR solution behaves behind ARCA for all datasets. This means that ARCA has less level of over-allocation to a high degree, especially for the DS-8 and DS-8b, to which THR has demonstrated a very high level of over-allocation. For DS-7, the THR is close to ARCA but, observing the zoom 1 and 2, we clearly see that, for each moment of time, THR is consistently allocating between 2 and 7 servers more than ARCA. Moreover, in zoom 2 we can see that ARCA fits the demands perfectly of DS-7 (0 servers over-allocated) for more than 96 % of the time. As shown in zoom 1, ARCA *jumps* to allocate 2 servers in DS-8 for 75 % of the time and DS-8b for 78 % of the time. This means that, although a dataset like DS-8 is more disparate, as clearly demonstrated by the results given by THR, ARCA is still able to be close to the optimum. The difference from DS-8 and DS-8b, as nuanced below with absolute values, clearly states that the notifications from external events improve the server allocation for 3 % of time.

Regarding the amount of requests to the infrastructure, as depicted in the second column of the figure, we can see that, for most of the time, both solutions keep the infrastructure untouched (0 requests) for all datasets, so effectively minimizing the operational cost. After zooming on the results, we see that ARCA also outperforms THR in this metric.

Table I shows the total amounts of server over-allocations, infrastructure requests, and packet drops. On it we clearly see that ARCA improves the threshold-based solution (*THR*) in terms of server over-allocation by reducing the amount of servers over-allocated to around the fifth for DS-7, to around the twelfth part for DS-8, and around the fifteenth part for the DS-8b. Also, it shows that there are around 20 % more servers allocated to the virtual network system when the external events are considered in the decision procedure (DS-8 vs DS-8b).

Regarding the amount of infrastructure requests, ARCA also improves THR by reducing them around 25 % for DS-7, almost 50 % for DS-8, and around 60 % for Ds-8b. It is woth to

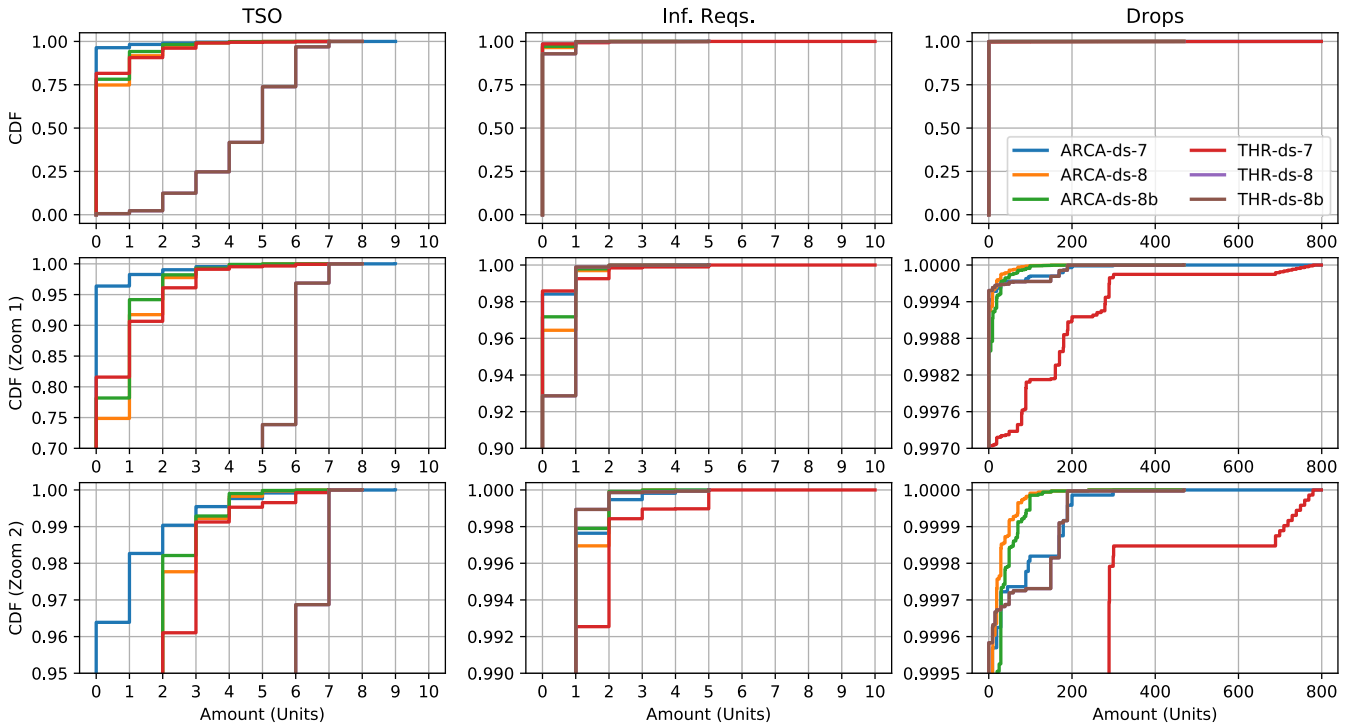

Figure 4. Results obtained from the evaluation.

mention that, although those are high figures, the cost induced by the amount of infrastructure requests usually is quite less (weighted down) than other costs. Anyway, a clear reduction is always beneficial for the system and to reduce its volatility. As a side or negative point, when considering the external events, ARCA increments the total amount of requests around 20 %.

Regarding the amount of packets dropped by the system, ARCA is able to reduce it to the tenth part for DS-7, to 40 % for DS-8, and to 70 % for DS-8b. It clearly demonstrates the advantage of the algorithm we propose in this paper which performs the self-assessment based on the cost estimation instead of using a more typical threshold-based method previously used by ARCA [5]. When facing the results obtained from ARCA for DS-8 and DS-8b we clearly see that the total amount of drops is almost the half when the external events are considered into the management decision.

Overall, the results clearly show that the advantage of ARCA with the new decision algorithm is retained over the most classical THR solution, even though the new algorithm drops the threshold-based assessment sub-algorithm in favor of a more risky but optimum cost-focused method. However, the most interesting result is that those results demonstrate the benefits of integrating notifications from external event detectors into the management process in terms of reducing the amount of packet drops experienced by the system and, generally, for taking proper decisions on resource allocation. Nevertheless, in order to reduce the total amount of drops, the amount of servers is higher, so the over-allocations are higher. Also, the sole reactions to external events increment the amount of adaptation requests to the underlying infrastructure.

V. Conclusions and Future Work

In this paper we have presented an algorithm that optimizes OPEX of a virtual network system by determining the amount of VNF instances that must be assigned to it in response to the changing requirements of the system and events reported by external detectors. We have incorporated such algorithm to ARCA [3], [5] and evaluated it on an environment encompassing different client 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 cost in a network 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.

Finally, for the future work we will keep improving the decision mechanism in order to further improve it in therms of reduced over-allocations and drops. Moreover, we will incorporate more sources of external events, such as different types of physical detectors and even Big Data, in order to increase the richness of the deliberation and potentially improve the resulting adaptations of the controlled system, especially confronting uncertain situations.

REFERENCES

[1] A. Pras, J. Schonwalder, M. Burgess, O. Festor, G. Perez, R. Stadler, and B. Stiller, "Key research challenges in network management," *IEEE Communications Magazine*, vol. 45, no. 10, pp. 104–110, 2007.

[2] 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.

[3] P. Martinez-Julia, V. P. Kafle, 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.

[4] T. Lorido-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.

[5] P. Martinez-Julia, V. P. Kafle, and H. Harai, "Anticipating minimum resources needed to avoid service disruption of emergency support systems," in *Proceedings of the 21th ICIN Conference (Innovations in Clouds, Internet and Networks, ICIN 2018)*. Washington, DC, USA: IEEE, 2018, pp. 1–8.

[6] A. Galis, S. Clayman, L. Mamatas, J. R. Loyola, A. Manzalini, S. Kuklinski, 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.

[7] 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.

[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] ETSI AFI GS, "Autonomic network engineering for the self-managing Future Internet (AFI); Generic Autonomic Network Architecture (An Architectural Reference Model for Autonomic Networking, Cognitive Networking and Self-Management)," 2013, ETSI GS AFI 002.

[12] U. Thissen, R. van Brakel, A. P. de Weijer, W. J. Melssen, and L. M. C. Buydens, "Using support vector machines for time series prediction," *Chemometrics and Intelligent Laboratory Systems*, vol. 69, no. 1–2, pp. 35–49, 2003.

[13] N. Matloff, "Introduction to discrete-event simulation and the simpy language," Dept. of Computer Science, University of California at Davis, Tech. Rep., 2008.

[14] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake shakes twitter users: Real-time event detection by social sensors," in *Proceedings of the 19th International Conference on World Wide Web*. New York, NY, USA: ACM, 2010, pp. 851–860.