Evaluating Allocation Heuristics for an Efficient Virtual Network Function Chaining

Martin Andreoni Lopez, Diogo M. F. Mattos, and Otto Carlos M. B. Duarte
Grupo de Teleinformática e Automação - Universidade Federal do Rio de Janeiro (COPPE/UFRJ)
Rio de Janeiro, Brazil - Email:{martin,diogo,otto}@gta.ufrj.br

***Abstract*—Enterprise networks widely deploy middleboxes to apply load-balancing techniques, to enforce policy compliance, and to improve security. Middlebox platforms, however, are closed systems and expensive. In turn, Network Function Virtualization (NFV) allows to deploy packet-processing middleboxes as virtual network functions, and to decouple the function from the physical realization. In this paper, we address the challenge of efficiently chaining virtual network functions. We propose and compare four heuristics for allocating virtual network functions over a network topology. Our proposal focuses on a greedy algorithm that allocates on demand a sequence of virtual network functions. We compare our four heuristics: (i) minimum introduced latency between source and destination nodes; (ii) minimum resource usage on the network nodes; (iii) the most central nodes first; and (iv) weighted decision between minimum latency and resource usage. We simulate our proposal over a real network topology, and the results show that we allocate 53% more requests when using the resource usage heuristic, and we reduce into 52% the average delay when using the latency heuristic.**

I. Introduction

One of the main pillars of the TCP/IP stack is the end-to-end communication over a simple-core network [1]. Nevertheless, today's enterprise networks rely on middleboxes, *i.e.* packet-processing nodes in the core of the network that improve security, such as firewalls; improve performance, such as load-balancers; or reduce bandwidth costs, such as proxies [2]. Middleboxes are usually dedicated hardware nodes, which perform a specific network function. Hence, middlebox platforms are actually expensive, closed, and hard to be extended [3]. In this way, the Network Function Virtualization (NFV) comes to leverage standard virtualization technology into the network core, and to consolidate network equipment into commodity server hardware [4]. In NFV, the network functions are deployed into virtualized environment and, thus, called Virtual Network Functions (VNF).

When considering the deployment of middleboxes as Virtual Network Functions, a key challenge is the Service Function Chaining (SFC) [5]. The SFC problem stands for the requirement of traffic to pass through multiple middleboxes for packet-processing in a previously defined order. It becomes harder when considering the NFV environment, because allocating virtual network functions over the physical nodes have to consider the packet-processing chaining order among all VNFs in the traffic path, as shown in Figure 1. Therefore, two

This work was sponsored by CNPq, CAPES, and FAPERJ.

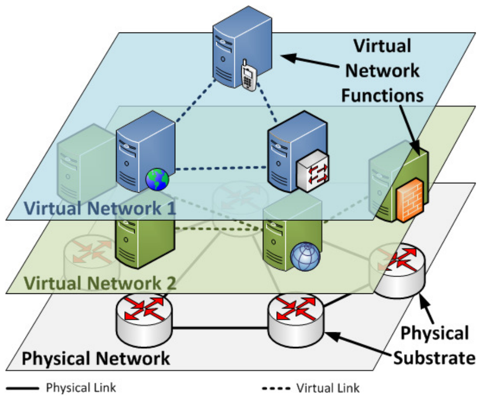


Figure 1. Two separated plans of virtual network functions decoupled from the underlying physical substrate.

main constraints for VNF chaining are to reduce the delay introduced by placing the VNFs on the network topology, and to allocate VNFs on physical nodes that can provide enough resources to run the hosted functions [6].

In this paper, we propose a scheme for placing and chaining Virtual Network Functions over a network topology according to four different heuristics. The first heuristic places the VNF nodes into physical nodes that introduce the minimum delay between the traffic source and destination. The second heuristic searches for the best placement of VNF nodes considering the nodes that have the biggest amount of available resources and, thus, places the VNF over the most available node. This approach increases the number of accepted requests of VNFs in a network. The third heuristic places the VNF nodes according to the betweenness-centrality of the topology nodes. In the betweenness-centrality approach, the requests are primarily responded by allocating the most central nodes on the topology, which reduces the introduced delay. However, as the resources of the most central nodes are used, the following requests are allocated into peripheral network nodes, introducing a greater delay on the VNF chaining. The fourth heuristic weighs the available resources and the introduced delay for each physical node. This approach allocates the VNFs on the nodes that present the greatest probability of supplying enough resources and the lower delay. We deploy a greedy algorithm for all three approaches and we simulate the allocation of VNFs over a real network topology. Our results show that we are able to answer up to 53% more requests when using maximum resource usage approach, and we reduce up to 52% of the packet-processing delay when considering the minimum delay approach.

Chaining virtual network functions is essentially an optimization problem that recalls the facility location problem [5]. Previous works propose linear programming models to search for a solution for VNF chaining, considering resource allocation and the routing between virtual instances [5], [7], [8]. Other works propose to outsource network functions to the cloud [2], [9], but do not specify an algorithm for placing VNFs on the network. Moreover, there are also works that place specific-purpose VNFs on the network, such as traffic sensors or network controllers [10], [11]. In turn, our proposal uses a greedy algorithm to place general-purpose VNFs over a network topology and compares different heuristics. The proposed scheme estimates the resources at each physical node on the network and, then, places the VNFs according to the available resources of the physical nodes and the requested resources for the VNF.

The remainder of the paper is organized as follows. Section II describes the related works. In Section III, we define the problem of chaining Virtual Network Functions. The proposed heuristics for allocating VNFs on a network are discussed in Section IV. Our simulations and results are shown in Section V. Section VI concludes the paper.

II. Related Works

Virtual Network Function chaining is currently a trend topic in research. Several researches deal with the optimization problem to place a set of VNFs [7], [5], [8]. Addis *et al.* propose a mixed integer linear programming formulation to solve the VNF placement optimization from the Internet Service Providers (ISPs) point of view [7]. In a similar way, Bari *et al.* use a Integer Linear Programming in order to optimize the cost of deploying a new VNF, the energy cost for running a VNF, and the cost of forwarding traffic to and from a VNF [5]. A Pareto optimization is used for placing chained VNFs in an operator's network with multiple sites, based on requirements of the tenants and of the operator [8]. In addition, other works propose the optimal placement of specific VNF [12], [13], [11]. A virtual Deep Packet Inspection (vDPI) placement is proposed by Bouet *et. al.* to minimize the cost that the operator faces [12]. In a previous work [13], we proposed the placement of an Intrusion Detection and Prevention System (IDPS) by a heuristic that maximize the traffic passing through each node. In another previous work [11], we proposed a heuristic to optimize the placement of distributed network controllers in a Software Defined Network environment. Nevertheless, none of these works considers the trade-off of the customers requests and infrastructure provider availability.

Estimating resource usage for optimizing allocation has been proposed in many other contexts. Sandpiper [14] is a resource management tool for datacenters. It focuses on managing the allocation of virtual machines over a physical infrastructure. Other proposal that estimates the resource usage for allocating virtual machines in a datacenter is Voltaic [15]. Voltaic is a management system focused on cloud computing which aims to ensure compliance with service level agreements (SLAs) and optimize the use of computing resources.

In our work, we propose three heuristics in order to minimize the delay between source and destination nodes, and one to maximize the resources use, achieving the best customer Quality of Experience (QoE). Another heuristic is proposed to minimize the resource usage on the network nodes to increase Infrastructure Provider (IP) benefits, and finally, a heuristic for using the most central nodes first to improve customer QoE and IP benefit. We compare the four proposed heuristics with a greedy algorithm and we run simulations over a real Service Provider topology.

III. The Virtual Network Function Chaining Problem

Service chaining simply consists of a set of network services interconnected through the network infrastructure to support an application requested by the customer. Traditionally, Service Function Chaining (SFC) was built in the early years of high-performance computing, being rigid and static, installed at fixed locations in or at the edge of the carrier core network [16]. The SFC is enhanced with the advent of NFV that enables operators to configure network services dynamically in software without having to make changes to the network at the hardware level. Therefore, virtualized NFs (VNFs) can be placed when and where needed. This implies an optimization problem that uses VNFs or services as a graph to address the requirement for a better utilization of resources, for latency decrease and for network optimization [7].

Typically, network flows go through several network functions as shown in Figure 2. When a NF or a set of NFs are specified, the flows traverse these NFs in a specific order so that the required functions are applied to the flows. Normally, the NFs demand certain dependency among them that should be chaining to the traffic in a network with a specific order. Depending on how each function is set in the chain, it affects in the network traffic, application performance and latency.

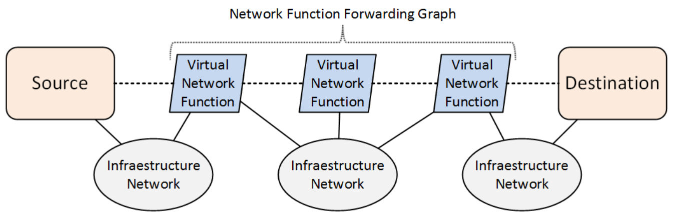


Figure 2. Example of Network Function Forwarding Graph. Three Virtualized Network Functions (VNF) are chained to establish a service from the source to the destination. The VNFs are executed over the physical infrastructure network.

The VNF chaining problem consists, thus, of two sub-problems. The first sub-problem is the placement problem, in which the VNF instances are allocated onto physical nodes. This problem consists into finding a physical node that has enough resources to host the VNF, serving it with the requested resources. The second sub-problem consists into a routing challenge, because mapping a set of VNFs over a physical topology should consider the iteration among all VNFs. The routing problem should ensure that the traffic between VNFs would always have a limited delay, and enough bandwidth. If any of these constraints are not satisfied by the chaining scheme, the VNF request may not be accepted. Therefore, deciding for accepting VNF requests is also part of the VNF chaining problem.

IV. The Proposed VNF Chaining Scheme

Our proposal considers a scenario in which the requests for a new Virtual Network Function arrive to a network manager, who has to allocate them into the available nodes. We consider as a request a sorted list of VNFs that describes the order in which traffic has to be processed. Therefore, the allocation of the request on the network has to consider the order of the VNFs as well as the source and the destination of the traffic handled by the set of VNFs in the request. We also consider that, when allocating a VNF over a physical node, the physical node must have enough resources to answer the needs of all hosted VNFs. Our proposed scheme is composed of two main phases. The first phase estimates the resources available on the physical nodes and the resources requested by the VNFs. The second phase runs a greedy algorithm that takes as input the VNF requests as they arrive, and then it places each VNF on a physical node that have enough resources. Our greedy algorithm considers four different heuristics to place the VNFs on the network.

Estimating the available resources on physical and virtual nodes is challenging because there are three main resources, which should be considered: CPU, memory, and network. In order to summarize all resources into one single variable, we consider the `Volume` metric introduced by Wood *et al.* [14]. We consider that the `volume` of a physical server is 1, and the `volume` of each VNF is:

$$Volume_{VNF} = \frac{1}{1-cpu} * \frac{1}{1-mem} * \frac{1}{1-net}, \qquad (1)$$

where cpu stands for the normalized CPU usage of the VNF, mem, for memory, and net, for network. Thus, for each VNF, the `volume` metric is the ratio of the resources on physical node that the VNF is requesting. The VNF `volume` ranges from 0 to 1, where 1 means that a VNF is requesting an entirely available physical node to be installed. If the requested volume of any resource is equal to one, it is not possible to allocate the VNF. This is due to some physical resources, such as CPU, is reserved to the host Operating System and, thus, it cannot be allocated to a VNF.

Following, on the second phase, we run a greedy algorithm that allocates a VNF request as it arrives. Our algorithm adopts one of the four heuristics:

- **minimum latency**, in which the algorithm chooses the node that introduces a minimum delay to the path, in comparison to the previous selected nodes to host the other VNFs, or the source of the traffic;
- **maximum usage of resources**, in which the algorithm chooses the node that has the biggest amount of available resources to host a VNF, without considering the routing constraints between the already placed VNFs;
- **most central nodes**, in which the algorithm chooses to place the VNF into the most central node, i.e. the node that presents the greatest betweeneess-centrality value, and has enough resources to host the VNF;
- **weighted latency and resource**, in which the probability of choosing each node for hosting a VNF is weighted based on the latency that it introduces to the path and the available resources that it has. The weight of the node i is given by:

$$w_i = \left(1 - \frac{lat_i}{max_{j \in N}(lat_j)}\right) \times \left(\frac{rec_i}{max_{j \in N}(rec_j)}\right),$$

  where lat_i stand for the latency introduced by node i, rec_i is the available resources in node i, and N is the set of all nodes in the network. The greedy algorithm searches for hosting VNFs on the nodes that have the biggest w_i value first.

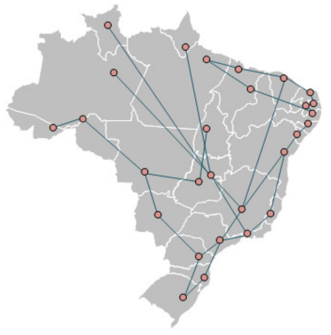
Our proposal works as follows. First, the network manager receives a sorted list with the requested VNFs, the source and the destination of the traffic, and the requested resources of each VNF. Then, our algorithm selects the first VNF on the request and search for a node in which the requested resources meet the available resources on the physical node. To verify if the physical node has enough resources, the algorithm compares the VNF `volume` with the available `volume` of the physical node. If the available `volume` is greater than the requested, the VNF is installed over this candidate physical node. Otherwise, the algorithm selects the next physical node until finding an available node. If there is no available physical node that meets the requested VNF `volume`, the VNF request is entirely rejected and no VNF is allocated. After mapping all VNFs over the physical nodes, the VNFs are installed and the `volume` of each physical node that receives a VNF is decremented by the `volume` of the VNF it hosts.

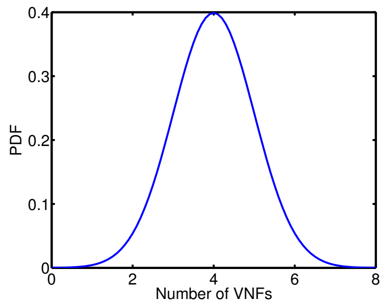
It is worth noting that a VNF request should be entirely accepted or rejected. If the algorithm realizes that there is not enough resource in any node in the network to complete the VNF request allocation, the request is completely rejected and no node is allocated on the network. We adopt the all-or-nothing approach, because a partially allocated VNF request do not implement all packet-processing functions that it supposed to deploy, thus it is not a feasible solution.

V. The Evaluation of the Proposal

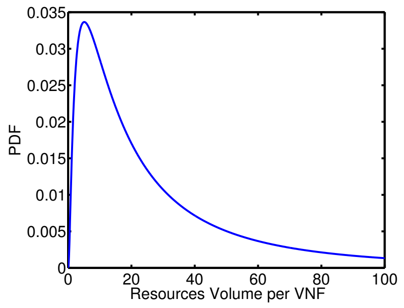
We evaluate the proposed greedy algorithm through simulation. We implemented a simulator, written in Python language, in which the VNF requests arrive at each simulation step. At a simulation step, the proposed scheme evaluates the used resources of each physical node on the network topology and generates the available $volume$ metric for each physical node. Then, our scheme gets the next request and allocates the VNFs on the network according to one of the proposed heuristics. We consider as the initial conditions of the topology when all nodes are unallocated with volume equal to one, it means 100% of available resources.

We establish the proposal evaluation in two steps. The first step is to simulate the customer Virtual Network Functions (VNFs) requests. The requests are generated based on a normal distribution with $\mu = 4$ and $\sigma = 1$, as it is shown in Figure 3(b). In this way, the customer requests are generated randomly associating different number of VNF for each request. We model the number of VNFs in each request based


(a) Brazilian *Rede Nacional de Ensino e Pesquisa* (RNP) real topology, with 31 vertex and 34 edges.



(b) Probability Density Function of the number of VNFs in a request.



(c) Probability Density Function of the `volume` of resources used by each VNF.

Figure 3. Simulation scenario. a) The VNF chaining scheme was evaluated over the topology of the RNP network. b) The modeled number of VNFs in each request follows a normal distribution, with mean equals to 4, and standard deviation equals to 1 ($\mu = 4$ and $\sigma = 1$). c) The modeled resource consumption of each VNF follows a lognormal distribution, with mean equals to 3, and standard deviation equals to 1.17 ($\mu = 3$ and $\sigma = 1.17$). The lognormal distribution is truncated at 100, which represents the maximum `volume` usage ($volume = 1$).

on the studies realized by Sekar *et al.* [3]. We consider that a Virtual Machine (VM) deploys each VNF. Then, we simulate the resource consumption of each VNF. We model the resource consumption based in a lognormal distribution, with $\mu = 3$ and $\sigma = 1.17$, as it is shown in Figure 3(c). The resource consumption of each VNF is modeled as a truncated lognormal distribution because it should reflects the behavior of the middleboxes, in which it usually uses a small amount of resources. The distribution is truncated at 100, because it is the maximum *volume* that a VNF can assume ($volume_{VNF} = 1$). As result, we obtain the customer VNF request with different resource $volume$ and a selected order of chaining. We highlight that the resources of all the VNFs over a single physical host is never higher than 100%. An example of a VNF customer request is $[VNF_1 = 15\%, VNF_2 = 26\%, VNF_3 = 45\%]; src, dst$, where the number of VNF are randomly chosen, and the src and dst are the source and destination of each chaining request. The source and the destination are uniformly chosen on the network topology. Our model does not consider that VNFs quit the network after being allocated.

The second step of the proposal evaluation asserts the optimization heuristics. Our experiments evaluate the placement heuristics against the RNP (Nation Research Network) real network topologies[1], which has 31 nodes distributed along the Brazilian territory, Figure 3(a). Using a greedy algorithm, we place the VNF in different nodes and we evaluate the amount of VNF requested for each heuristic. We consider only the propagation delay between the nodes to estimate the latency between the nodes. The propagation delay is estimated according to the distance between nodes. We consider the propagation speed of $2 \times 10^8\ m/s$, which is commonly used in other works [17]. The distance between each node is calculated based on the geographic location of each node.

The results in Figure 4 show that the maximum resource allocation heuristic is the one that accepts more requests, around 53% more request than the betweenness-centrality heuristic. The betweenness-centrality heuristic is the simplest to calculate, as it only depends on the topology characteristics. Nevertheless, it is the one that rejects the greatest number of requests. In addition, the latency heuristic presents a better performance when compared with the betweenness-centrality, however, this heuristic shows the worst complexity when executed. It is worth noting that, although the maximum resource allocation heuristic optimizes the acceptation rate of VNFs on the network, it does not consider the routing constraints between VNFs. In this way, it increases the delay introduced by the deployment of network functions as VNFs because the packets may pass through distant nodes in order to follow the entire packet-processing path.

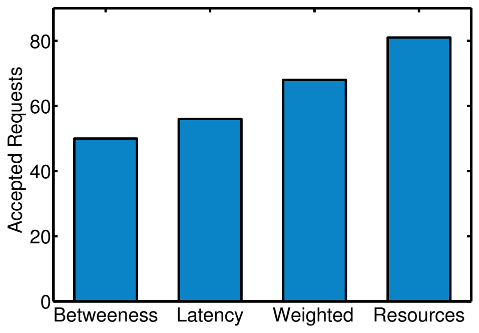
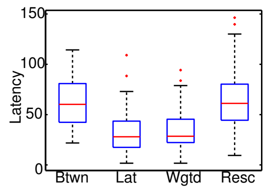


Figure 4. Number of accepted requests for each proposed heuristic. The maximum resource allocation heuristic is the one that accepts more requests. The betweenness-centrality heuristic is the simplest to calculate, as it only depends on the topology characteristics, but it is the one that rejects the greatest number of requests.

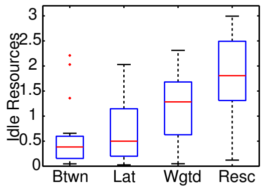
We also compare the dispersion of the latency distribution of the allocated VNFs for each heuristic. As shown in Figure 5(a), the minimum latency heuristic introduces the lowest average delay on the packet-processing path. The maximum resource usage heuristic is the one that presents the greatest dispersion into the latency distribution thanks to ignoring the latency concerns when placing the VNFs. This result shows that the latency heuristic reduces 52% the average delay when compared with the betweeness-centrality heuristic. Moreover, the latency heuristic also achieves the greatest number of accepted VNF requests, which have the minimum latency,

[1]The topology graphs are available at The Internet Topology Zoo (http://www.topology-zoo.org/).

even when compared with maximum resource allocation that achieves to allocate more requests than all others do. Figure 5(a) also reveals that the latency is limited in all heuristics and, even in the highest delay scenario, it is still limited to 150 ms. Figure 5(b) shows the remaining resources after all VNF allocation. Although the maximum resource heuristic instantiates more VNFs, it presents the biggest amount of idle resources. Nevertheless, it is also the most distributed idle resource pattern, which implies a load distribution between all physical nodes.



(a) Dispersion of the latency distribution into the allocated VNFs.



(b) Dispersion of the remaining idle resource distribution after allocating all VNFs.

Figure 5. Simulation results. `Btw` stands for the betweeness-centrality; `Lat`, for latency; `Wgtd`, for weighted latency and resource; `Resc`, for maximum resource. a) The minimum latency heuristic introduces the lower average delay on the packet-processing path. The maximum resource usage heuristic is the one that presents the greatest dispersion into the latency distribution thanks of ignoring the latency concerns when placing the VNFs. b) The Maximum Resource heuristic presents the most distributed remaining resources.

Comparing Figures 4 and 5(a), we emphasize that the greater dispersion of the latency achieved by the maximum resource allocation heuristic is a reflect of the greater number of accepted requests, when compared with the minimum latency heuristic. It is worth noting that all heuristics are compliant with the resource constraints. Therefore, choosing among the four heuristics, when designing a NFV environment, should consider the goals of the network manager. In case of the main goal is to maximize the number of accepted VNFs, the selected heuristic should be the maximum resource allocation, as it still finds good results of latency between nodes. Nevertheless, if the main goal is to achieve the maximum performance of VNFs, the minimum latency heuristic is best choice. A tradeoff solution is the weighted latency and resource solution, which keeps the bounded latency, and it increases up to 22% the acceptance rate when compared to the latency heuristic.

VI. Conclusion

Network Function Virtualization (NFV) is a promising technique that enables to decouple the network function from its physical realization by virtualizing the network equipment. In this sense, network functions are deployed within virtual environment and, thus, called Virtual Network Functions (VNF). VNF commonly implements packet-processing functions, previously implemented by specific-purpose middleboxes. This scenario raises the challenge of placing and chaining VNFs, which is a NP-hard optimization problem. VNF chaining depends on finding a network node, which provides enough resources for the VNF and introduces a minimum delay to the packet-processing path. In this paper, we proposed a VNF chaining scheme, in which a greedy algorithm places the VNFs on the network according to four different heuristics. Our simulation and results show that using a heuristic for placing VNFs on nodes with the biggest amount of available resources increases the acceptation rate of VNF requests by 53%. Moreover, we also show that using a heuristic for introducing minimum delay on the path, we are able to reduce the average packet-processing delay by 52%.

As future works, we will deploy our VNF chaining scheme as an orchestrator of the OPNFV platform, and we will perform experiment into a large-scale NFV network.

References

[1] M. S. Blumenthal and D. D. Clark, “Rethinking the design of the internet: The end-to-end arguments vs. the brave new world,” *ACM Trans. Internet Technol.*, vol. 1, no. 1, pp. 70–109, Aug. 2001.

[2] J. Sherry, S. Hasan, C. Scott, A. Krishnamurthy, S. Ratnasamy, and V. Sekar, “Making middleboxes someone else’s problem: Network processing as a cloud service,” *SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 13–24, Aug. 2012.

[3] V. Sekar, N. Egi, S. Ratnasamy, M. K. Reiter, and G. Shi, “Design and implementation of a consolidated middlebox architecture,” in *9th Symposium on Networked Systems Design and Implementation (NSDI)*. San Jose, CA: USENIX, 2012, pp. 323–336.

[4] H. Jeon and B. Lee, “Network service chaining challenges for vnf outsourcing in network function virtualization,” in *International Conference on Information and Communication Technology Convergence (ICTC)*, Oct. 2015, pp. 819–821.

[5] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, “On orchestrating virtual network functions,” in *11th International Conference on Network and Service Management (CNSM)*, Nov. 2015, pp. 50–56.

[6] D. M. F. Mattos and O. C. M. B. Duarte, “AuthFlow: authentication and access control mechanism for software defined networking,” *Annals of Telecommunications*, pp. 1–9, 2016.

[7] B. Addis, D. Belabed, M. Bouet, and S. Secci, “Virtual network functions placement and routing optimization,” in *IEEE 4th International Conference on Cloud Networking (CloudNet)*, Oct. 2015, pp. 171–177.

[8] S. Mehraghdam, M. Keller, and H. Karl, “Specifying and placing chains of virtual network functions,” in *IEEE 3rd International Conference on Cloud Networking (CloudNet)*, Oct. 2014, pp. 7–13.

[9] R. Laufer, M. Gallo, D. Perino, and A. Nandugudi, “Climb: Enabling network function composition with click middleboxes,” in *Proceedings of the 2016 Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, ser. HotMIddlebox ’16. New York, NY, USA: ACM, 2016, pp. 50–55.

[10] M. Andreoni Lopez and O. C. M. B. Duarte, “Providing elasticity to intrusion detection systems in virtualized software defined networks,” in *IEEE ICC 2015 - Communication and Information Systems Security Symposium (ICC’15 - CISS)*, London, United Kingdom, Jun. 2015.

[11] D. M. F. Mattos, O. C. M. B. Duarte, and G. Pujolle, “A resilient distributed controller for software defined networking,” in *IEEE ICC 2016 - Next Generation Networking and Internet Symposium (ICC’16 - NGN)*, Kuala Lumpur, Malaysia, May 2016.

[12] M. Bouet, J. Leguay, T. Combe, and V. Conan, “Cost-based placement of vDPI functions in NFV infrastructures,” *International Journal of Network Management*, vol. 25, no. 6, pp. 490–506, 2015.

[13] M. Andreoni Lopez, D. M. Ferrazani Mattos, and O. C. M. B. Duarte, “An elastic intrusion detection system for software networks,” *Annals of Telecommunications*, pp. 1–11, 2016.

[14] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, “Sandpiper: Black-box and gray-box resource management for virtual machines,” *Comput. Netw.*, vol. 53, no. 17, pp. 2923–2938, Dec. 2009.

[15] H. E. T. Carvalho and O. C. M. B. Duarte, “Voltaic: volume optimization layer to assign cloud resources,” in *Proceedings of the 3rd International Conference on Information and Communication Systems*, ser. ICICS’12, 2012, pp. 3:1–3:7.

[16] P. Quinn and T. Nadeau, “Problem statement for service function chaining,” Active Internet-Draft, TETF Secretariat, Tech. Rep. RFC 7498, 2015.

[17] d. R. S. Couto, S. Secci, M. E. M. Campista, and L. H. M. K. Costa, “Reliability and survivability analysis of data center network topologies,” *Journal of Network and Systems Management*, vol. 24, no. 2, pp. 346–392, 2016.