Affinity Measurement for NFV-enabled Networks: A Criteria-based Approach

Arthur Selle Jacobs, Ricardo Luis do Santos, Muriel Figueredo Franco, Eder John Scheid,
Ricardo José Pfitscher, Lisandro Zambenedetti Granville
Institute of Informatics — Federal University of Rio Grande do Sul
{asjacobs, rlsantos, mffranco, ejscheid, rjpfitscher, granville}@inf.ufrgs.br

***Abstract*—Network Functions Virtualization (NFV) offers several benefits for Service Providers (SPs), such as mitigating equipment cost and increasing business agility. In NFV-enabled networks, inadequate placement of Virtualized Network Functions (VNFs) creates bottlenecks, impacting negatively on performance. Therefore, network operators must establish affinity and anti-affinity rules to avoid network and processing bottlenecks, and thus comply with Service Level Agreement (SLA) requirements of tenants. Affinity and anti-affinity rules in NFV must be broad and carefully elaborated to maintain service performance. Network operators must consider further than simply resource allocation when identifying affinity among VNFs. The criteria for VNFs affinity varies for different forwarding graphs. Geolocation, latency, packet loss, and bandwidth usage are some examples of criteria that can be considered as indicators of bottlenecks in high traffic networks. In this paper, we propose a solution to measure affinity between pairs of VNFs, based on a weighted set of affinity criteria considered relevant by a network operator. To evaluate the feasibility of our affinity model, we analyze three case studies over an experimental NFV scenario. We conclude that our affinity model can help network operators identify the cause of issues in NFV-enabled networks, as well as it may be used by NFV orchestrators to aid on VNFs migration and embedding.**

I. Introduction

Network Functions Virtualization (NFV) offers several benefits for Service Providers (SPs), such as mitigating equipment cost and increasing business agility [1]. NFV migrates network functions from dedicated hardware to software running on general-purpose servers, often referred to as Virtualized Network Functions (VNFs). Virtual Machines (VMs) are used to host VNFs, which can then be created, migrated, and destroyed on-the-fly. In the end of the day, VNFs provide flexibility and scalability, avoiding ossification and introducing innovation in the network core [2].

In NFV-enabled networks, inadequate placement of VNFs creates bottlenecks, impacting negatively on performance [3]. Network operators establish affinity and anti-affinity rules to avoid network and processing bottlenecks [4], and thus comply with Service Level Agreement (SLA) requirements of tenants. These rules help operators improve service execution and minimize resources waste [5]. Moreover, affinity and anti-affinity rules can be based on several different aspects, such as VNFs minimum resource requirements, latency, number of processed packets, or even network operators needs for the service [6]. Although affinity is a critical issue, to the best of our knowledge, efforts to determine affinity among VNFs have been scarce [7] [8]. Besides, these efforts focus solely on resource allocation, disregarding the service being provided.

Affinity and anti-affinity rules in NFV must be broad and carefully elaborated to maintain service performance. VNFs are chained in a Forwarding Graph (FG) to provide a service (*i.e.*, service chaining), increasing substantially management complexity. Network operators must consider further than simply resource allocation when identifying affinity among VNFs. The criteria for VNFs affinity varies widely for different forwarding graphs. For instance, geolocation can be taken into account to minimize latency and propagation delay among chained VNFs located far from each other, while packet loss and bandwidth usage can be considered as an indicator of bottlenecks in high traffic networks. All this backs up the argument that network operators must be able to select what criteria are relevant when establishing VNFs affinities.

In this paper, we introduce an extendable solution to measure affinity between pairs of VNFs, given a weighted set of affinity criteria considered relevant by a network operator. First, we provide a definition of affinity between a pair of VNFs, to specify the semantics of our affinity measure. Second, we specify an extendable set of affinity criteria for an NFV-enabled network, which an operator may provide weights to, according to the relevance of each criterion. Third, we propose a mathematical solution to measure affinity between two VNFs, based on the criteria and weights provided by the operator. Thus, our solution can help network operators identify the root cause of problems in NFV-enabled networks by analyzing affinity measures between VNFs. In addition, an affinity measure supports the creation of more concise and improved affinity rules, avoiding performance degradation of services. Finally, affinity measures may be used by NFV orchestrators — in addition to network operators — to aid on VNFs migration and embedding.

The remaining of this paper is organized as follows. In Section II, we present the related work regarding affinity in NFV and network virtualization. In Section III, we define a set of admissible affinity criteria and introduce our solution for affinity measure. In Section IV, we evaluate our solution by analyzing experimental scenarios. In Section V, we conclude this paper and present future work.

II. Related Work

Affinity and anti-affinity have been discussed in the context of Cloud-based environments. However, most of current affinity solutions disregard the nature of functions being executed by each VM, focusing primarily on resource allocation. Most commonly, solutions propose affinity relations based on CPU [5], bandwidth [8], or even memory page sharing [9]. Next, we discuss the relevant affinity related solutions for both Cloud and NFV environments.

Chen *et al.* [7] proposed a method to identify affinity relations based on resource demands and dependency among VMs, alongside an algorithm to group these affine VMs. By grouping affine VMs as a unit, they aim to co-locate them on the same physical machine to improve system performance. Despite their solution showing positive results for multi-VM applications, the authors only consider communication patterns to identify affinity, disregarding other criteria relevant in the NFV context, such as latency and FGs.

Yoshida *et al.* [10] propose a Multi-objective Resource Scheduling Algorithm (MORSA) for NFV. They use genetic algorithms to obtain the best possible placement over multiple data centers for VNFs, considering a dynamic set of criteria. These criteria are determined by several plugins inserted in their solutions. They present plugins for common issues in the NFV context, such as minimizing physical machine load, intra-datacenter traffic and protocol requirements for linked VNFs. Although their solution takes into account many affinity related matters intrinsic to NFV environments, it does not present any kind of affinity metric. In addition, that paper also disregards the services being provided by each VNF.

Franco *et al.* [11] present VISION, a visualization platform with multiple interactive and selective techniques for NFV-enabled networks. Network operators may take advantage of VISION to identify problems on the network that impact on the VNFs performances. Also, their solution provide unique perspectives on NFV-enabled networks that assist in recognizing behavioral patterns, allowing a better service for tenants. Although this study allows network operators to identify affinity and anti-affinity relations through visualizations, they do not propose a distinct visualization technique to achieve this objective. Thus, the result from the model we propose can improve their visualizations to identify performance issues.

Yousaf and T. Taleb [12] propose fine-grained resource-aware VM management solution for NFV-enabled networks, based on a reference resource affinity score (RRAS). The authors consider affinity as a correlation between different entities, which for their specific case is the correlation between different Resource Units (RUs) of a VM running a VNF, such as processing, memory, I/O module and storage. Their affinity calculation results on a vector quantity representing the impact of one reference RU (*e.g.*, memory) on other RUs (*e.g.*, processing, storage, and I/O module). This affinity score is calculated for each VM, from time to time, and stored for further analysis. This data can then be used to trace behavior patterns for each VNF, which can be used by the NFV Management and Orchestration (MANO) [12] for short-term and long-term decision making regarding VM deployment and migration, for instance. Although the authors provide an affinity calculation for NFV environments, their solution only provides affinity values for RUs of a single VM, requiring extra work to determine patterns among VMs.

Even though affinity and affinity-related issues have been discussed by several authors, their approaches have focused mainly on computational resources awareness. Consequently, these solutions fall short for NFV-enabled networks, which requires further considerations when defining affinity. In addition to computational resources, requirements such as geolocation, FGs, and service performance, must be taken into account. Furthermore, most current solutions lack in dynamicity, since they do not consider any interaction with operators. In the following section, we present our affinity measure solution, which tackles these issues.

III. Solution

In this section, we present our solution to measure the affinity between a pair of VNFs. To measure that, we must primarily establish the semantics of an affinity relationship. The concept of affinity refers to the correlation of different entities, representing their ability, or inability, to perform when combined in a certain way. The proposed solution considers affinity as an indicative of how well two VNFs operate, either when placed on the same Physical Machine (PM), or when chained on the same FG. Bearing this concept in mind, our solution provides a numerical value that represents the affinity between a pair of VNFs, for each FG both VNFs are chained. This affinity value can be used to help either a network operator to rearrange the VNFs, or an orchestrator better resolve VNF placement problems.

The proposed solution receives as input a list of weights for each affinity criterion (see subsection III-A) and a pair of VNFs, returning a normalized value between 0 and 1, which represents the affinity value based on the input criteria. The input criteria are chosen by a network operator among a set of pre-established criteria. Additionally, one can extend this initial set to include other criteria not considered yet.

We define two sets of admissible criteria: static and dynamic. The former refers to data that can be collected without the need of VNF deployment (*e.g.*, CPU requirements, and VNFs conflicts). The latter relates to information of running VNFs (*e.g.*, memory usage, and latency). Dynamic VNF data can be collected using a monitoring solution for NFV-enabled networks, such as the one proposed by DReAM [13]. It is important to distinct these two types of criterion, due to the nature of the criteria in each set. Static criteria can be used to measure affinity regardless whether the target VNFs are running or not, whereas dynamic criteria can only be used when VNFs are running. The complete set of pre-established criteria is presented below.

Type	Scope	Criterion	Description
Static	PM	Minimum CPU	The minimum CPU requirement, in MHz, is declared on the NSD and should be used when instantiating VNFs for a network service. Notice that if a VNF is instantiated disregarding these requirements, then it could have a negative impact on the service due to lack of resources. If these requirements are not met for the VNF being evaluated, these VNFs would have a lower affinity.
		Minimum memory	The minimum memory requirement, in MB, is also declared on the NSD and should be used when instantiating VNFs for a network service. If these requirements are not met for the VNF being evaluated, these VNFs would have a lower affinity.
		Minimum storage	The minimum storage requirement, in IOPS, is declared on the NSD and should be used when instantiating VNFs for a network service. If these requirements are not met for the VNF being evaluated, these VNFs would have a lower affinity.
	FG	NFV conflicts	Check if the two VNFs are placed correctly according to a list of known VNF conflicts. VNFs with known conflicts should not be chained on the same FG, or placed on the same PM. This criterion's calculation will return 1 if the conflicts are respected and 0.001 if not.
Dynamic	PM	CPU usage	CPU usage is an important metric to monitor the stress on PMs, specially because communication between VNFs hosted by the same PM is made through memory sharing, which causes great stress to CPU. If two VNFs were responsible for a large percentage of the PM CPU usage, then these VNFs would have a low affinity.
		Memory usage	Just as CPU usage, indicates stress levels on PMs. If two VNFs were responsible for a large percentage of the PM memory usage, then these VNFs would have a low affinity.
		Storage usage	This criterion is also an indicator of stress levels on PMs. A higher percentage of storage usage of two VNFs would impact negatively on these VNFs affinity.
	FG	Bandwidth usage	Bandwidth usage is an indicator of how much two VNFs are stressing the links connecting them. If these VNFs were responsible for a large percentage of bandwidth consumption, then they would have a lower affinity.
		Packet loss	Packet loss, just as latency and bandwidth usage, is an indicator of issues in the network. A higher packet loss percentage between two VNFs would cause these VNFs to have a lower affinity.
		Latency	Latency is an indicator of several issues in the network, including large distances between VNFs and bottlenecks in the service. How much latency — and all the other FG graph criteria above — influence the service performance, and therefore affinity, depends on the amount of traffic between the VNFs. If latency is very high and traffic is also high between two VNFs, then these VNFs would have a very low affinity. If latency is very low and traffic is high, then these two VNFs would have a very high affinity. However, if traffic is low between two VNFs, latency, either high or low, would not influence the service much, and therefore, the VNFs would have a somewhat medium affinity.

Table I: Set of criteria.

A. Criteria

In our affinity model, each dynamic and static criterion is labeled regarding their scope: PM or FG. All static criteria are used when calculating affinity, according to the operator's input weights, no matter the scope of the selected criteria. However, dynamic criteria usage depends, in addition to the operator's input, on their scope. If two VNFs are running on the same PM, then the dynamic PM criteria will be considered on the result. If two VNFs are chained on the same FG, then the dynamic FG will be considered on the result. It is important to point out that a VNF may fit in both scopes presented, and therefore, all dynamic criteria will be used.

Table I presents a brief description, type, and scope of all admissible criteria. Table II shows the affinity calculation equation for each criterion. All affinity measures in Table II follow the same principle: if two VNFs are performing well together, and respect all resource requirements, the resulting affinity will be closer to 1; further, if there is any performance or resource allocation problem related to those two VNFs, the resulting affinity measure will be closer to 0.001. Hence, the affinity of each criterion must be a normalized value between 0.001 and 1 for the overall affinity calculation to work.

The proposed initial set of criteria can be easily extended without changing the overall affinity measurement solution. For example, if a network operator wants to consider other criteria not presented in our solution, he/she can provide the necessary information in the criteria tables (*i.e.,* criterion's type, scope and affinity equation) to define a new criterion. Thus, we allow the affinity measure to be customized according to the operator's need.

B. Affinity measurement

Our affinity measure solution combines several equations into one. The affinity measure calculation between two VNFs, presented in Equation 1, is a harmonic mean of the static affinity (Equation 3) and dynamic affinity (Equation 4). However, whether or not dynamic affinity is considered in the mean depends on both VNFs being running. This behavior is represented by p (Equation 2). If both VNFs are running, p will be 1 and the dynamic affinity will be considered in the final result. If any of the two VNFs is not running, p will be 0 and the result will be the same as the static affinity. In addition, since two VNFs may be chained in more than one FG, which could imply on different values for FG criteria such as latency, our affinity measure is calculated for each FG both

Criterion	Formula
Minimum CPU	$\alpha = \begin{cases} 1 \quad if\ (cpu_{vnf_a} >= cpu_{NSD}) \wedge (cpu_{vnf_b} >= cpu_{NSD}), \\ (1 + max(0.001, {}^{cpu_{vnf_b}}/_{cpu_{NSD}})) \times 0.5 \quad if\ (cpu_{vnf_a} >= cpu_{NSD}) \wedge (cpu_{vnf_b} < cpu_{NSD}), \\ (max(0.001, {}^{cpu_{vnf_a}}/_{cpu_{NSD}}) + 1) \times 0.5 \quad if\ (cpu_{vnf_a} < cpu_{NSD}) \wedge (cpu_{vnf_b} >= cpu_{NSD}), \\ (max(0.001, {}^{cpu_{vnf_a}}/_{cpu_{NSD}}) + max(0.001, {}^{cpu_{vnf_b}}/_{cpu_{NSD}})) \times 0.5 \quad \text{otherwise.} \end{cases}$
Minimum memory	$\alpha = \begin{cases} 1 \quad if\ (mem_{vnf_a} >= mem_{NSD}) \wedge (mem_{vnf_b} >= mem_{NSD}), \\ (1 + max(0.001, {}^{mem_{vnf_b}}/_{mem_{NSD}})) \times 0.5 \quad if\ (mem_{vnf_a} >= mem_{NSD}) \wedge (mem_{vnf_b} < mem_{NSD}), \\ (max(0.001, {}^{mem_{vnf_a}}/_{mem_{NSD}}) + 1) \times 0.5 \quad if\ (mem_{vnf_a} < mem_{NSD}) \wedge (mem_{vnf_b} >= mem_{NSD}), \\ (max(0.001, {}^{mem_{vnf_a}}/_{mem_{NSD}}) + max(0.001, {}^{mem_{vnf_b}}/_{mem_{NSD}})) \times 0.5 \quad \text{otherwise.} \end{cases}$
Minimum storage	$\alpha = \begin{cases} 1 \quad if\ (sto_{vnf_a} >= sto_{NSD}) \wedge (sto_{vnf_b} >= sto_{NSD}), \\ (1 + max(0.001, {}^{sto_{vnf_b}}/_{sto_{NSD}})) \times 0.5 \quad if\ (sto_{vnf_a} >= sto_{NSD}) \wedge (sto_{vnf_b} < sto_{NSD}), \\ (max(0.001, {}^{sto_{vnf_a}}/_{sto_{NSD}}) + 1) \times 0.5 \quad if\ (sto_{vnf_a} < sto_{NSD}) \wedge (sto_{vnf_b} >= sto_{NSD}), \\ (max(0.001, {}^{sto_{vnf_a}}/_{sto_{NSD}}) + max(0.001, {}^{sto_{vnf_b}}/_{sto_{NSD}}) \times 0.5 \quad \text{otherwise.} \end{cases}$
NFV conflicts	$\alpha = \begin{cases} 1 \quad \text{if the two VNFs respect conflicts,} \\ 0.001 \quad \text{otherwise.} \end{cases}$
CPU usage	$\alpha = max(0.001, 1 - ({}^{\%cpu_{vnf_a} + \%cpu_{vnf_b}}/_{100}))$
Memory usage	$\alpha = max(0.001, 1 - ({}^{\%mem_{vnf_a} + \%mem_{vnf_b}}/_{100}))$
Storage usage	$\alpha = max(0.001, 1 - ({}^{\%sto_{vnf_a} + \%sto_{vnf_b}}/_{100}))$
Bandwidth usage	$\alpha = max(0.001, 1 - ({}^{\%bnd_{(vnf_a, vnf_b)}}/_{100}))$
Packet loss	$\alpha = max(0.001, 1 - ({}^{\%pkt\_loss_{(vnf_a, vnf_b)}}/_{100}))$
Latency	$\alpha = \begin{cases} 1 \quad if\ 2 \times lat_{(vnf_a, vnf_b)} <= lat_{SLA}, \\ max(0.001, 1 - \dfrac{2 \times lat_{(vnf_a, vnf_b)} - lat_{SLA}}{lat_{SLA}}) \quad \text{otherwise.} \end{cases}$

Table II: Criteria formulas.

VNFs belong to. If the two VNFs are not directly chained in any FG, the affinity measure will be only calculated once, taking into account solely PM criteria.

Using a harmonic mean to combine bottom-level calculations keeps the final result value high in case the bottom-level results are high, and decreases the result value as bottom-level results decrease. Also, by using a harmonic mean to combine affinities ensures that low measures are not masked by a higher measure, since any low affinity value will decrease the final result significantly. However, because of the harmonic mean behavior, it is crucial that no bottom-level calculation results on zero, since any zeros in the mean would simply result on a zero result, possibly masking any other higher values in the mean.

$$\alpha_{(vnf_a, vnf_b)} = \frac{1 + p}{\frac{1}{\alpha_s} + \frac{p}{\alpha_d}}, \quad \forall fg \in \{vnf_a \cap vnf_b\} \tag{1}$$

$$p = \begin{cases} 1 & \text{if the two VNFs are running,} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The static affinity calculation (Equation 3) is a harmonic mean of the static PM affinity and the static FG affinity. The input for this measure is the constant information from all static criteria, such as resource requirements and historical data. In this way, the static affinity measure can be used before VNFs deployment, to aid on the embedding process.

$$\alpha_s = \frac{2}{\frac{1}{\alpha_{s_{pm}}} + \frac{1}{\alpha_{s_{fg}}}} \tag{3}$$

The dynamic affinity calculation (Equation 4) is a harmonic mean of the dynamic PM affinity and the network affinity. However, whether or not PM and network affinities are taken into account depends on a couple of parameters: x (Equation 5) and y (Equation 6). If both VNFs being evaluated are hosted by the same PM, then x will be 1, and therefore, PM affinity will be considered in the harmonic mean. Likewise, if both VNFs are directly chained on the FG being evaluated, according to the NSD, y will be 1 and network affinity will be considered on the harmonic mean. If those conditions are not met, then x or y will be zero, disregarding either PM or network affinity from the equation.

$$\alpha_d = \frac{x + y}{\frac{x}{\alpha_{d_{pm}}} + \frac{y}{\alpha_{net}}} \tag{4}$$

$$x = \begin{cases} 1 & \text{if the two VNFs are hosted by the same PM,} \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

$$y = \begin{cases} 1 & \text{if the two VNFs are directly chained on the FG,} \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

The network affinity (Equation 7) is used to adjust the dynamic FG affinity. To do so, a specific parameter is used to drive the result: traffic affinity. This formula follows the following behavior: if two VNFs have a high traffic affinity, that is, there is a relatively large amount of traffic flowing between them, the dynamic FG affinity will have a large influence on the overall dynamic affinity result; if two VNFs have a low traffic affinity, the dynamic FG will not have as much influence on the overall dynamic affinity. Thus, as there is a low amount of traffic flowing through the VNFs, the dynamic FG criteria will not impact the service provided by the FG.

$$\alpha_{net} = 0.5 + ((\alpha_{trf}/_2) \times (\alpha_{d_{fg}} - (1 - \alpha_{d_{fg}}))) \tag{7}$$

Figure 1 depicts a heat map demonstrating the network affinity behavior. As the traffic affinity increases, the dynamic FG affinity determines whether the resulting network affinity will be high or low. For example, considering a fixed value of traffic affinity measure of 0.9, if the dynamic FG affinity is 0.2, the network affinity will be 0.23, whereas if the dynamic is 0.9, the network affinity will be 0.85. On the contrary, as traffic affinity decreases, such as 0.2, the dynamic FG affinity has a lower impact on the resulting value of network affinity, which tends to stay around 0.5.

The traffic affinity measure (Equation 8) is a proportion of how much traffic is passing through the virtual links between the two VNFs being evaluated. Since this value is only calculated if the VNFs are directly chained, we consider the traffic value going through a single virtual link as the traffic between the two VNFs. This value is proportioned relative to the highest single virtual link traffic rate between any two VNFs in the FG.

1.00	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00
0.90	0.14	0.23	0.32	0.41	0.50	0.59	0.68	0.77	0.86	0.95
0.80	0.18	0.26	0.34	0.42	0.50	0.58	0.66	0.74	0.82	0.90
0.70	0.22	0.29	0.36	0.43	0.50	0.57	0.64	0.71	0.78	0.85
0.60	0.26	0.32	0.38	0.44	0.50	0.56	0.62	0.68	0.74	0.80
0.50	0.30	0.35	0.40	0.45	0.50	0.55	0.60	0.65	0.70	0.75
0.40	0.34	0.38	0.42	0.46	0.50	0.54	0.58	0.62	0.66	0.70
0.30	0.38	0.41	0.44	0.47	0.50	0.53	0.56	0.59	0.62	0.65
0.20	0.42	0.44	0.46	0.48	0.50	0.52	0.54	0.56	0.58	0.60
0.10	0.46	0.47	0.48	0.49	0.50	0.51	0.52	0.53	0.54	0.55
α_{trf} / $\alpha_{d_{fg}}$	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00

Figure 1: Network affinity values by dynamic FG affinity and traffic affinity.

$$\alpha_{trf} = \frac{trf_{(vnf_a, vnf_b)}}{hgst\_trf_{fg}} \tag{8}$$

Finally, the static FG and PM affinity, as well as the dynamic FG and PM affinity, are presented in Equation 9. All four affinity measures are calculated in the same way, a weighted harmonic mean of the affinity measures of each criterion, differing only by the criteria it takes into account. Each criterion has an associated weight provided as input by the network operator. If a network operator wants to disregard any criterion, he needs to provide zero as the criterion's weight.

$$\alpha_x = \frac{\sum_{i=1}^{n_x} w_i}{\sum_{i=1}^{n_x} \frac{w_i}{\alpha C_i}}, \quad \forall x \in \{s_{pm}, s_{fg}, d_{pm}, d_{fg}\}$$
$$n_x, \quad \text{number of criteria of } x. \tag{9}$$
$$w_i \in \mathbb{N}, \quad \text{weight for criterion } i.$$
$$\alpha C_i, \quad \text{affinity measure for criterion } i.$$

IV. Case Studies

To evaluate our affinity model, we analyze a VNF as a Service [14] (VNFaaS) scenario with multiple tenants sharing the same infrastructure and VNFs. We provide case studies to demonstrate how our solution helps a network operator to identify three distinct issues: physical machine resources contention; latency on NFV-enabled networks, and VNFs dependency issues.

Figure 2 illustrates our evaluation scenario, which contains seven VNFs running over three PMs and three distinct tenants with their respective FGs. PM 1 hosts three VNFs — a load balancer, a firewall, and an Intrusion Detection System (IDS); PM 2 hosts three VNFs as well — a Deep Packet Inspection (DPI), another firewall, and an Intrusion Prevention System

(IPS); PM 3 hosts a single VNF — a packet sniffer. FG 1, contracted by an university, includes a load balancer, two firewalls and an IDS; FG 2 provides to a bank a DPI, a firewall and an IPS; and FG 3, contracted by an Information Technology (IT) company, consists of a load balancer, two firewalls, a DPI and a packet sniffer. Figure 2 also includes the resource capacities of each PM, described by: number of CPUs and clock frequency, amount of memory; and I/O operations per second (IOPS). In addition, Figure 2 includes the bandwidth capacity of physical links between PMs, PMs to the Internet, and PMs to tenants.

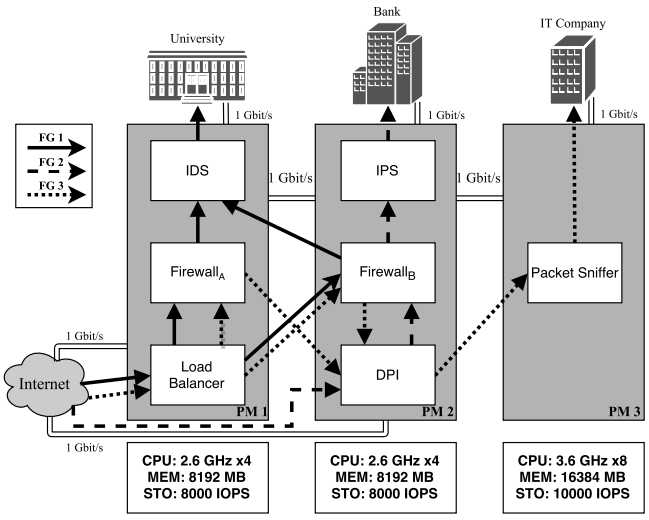

Figure 2: NFV example scenario.

Table III shows a snapshot of the current resource usage for each PM: CPU, memory and IOPS. In addition, Table III also informs how much of the usage percentage each VNF hosted on one particular PM is responsible for. For instance, in this scenario, PM 1 has a 85% of CPU Usage, from which the load balancer is responsible for 40%. Meanwhile, Table IV presents the usage data relevant for each FG, divided by each flow: traffic, bandwidth usage, packet loss, and latency. It is important to point out that FG 1 and FG 2 share a physical link, since they both have the same flow from the Internet to the load balancer. For all case studies, consider an empty conflicts lists and a 30 ms SLA for latency. Also, consider that all VNFs fulfill their resource requirements, resulting on a static affinity value of 1.

A. Case Study #1

Consider the three VNFs running on PM 1. The load balancer receives traffic from two different FGs; as a result, it consumes 40% of PM 1 CPU capacity. IDS is a CPU bound network function, being responsible for 35% of the CPU usage. The firewall only performs rules matching, thus, its consumption is low (10%) in comparison with the other two VNFs. This CPU usage behavior can lead to resource contention in PM 1, which incurs in performance degradation. Equation 10 presents the resulting affinities, given that all the criteria have the same weight 1.

$$\begin{aligned} \forall\ \alpha C_i, \quad w_i &= 1 \\ \alpha_{(LoadBalancer,\ Firewall_A)} &= 0.755 \\ \alpha_{(Firewall_A,\ IDS)} &= 0.743 \\ \alpha_{(LoadBalancer,\ IDS)} &= 0.554 \end{aligned} \tag{10}$$

The provided values show that the IDS and the load balancer have a lower affinity when compared to the affinities they have with the firewall. The model results in an affinity close to 0.75 between the firewall and the other two VNFs, and 0.55 for the relation between the load balancer and IDS. However, it is important to notice that the static affinity for this calculation was at its maximum value, which increases the total affinity.

If a network operator wants to ignore the static criteria, to produce an affinity result more focused on execution stats, he/she can provide input weights with zero value for the static criteria. Equation 11 produces the affinity results directed for the dynamic behavior in this scenario. Therefore, all static criteria receives zero as input weight.

$$\begin{aligned} \forall\ \alpha C_i \in \{s_{pm}, s_{fg}\}, \quad w_i &= 0 \\ \forall\ \alpha C_i \in \{d_{pm}, d_{fg}\}, \quad w_i &= 1 \\ \alpha_{(LoadBalancer,\ Firewall_A)} &= 0.609 \\ \alpha_{(Firewall_A,\ IDS)} &= 0.592 \\ \alpha_{(LoadBalancer,\ IDS)} &= 0.383 \end{aligned} \tag{11}$$

The results without the static criteria reduces the overall affinity among all the VNFs. However, it provides a significant result for identifying the root cause of the problem. The change in criteria incur in a conceptual shift for the relation between the load balancer and the IDS. By considering only dynamic criteria, they expose an anti-affinity behavior, which is not true for the relationships that include the firewall. In this scenario, the network operator, noticing the anti-affinity relation, could move either the load balancer or the IDS to a separate hardware, with less stress on physical resources.

B. Case Study #2

In the second case study we assess how FG dynamic criteria impact the affinity measure. For this analysis, consider the four VNFs chained on FG 3, in which VNFs are placed on different PMs. In addition, as we are focusing on FG 3, affinities from other FG that may apply to any VNF will not be analyzed in this case study. Most VNFs on FG 3 run on distinct hardware, so the dynamic PM criteria will not interfere in the affinity measurement. Thus, let us focus on the network behavior exhibited in the FG. All VNFs in FG 3 have low values for bandwidth usage and packet loss, and there is a high input traffic in the load balancer. However, as the function distributes traffic among FGs, it causes a decrease in the amount of traffic reaching both firewalls. In the case of latency, the values are below the 30 ms established in SLA for all VNFs relations, except for the one between the DPI and the packet sniffer,

PM	CPU Usage	Memory Usage	Storage Usage	VNF	VNF's CPU Usage	VNF's Memory Usage	VNF's Storage Usage
1	85%	65%	60%	Load Balancer	40%	20%	30%
				Firewall A	10%	20%	10%
				IDS	35%	25%	20%
2	90%	60%	45%	DPI	60%	30%	20%
				Firewall B	10%	20%	10%
				IPS	20%	10%	15%
3	20%	15%	20%	Packet Sniffer	20%	15%	20%

Table III: PMs resources usage of example scenario.

FG	Flow	Traffic	Bandwidth Usage	Packet Loss	Latency
1	$Internet \rightarrow Load\ Balancer$	500 Mbit/s	50%	1%	10 ms
	$Load\ Balancer \rightarrow Firewall_A$	200 Mbit/s	25%	1%	1 ms
	$Load\ Balancer \rightarrow Firewall_B$	200 Mbit/s	37%	1%	5 ms
	$Firewall_A \rightarrow IDS$	100 Mbit/s	10%	1%	1 ms
	$Firewall_B \rightarrow IDS$	100 Mbit/s	37%	1%	5 ms
	$IDS \rightarrow University$	200 Mbit/s	20%	1%	10 ms
2	$Internet \rightarrow DPI$	200 Mbit/s	20%	1%	10 ms
	$DPI \rightarrow Firewall_B$	200 Mbit/s	22%	1%	35 ms
	$Firewall_B \rightarrow IPS$	40 Mbit/s	4%	1%	1 ms
	$IPS \rightarrow Bank$	40 Mbit/s	4%	1%	10 ms
3	$Internet \rightarrow Load\ Balancer$	500 Mbit/s	50%	1%	10 ms
	$Load\ Balancer \rightarrow Firewall_A$	50 Mbit/s	25%	1%	1 ms
	$Load\ Balancer \rightarrow Firewall_B$	50 Mbit/s	37%	1%	5 ms
	$Firewall_A \rightarrow DPI$	20 Mbit/s	37%	1%	5 ms
	$Firewall_B \rightarrow DPI$	20 Mbit/s	22%	1%	1 ms
	$DPI \rightarrow Packet\ Sniffer$	40 Mbit/s	4%	1%	28 ms
	$Packet\ Sniffer \rightarrow IT\ Company$	40 Mbit/s	4%	1%	10 ms

Table IV: FGs resources usage of example scenario.

which is close to the SLA. This high value indicates that PM 3 is physically distant from PMs 1 and 2.

With these exposed values, a network operator might identify this abnormal latency as a problem. However, taking a closer look, the amount of traffic that is being transmitted in this flow is low, when compared to the highest value in the FG. Hence, even though latency is high, it compromises just a bit of the service being provided, reducing its overall impact. Equation 12 exposes the affinity measure from our model in this case.

$$\begin{aligned} fg &= 3 \\ \forall\ \alpha C_i, \quad w_i &= 1 \\ \alpha_{(LoadBalancer,\ Firewall_A)} &= 0.707 \\ \alpha_{(LoadBalancer,\ Firewall_B)} &= 0.695 \\ \alpha_{(Firewall_A,\ DPI)} &= 0.678 \\ \alpha_{(Firewall_B,\ DPI)} &= 0.646 \\ \alpha_{(DPI,\ PacketSniffer)} &= 0.653 \end{aligned} \tag{12}$$

These values reveal an affinity relationship between any pair of VNFs in the FG. This occur because the combination of small traffic and high latency reduces the network affinity in Equation 7. Thus, since the static criteria affinity equals to 1, the total affinity results in a measure larger than 0.5. In summary, the observance of theses values can prevent the operator from misplacing VNFs based only on latency observations.

C. Case Study #3

Finally, consider the VNFs chained on FG 2. Analyzing the FG criteria, it is possible to notice a high discrepancy on the latency values. The normal latency value for links between PM 1 and 2 is 5 ms, but the latency between the DPI and the firewall on FG 3 is 35 ms, higher than the 30 ms SLA. In addition, evaluating the PM criteria, it is clear that the DPI is consuming a great portion of the PM resources, stressing the CPU and memory.

Considering the amount of traffic that gets blocked by the firewall, decreasing from 2 Mbit/s to 0.4 Mbit/s, it is safe to assume that it was mistakenly chained after the DPI on FG

2, causing the identified issues. To confirm this insight, we apply our affinity measure model considering a higher weight for the suspect resources, CPU usage and latency. Equation 13 presents the resultant affinities for the VNFs on FG 2.

$$\begin{aligned} fg &= 2 \\ \forall\ \alpha C_i \in \{latency, cpu\_usage\}, \quad w_i &= 2 \\ \forall\ \alpha C_i \in \{s_{pm}, s_{fg}\}, \quad w_i &= 0 \\ \forall\ \alpha C_i \notin \{latency, cpu\_usage, s_{pm}, s_{fg}\}, \quad w_i &= 1 \\ \alpha_{(DPI,\ Firewall_B)} &= 0.003 \\ \alpha_{(Firewall_B,\ IPS)} &= 0.655 \\ \alpha_{(DPI,\ IPS)} &= 0.391 \end{aligned} \tag{13}$$

The resulting affinities show an anti-affinity relation between the DPI and the firewall, due to the high latency in the flow and the high CPU usage of the DPI. The DPI and the IPS also present a low affinity, since the IPS is a CPU-bound VNF and the DPI is consuming 60% of PM 2 CPU. Notice that the DPI and the firewall belong to both FG 2 and FG 3. For such case, we measure the affinity for each FG that the VNFs are part of. For instance, FG 2 and FG 3 have distinct values of latency, which further implies that the DPI is mistakenly chained before the firewall on FG 2. Equation 14 presents the affinity measure for the DPI and the firewall B for FG 3, considering the same weights as in Equation 13. The resulting affinity is low due to PM resource contention, but since the FG criteria are normalized for FG 3, the measure is much higher than the value presented on Equation 13.

$$\begin{aligned} fg &= 3 \\ \forall\ \alpha C_i \in \{latency, cpu\_usage\}, \quad w_i &= 2 \\ \forall\ \alpha C_i \in \{s_{pm}, s_{fg}\}, \quad w_i &= 0 \\ \forall\ \alpha C_i \notin \{latency, cpu\_usage, s_{pm}, s_{fg}\}, \quad w_i &= 1 \\ \alpha_{(DPI,\ Firewall_B)} &= 0.448 \end{aligned} \tag{14}$$

To solve the stated issues, and the results provided by the affinity model, an operator can change the dependencies in FG 2, so that the DPI is chained after the firewall. In this way, the traffic load processed by the DPI decreases, as the firewall blocks packets. By changing the dependencies in FG 3, the DPI and firewall will have the same flow as in FG 2, in which the affinity measure is higher.

V. Conclusion and Future Work

In this paper, we introduced a measure to estimate affinity between pairs of VNFs, based on a weighted set of criteria. This measure helps network operators identify issues on NFV-enabled networks. In addition, this measure can be used to aid NFV orchestrators — and network operators — on VNFs embedding and migration. In summary, we *(i)* provide an affinity relationship definition, *(ii)* define an extendable set of affinity criteria; and *(iii)* describe a mathematical model to measure the affinity between any pair of VNFs.

We analyzed our affinity measure over an NFVaaS scenario, with multiple tenants sharing infrastructure and VNFs. We provided three distinct case studies, demonstrating how our affinity measure helps the network operator identify different issues on the network. In case study #1, we use our affinity model to expose PM resource contention, derived from two resource consuming VNFs on the same PM. In case study #2, we analyze latency among VNFs, and how our affinity measure might prevent network operators from misplacing VNFs simply on latency observations. Finally, in case study #3, we use our affinity model to highlight dependency issues on the network, in which a DPI is mistakenly placed before a firewall, causing high latency and CPU consumption. All these case studies reveal that the proposed affinity measure provides insight on NFV-enabled network issues. In addition, instead of having to analyze dozens of different metrics, our model combine all those metrics into a single value that expresses how well two VNFs run together, simplifying management and scalability.

As future work, we intend to add different criteria, to improve the affinity measure, such as considering the function being provided by each VNF (*e.g.*, which rules are running on a firewall, and what kind of inspection is being provided by a IPS). Also, we plan to store historical data for each measure calculated, and use it as input to retroactively improve the affinity measure for the evaluated VNFs. Finally, this measure might be incorporated on a visualization platform, such as VISION [11], to aid network operators on having insights about the network.

Acknowledgment

This work has been partially supported by the project "GREEN-CLOUD: Computação em Cloud com Computação Sustentavel" (#16/2551-0000), from FAPERGS and CNPq Brazil, program PRONEX 12/2014.

References

[1] ETSI NFV ISG. Network Functions Virtualisation. White Paper 1, October 2012.

[2] R. Luis dos Santos, O. M. C. Rendon, J. A. Wickboldt, and L. Z. Granville. App2net: A platform to transfer and configure applications on programmable virtual networks. In *2015 IEEE Symposium on Computers and Communication (ISCC)*, pages 315–322, July 2015.

[3] Rashid Mijumbi, Joan Serrat, Juan-Luis Gorricho, Niels Bouten, Filip De Turck, and Raouf Boutaba. Network Function Virtualization: State-of-the-art and Research Challenges. *CoRR*, abs/1509.07675, 2015.

[4] S. Oechsner and A. Ripke. Flexible support of VNF placement functions in OpenStack. In *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, pages 1–6, April 2015.

[5] S. Sudevalayam and P. Kulkarni. Affinity-Aware Modeling of CPU Usage for Provisioning Virtualized Applications. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 139–146, July 2011.

[6] Alcatel-Lucent. Network Functions Virtualization - Challanges and Solutions. White paper, June 2013.

[7] J. Chen, K. Chiew, D. Ye, L. Zhu, and W. Chen. AAGA: Affinity-Aware Grouping for Allocation of Virtual Machines. In *Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on*, pages 235–242, March 2013.

[8] J. Sonnek, J. Greensky, R. Reutiman, and A. Chandra. Starling: Minimizing Communication Overhead in Virtualized Computing Platforms Using Decentralized Affinity-Aware Migration. In *2010 39th International Conference on Parallel Processing*, pages 228–237, Sept 2010.

[9] Timothy Wood, Gabriel Tarasuk-Levin, Prashant Shenoy, Peter Desnoyers, Emmanuel Cecchet, and Mark D. Corner. Memory Buddies: Exploiting Page Sharing for Smart Colocation in Virtualized Data Centers. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, VEE '09, pages 31–40, New York, NY, USA, 2009. ACM.

[10] M. Yoshida, W. Shen, T. Kawabata, K. Minato, and W. Imajuku. MORSA: A multi-objective resource scheduling algorithm for NFV infrastructure. In *Network Operations and Management Symposium (APNOMS), 2014 16th Asia-Pacific*, pages 1–6, Sept 2014.

[11] Muriel Franco, Ricardo Luis dos Santos, Alberto E. Schaeffer-Filho, and Lisandro Z Granville. VISION - Interactive and Selective Visualization for Management of NFV-Enabled Networks. In *The 30-th IEEE International Conference on Advanced Information Networking and Applications (AINA-2016)*, Centre de Congrès le Régent, Crans-Montana, Switzerland, March 2016.

[12] F. Z. Yousaf and T. Taleb. Fine-grained resource-aware virtual network function management for 5G carrier cloud. *IEEE Network*, 30(2):110–115, March 2016.

[13] Ricardo J Pfitscher, Eder J Scheid, Ricardo L dos Santos, Rafael R Obelheiro, Mauricio A Pillon, Alberto E Schaeffer-Filho, and Lisandro Z Granville. DReAM-a distributed result-aware monitor for Network Functions Virtualization. In *2016 IEEE Symposium on Computers and Communication (ISCC)*, pages 663–668. IEEE, 2016.

[14] G. Xilouris, M. A. Kourtis, M. J. McGrath, V. Riccobene, G. Petralia, E. Markakis, E. Palis, A. Georgios, G. Gardikis, J. F. Riera, A. Ramos, and J. Bonnet. T-NOVA: Network functions as-a-service over virtualised infrastructures. In *Network Function Virtualization and Software Defined Network (NFV-SDN), 2015 IEEE Conference on*, pages 13–14, Nov 2015.