

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

ARTHUR SELLE JACOBS

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

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Science

Advisor: Prof. Dr. Lisandro Granville

Porto Alegre
December 2016

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Reitor: Prof. Rui Vicente Oppermann

Vice-Reitora: Prof^a. Jane Fraga Tutikian

Pró-Reitor de Graduação: Prof. Sérgio Roberto Kieling Franco

Diretora do Instituto de Informática: Prof^a. Carla Maria Dal Sasso Freitas

Coordenador do Curso de Ciência de Computação: Prof. Carlos Arthur Lang Lisbôa

Bibliotecária-chefe do Instituto de Informática: Beatriz Regina Bastos Haro

ACKNOWLEDGEMENTS

First of all, I would like to thank my advisor, Lisandro Granville, for all the guidance he provided me, not only throughout this project, as well as through all my graduation. I would also like to thank my first advisor, from my early years of college, Oscar Caicedo, for introducing me to the academic world, and teaching me important skills to thrive both on my academic and professional life.

I wish to express my sincere gratitude to my network labs colleagues, Ricardo Santos, Ricardo Pfitscher, Muriel Franco and Éder Scheid, for all the help they gave me making this project, guiding me through all my mistakes and making sure I stayed focused and motivated. My deepest gratitude to all my university colleagues, specially Matheus Prola and Tais Bellini, for never hesitating to give me a helping hand when I needed, and for motivating me during all our college courses. I would also like to thank my work colleagues, specially Daniel Becker, for validating and providing insight on my ideas and for encouraging me to make this project.

I would like to thank my mother, Beatriz Jacobs, my grandmother, Julieta Fleck and my father, Nelson Jacobs, for putting me in the right path when I drifted of my course. I would like to thank my brother, Lourenço Jacobs, with whom I had the pleasure of sharing the best and worst moments of my university life, for supporting me and for being with me in all the tiresome nights studying or programming for college classes.

Last but not least, I would like to thank Gabriela Lumi Yamashita Rodrigues, for all the love and support she gave me through the toughest times, for organizing my life when I needed most, and for comforting me when everything seemed to go wrong. I am certain I would not have made it this far without the help of these people.

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 document, 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 developed AMNESiA, an implementation of our affinity measurement solution. We then use AMNESiA to 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 placement.

Keywords: Network Functions Virtualization. Affinity. Virtualization. Networks. 5G.

Medição de Afinidade para Redes com Suporte a NFV: Uma Abordagem baseada em Critérios

RESUMO

Network Functions Virtualization (NFV) oferece diversos benefícios para Provedores de Serviço, como mitigar o custo de equipamentos e aumentar a agilidade do negócio. Em redes com suporte a NFV, a alocação inadequada de *Virtualized Network Functions* (VNFs) pode criar gargalos, impactando negativamente no desempenho. Portanto, operadores de rede devem estabelecer regras de afinidade e anti-afinidade para evitar gargalos de rede e processamento, e assim respeitar os requisitos de *Service Level Agreement* (SLA) estabelecidos com o usuário. Regras de afinidade e anti-afinidade em NFV devem ser abrangentes e cuidadosamente elaboradas para manter o desempenho dos serviços. Operadores de rede devem considerar mais que apenas a alocação de recursos ao identificar afinidades entre VNFs. Os critérios para afinidade de VNFs variam para grafos de encaminhamento diferentes. Geolocalização, latência, perda de pacotes, e largura de banda são alguns exemplos de critérios que podem ser considerados como indicadores de gargalos em redes de alto tráfego. Neste trabalho, é proposta uma solução para medir a afinidade entre pares de VNFs, baseando-se em um conjunto de critérios de afinidade, com pesos associados, considerados relevantes por um operador de rede. Para avaliar a viabilidade do modelo de afinidade, foi desenvolvida a solução AMNESiA, uma implementação do modelo proposto de medição de afinidade. AMNESiA foi então utilizado para analisar três casos de estudo em um cenário NFV experimental. Conclui-se que o modelo de afinidade pode ajudar operadores de redes identificar causas de problemas em redes com suporte a NFV, bem como pode ser utilizado por orquestradores NFV para auxiliar na alocação e migração de VNFs.

Palavras-chave: Network Functions Virtualization, Afinidade, Virtualização, Redes, 5G.

LIST OF FIGURES

Figure 2.1 NFV FGs example scenario.....	15
Figure 2.2 NFV architecture overview.....	16
Figure 3.1 Network affinity values by dynamic FG affinity and traffic affinity.....	33
Figure 4.1 App2Net login page.....	34
Figure 4.2 AMNESiA ER model.....	36
Figure 4.3 First step of AMNESiA's Affinity Measurement NetApp.....	37
Figure 4.4 Second step of AMNESiA's Affinity Measurement NetApp.....	37
Figure 4.5 Third step of AMNESiA's Affinity Measurement NetApp.....	38
Figure 4.6 Third step of AMNESiA's Affinity Measurement NetApp.....	38
Figure 4.7 First step of AMNESiA's Affinity Report NetApp.....	39
Figure 4.8 Third step of AMNESiA's Affinity Report NetApp.....	40
Figure 4.9 AMNESiA's Criteria Manager NetApp.....	41
Figure 4.10 AMNESiA's screen to add new criterion.....	41
Figure 5.1 NFV example scenario.....	44
Figure 5.2 AMNESiA Affinity Report for FG 1, considering all criteria.....	47
Figure 5.3 AMNESiA Affinity Report for FG 1, considering only dynamic affinity.....	47
Figure 5.4 AMNESiA Affinity Report for FG 3, considering all affinity.....	48
Figure 5.5 AMNESiA Affinity Report for FG 2, with given input weights.....	49
Figure 5.6 AMNESiA affinity results for Firewall B and DPI, with given input weights.....	49

LIST OF TABLES

Table 2.1 NSD overview.....	22
Table 3.1 Overview of set of criteria.....	24
Table 5.1 PMs resources usage of example scenario.	45
Table 5.2 FGs resources usage of example scenario.....	46

LIST OF ABBREVIATIONS AND ACRONYMS

CAPEX	Capital Expenditure
CDN	Content Delivery Networks
CPU	Core Processing Unit
DHCP	Dynamic Host Configuration Protocol
DPI	Deep Packet Inspection
EPC	Evolved Packet Core
ER	Entity-Relationship
ETSI	European Telecommunications Standards Institute
FG	Forwarding Graph
IDS	Intrusion Detection System
IOPS	I/O Operations Per Second
IPS	Intrusion Prevention System
IT	Information Technology
LTE	Long Term Evolution
PM	Physical Machine
MANO	Management and Orchestration
MORSA	Multi-objective Resource Scheduling Algorithm
NF	Network Function
NS	Network Service
NSD	Network Service Descriptor
NAT	Network Address Translation
NFV	Network Function Virtualization
NFVI	Network Function Virtualization Infrastructure
OPEX	Operational Expenditure

PM	Physical Machine
PoC	Proof of Concept
RRAS	Reference Resource Affinity Score
RU	Resource Unit
SLA	Service Level Agreement
SP	Service Provider
VIM	Virtualized Infrastructure Manager
VL	Virtual Link
VLD	Virtual Link Descriptor
VM	Virtual Machine
VNF	Virtualized Network Function
VNFaaS	Virtualized Network Function as a Service
VNFD	Virtualized Network Function Descriptor
VNFFGD	Virtualized Network Function Forwarding Graph Descriptor
VNFM	Virtualized Network Functions Manager

CONTENTS

1 INTRODUCTION	11
2 BACKGROUND AND RELATED WORK	13
2.1 Forwarding Graphs	14
2.2 NFV Architecture	14
2.3 Network Service Descriptor	18
2.4 Related Work	19
3 SOLUTION	23
3.1 Criteria	23
3.1.1 Static criteria	24
3.1.2 Dynamic criteria.....	27
3.2 Affinity measurement	30
4 IMPLEMENTATION	34
4.1 AMNESiA	35
4.1.1 Affinity Measurement	35
4.1.2 Affinity Report	38
4.1.3 Criteria Manager	40
5 CASE STUDIES	43
5.1 Case Study #1	44
5.2 Case Study #2	45
5.3 Case Study #3	48
6 CONCLUSION AND FUTURE WORK	51
REFERENCES	53
APPENDIX A — AMNESIA REFERENCE	55
A.1 VNF	55
A.2 Flavor	56
A.3 Physical Machine	57
A.4 Forwarding Graph and Flow	57
A.5 Descriptor and Conflict	58
A.6 Criterion	59
APPENDIX B — ARTICLE — IM 2017	60

1 INTRODUCTION

Network Functions Virtualization (NFV) offers several benefits for Service Providers (SPs), such as mitigating equipment cost and increasing business agility [ISG 2012]. 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 [Santos et al. 2015].

In NFV-enabled networks, inadequate placement of VNFs creates bottlenecks, impacting negatively on performance [Mijumbi et al. 2015]. Network operators establish affinity and anti-affinity rules to avoid network and processing bottlenecks [Oechsner and Ripke 2015], and thus comply with Service Level Agreement (SLA) requirements of tenants. These rules help operators improve service execution and minimize resources waste [Sudevalayam and Kulkarni 2011]. Affinity relates to the ability, or inability, of different entities to perform when combined in a certain way. Hence, 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 [Alcatel-Lucent 2013]. Although affinity is a critical issue, to the best of our knowledge, efforts to determine affinity among VNFs have been scarce [Chen et al. 2013] [Sonnek et al. 2010]. 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 document, we introduce an extensible 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 measurement. Second, we specify an extensible 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. Fourth, we present a prototype implementation of the proposed affinity measurement, used to evaluate the feasibility of our model. 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 placement.

The remaining of this document is organized as follows. In Chapter 2, we present the background regarding affinity in NFV and network virtualization. In Chapter 3, we define a set of admissible affinity criteria and introduce our solution for affinity measure. In Chapter 4, we present our implementation of the described affinity measurement. In Chapter 5, we evaluate our solution by analyzing experimental scenarios. In Chapter 6, we conclude this document and present future work.

2 BACKGROUND AND RELATED WORK

In this Chapter, we introduce the the background knowledge pertinent to this document, required to fully comprehend our solution. We also present the related work regarding affinity in NFV-enabled environments, as well as in Cloud-based environments.

Traditionally, service provisioning in the telecommunications industry has relied on network operators deploying each network function as proprietary physical devices. This increasing variety of proprietary devices required an ever growing amount of space and energy, and specialized knowledge to integrate and operate these functions. In addition, hardware-based appliances have a very short life-span, as technology advances and innovation accelerates, demanding new investments to improve service quality. Moreover, the usage of physical network functions to provide services results on a slow-paced innovation cycle and longer Time-to-Market.

NFV was proposed by the European Telecommunications Standards Institute (ETSI) [ISG 2012], aiming to address the challenges derived from using physical appliances. NFV focuses on decoupling the Network Functions (NFs) from their hardware equipment, allowing these functions to be deployed on a general-purpose rack of servers, referred to as VNFs. Using VMs to host these VNFs enables network operators to quickly and dynamically create, deploy, migrate and destroy functions. Hence, coupling networking functions with virtualization technology provides telecommunications companies reduced Capital and Operational Expenditure (CAPEX and OPEX, respectively), increased flexibility, scalability and availability, and reduced Time-to-Market by decreasing the operator's cycle of innovation.

With the advent of NFV, the ETSI provided several use-cases in which NFV may be applied [ISG 2013]. Among these use cases, some have been especially target by the industry, such as the virtualization of the Evolved Packet Core (EPC). The EPC is the core of Long Term Evolution (LTE) networks, providing essential features, including subscriber tracking, mobility and session management. Virtualizing the EPC increases flexibility and scaling, allowing for the advances of 5G networks [Hawilo et al. 2014]. Nonetheless, use cases for NFV include many other promising scenarios, from virtualizing the home environment to virtualized Content Delivery Networks (CDN).

2.1 Forwarding Graphs

Considering the ever-growing complexity of communication networks, migrating NFs from hardware appliances to software does not suffice for SPs to implement Network Services (NSs). To address these issues, the ETSI introduced the VNF FG use case. An FG defines a sequence of VNFs that packets must traverse, through point-to-point links. In this way, VNF FGs are the equivalent of connecting physical NFs through network cables. Ultimately, an FG provides a logical connectivity between VNFs.

An abstract definition of an NS based on an FG includes the types of the involved VNFs, the relationships between these VNFs, as well as the connection topology they compose, and the dependencies among each of them. Since SPs might choose to maintain certain NFs as hardware appliances, VNF FGs also have to allow interconnections between virtualized and physical appliances.

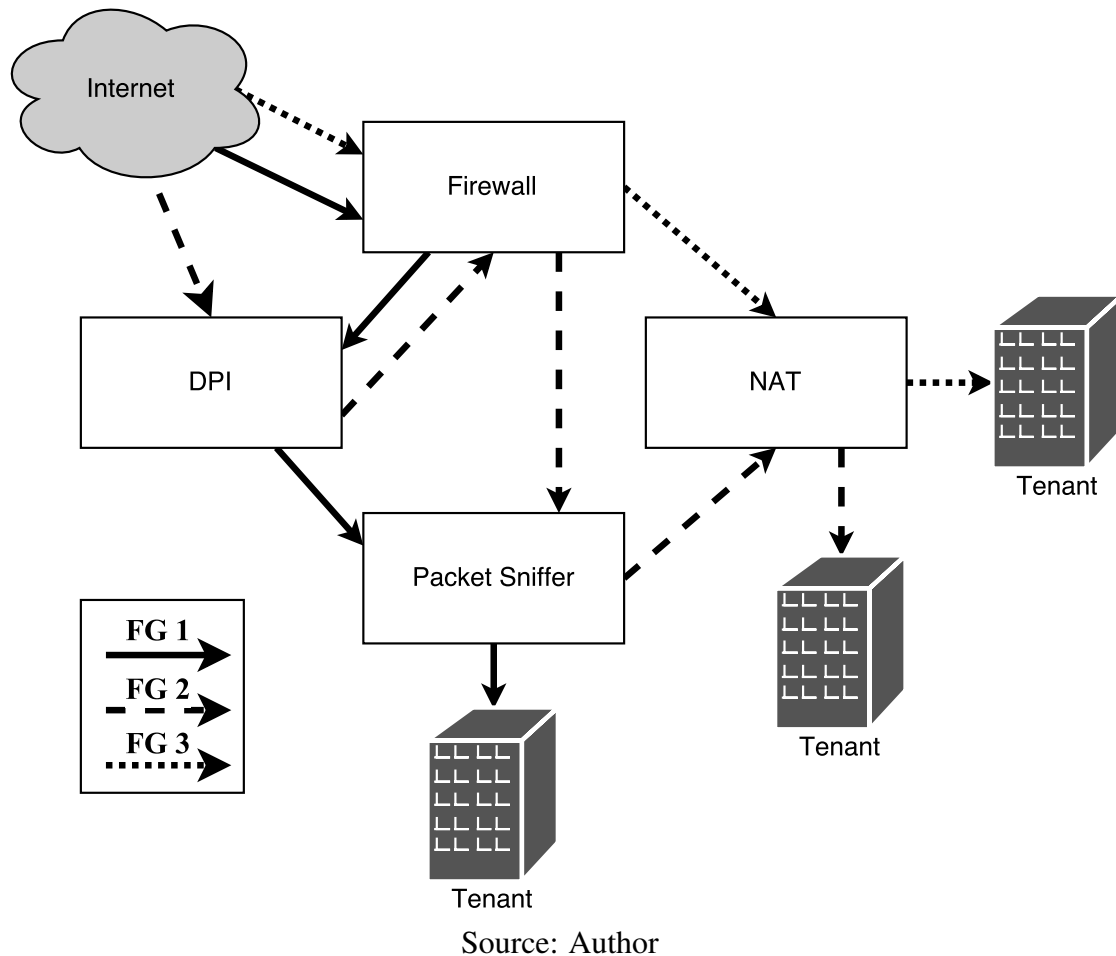
A VNF FG provides several benefits when compared to hardware-based NF FG, including efficiency, flexibility and deployability. A better efficiency is derived from having functions and network capacity sized to the current load, instead of having a dedicated hardware and network capacity sized to handle peak loads. A VNF FG provides augmented flexibility due to shorter deployment intervals for upgrades and new features, since functions are software-based. As for deployability, improvement is achieved by taking advantage of virtualization, accounting that it reduces configuration complexity and eliminates the need of physical boxes and interfaces to connect end-users.

Figure 2.1 provides an example scenario of multiple VNFs chained in different FGs. The example scenario depicts four different VNFs — a Deep Packet Inspection (DPI), a Network Address Translation (NAT) server, a packet sniffer and a firewall — being shared by three distinct tenants. Notice that the same VNFs can be chained in more than one FG, providing different NSs for different tenants.

2.2 NFV Architecture

The intrinsic advantages and innovation granted by the NFV concept also introduce a multitude of challenges and issues regarding management and infrastructure. Since NFs are virtualized in NFV, they need to be deployed on an agnostic NFV Infrastructure (NFVI), with carrier-grade servers, data centers and network, alongside virtualization enablers, such as hypervisors and VMs to host VNFs. As these entities are exposed by the

Figure 2.1: NFV FGs example scenario.



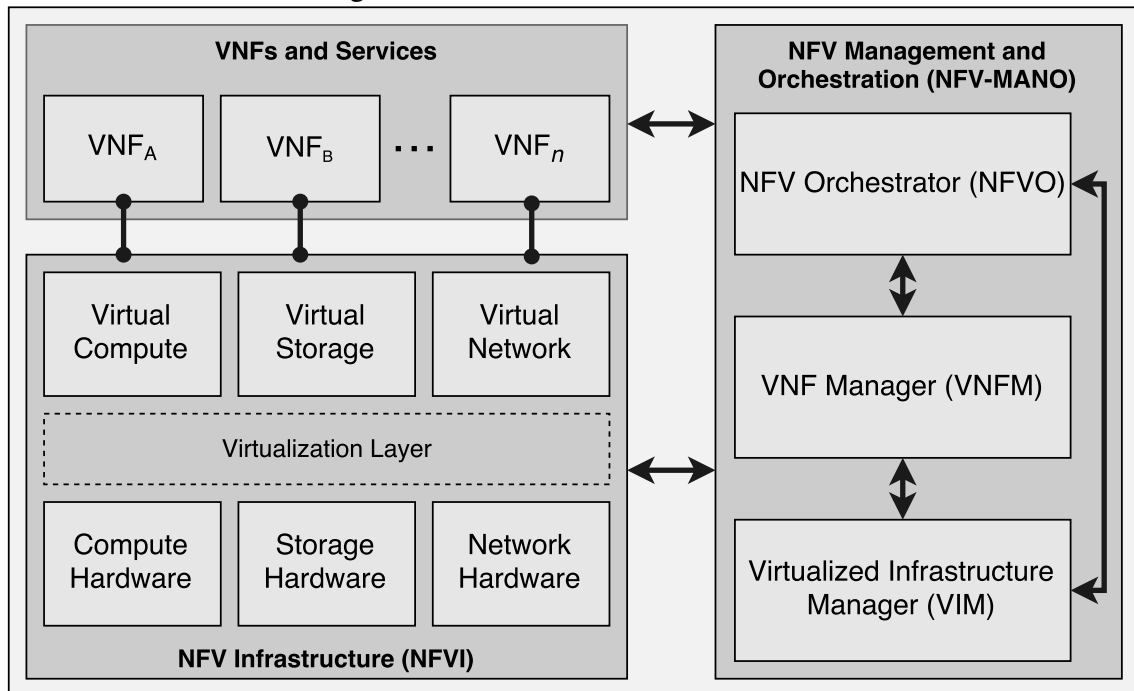
advent of NFV, the new set of relationships between those entities requires new management solutions. To address these challenges, the ETSI proposed the Management and Orchestration architectural framework (MANO) [ISG 2014], which aims to manage the NFVI and orchestrate the resource allocation for VNFs and NSs.

The overall architecture proposed by the ETSI can be seen in Figure 2.2. The NFVI corresponds to the underlying compute, storage and network infrastructure of the architecture. The VNFs, and services provided by FGs, are deployed on top of the NFVI. Finally, both the NFVI resource provisioning and orchestration of VNFs and FGs are performed by the NFV-MANO. Below, we describe in detail each different aspect of the architecture.

NFV Infrastructure

The NFVI includes all the necessary infrastructure, physical and virtual, needed for a VNF to be deployed, managed and executed [ISG 2013]. As shown in Figure 2.2,

Figure 2.2: NFV architecture overview.



Source: Author

the NFVI requires generic hardware resources, such as compute, storage and networking. On top of the physical resources, there is a virtualization layer, typically composed of hypervisors. With the addition of the virtualization, the NFVI then requires virtual resources (*i.e.*, compute, storage and networking) in which the VNFs shall be deployed. The computing and storage virtual resources may be represented as one or more VMs, while the network virtual resource are composed of Virtual Links (VLs) and nodes. A virtual node is a software component that provides either hosting or routing capabilities, while a VL represents a logical connection between virtual nodes.

The NFVI may also include partially virtualized NFs. In this scenario, an specific part or feature of the NF is virtualized, and is managed by the NFV-MANO, while other parts are comprised of physical components, due to physical constraints, for example. The management of physical appliances is not covered by the NFV-MANO. However, to ensure operational transparency, the operation of the VNF should be independent of its deployment scenario.

VNFs and Services

According to the ETSI, an NF is a functional block within a network infrastructure that has well-defined external interfaces and well-defined functional behavior

[ISG 2014]. Some common examples of NFs are Dynamic Host Configuration Protocol (DHCP) servers, firewalls and NAT servers, often used in home environments. In turn, a VNF is an NF that can be deployed on the NFVI [ISG 2014]. A VNF may consist of multiple components, and therefore may need to be deployed over multiple VMs, in which each VM hosts one single component of the VNF. However, the entire VNF may also be deployed on a single VM as well [ISG 2013].

A service is an offering composed of NFs, and defined by its functional and behavioral specification. On the NFV context, these NFs that compose a service are virtualized and deployed on the NFVI. Nonetheless, this virtualization abstraction should be transparent to users and tenants, having no performance shortages when compared to hardware-based services. The number and ordering of VNFs that constitute a service is determined by the corresponding FG of the service.

NFV Management and Orchestration

The NFV-MANO is responsible for providing end-to-end service to end-to-end NFV network mapping, instantiating and provisioning VNFs at appropriate locations to realize intended services, among other tasks. In addition, the NFV-MANO defines interfaces for the components to communicate among themselves, as well as with traditional management systems, as to allow for management of both VNFs and legacy equipment. Aiming to address such issues, the NFV-MANO was architected with three distinct functional blocks: the NFV Orchestrator (NFVO), the VNF Manager (VNFM); and the Virtualized Infrastructure Manager (VIM). It is important to point out that an NFV environment might have multiple instances of the same functional block, leveraging resources on the same infrastructure. Below, we describe in detail the responsibilities of each component.

NFVO The NFVO functional block has two main responsibilities: orchestration of NFVI resources across multiple VIMs, and the lifecycle management of NSs. To accomplish the former, the NFVO must be able to handle both virtualized and partially virtualized NFs, allocate and release resources in the NFVI dynamically; as well as support the management of the relationships between VNFs and NFVI, according to the information received from VIMs, among other capabilities. To fulfill the latter, the NFVO ought to be able to instantiate NSs and manage their lifecycles (*e.g.*, update, query, scaling, collecting measurements, termination, etc.), in coordination with VNFMs, manage the FGs of each NS; and validate and authorize NFVI

resource requests from VNFMs, preventing performance impacts on NSs, amidst other functions.

VNFM The VNFM is responsible for the lifecycle management of VNF instances. One VNFM is responsible for many VNFs, and therefore, each VNF is assumed to have an associated VNFM. For a VNFM to operate, it must be able to instantiate and configure VNFs, according to a VNF deployment template, upgrade/update VNF instances, modify VNFs, scale up or down VNF instances; and terminate VNFs, among other features. In addition, a VNFM has access to a repository of available VNFs and different versions of them, which it uses for lifecycle management. This repository may be maintained by the NFVO or an external entity.

VIM The VIM functional block is in charge of controlling the NFVI resources within one infrastructure domain. A VIM might be specialized on managing on single type of resource (*i.e.*, compute, network, storage) or able to manage multiple resources. The capabilities of a VIM include supporting the management of VNF FGs, by creating VLs and networks, for example, orchestrating the allocation, upgrading, release and reclamation of NFVI resources; and provisioning virtualized resources on physical hardware resources.

2.3 Network Service Descriptor

An NS describes the relationships between VNFs, and possible physical NFs, links needed to connect them, and the resources required to fulfill an FG. To provide all this information for the NFV-MANO to operate, a Network Service Descriptor (NSD) was introduced by the ETSI [ISG 2014]. The NSD is structured as a tree of information elements. An information element may contain a single value, in which case the element is a leaf; it may contain a reference to another information element; or it may contain a sub-element, which is an information element itself that specifies another level in the tree. Each information element has a unique name among the branches of the tree.

Each information element clause contains a table with four columns: (*i*) the name of the element, (*ii*) the type of the element, (*iii*) the cardinality of the element, which may be a positive integer or a range; and (*iv*) the content description of the information element. If the element is a sub-element, its content can be found on a subsequent clause.

An NSD is a top level information element, used as a deployment template, that

references all other descriptors that compose an NS, such as the VNF Descriptor (VNFD), the VNF FG Descriptor (VNFFGD) and the VL Descriptor (VLD). The VNFD is a deployment template describing the operational and instantiation requirements of a VNF. A VNFD is used primarily by the VNFM to instantiate and manage VNFs. A VNFFGD describes the topology of the NS, referencing VNFs and physical NFs, as well as VLs connecting them. Finally, a VLD constitutes of a deployment template describing the requirements of VLs between NFs, such as minimum bandwidth. Table 2.1 contains the an overview of the NSD base element. Further information on other descriptors may be checked on [ISG 2014].

The NFVO stores all descriptors into catalogues, which can be accessed anytime by each functional block of the NFV-MANO. Multiple versions of a descriptor may exist in a catalogue, so the NFVO may be required to keep the network configuration updated to match the latest descriptors.

2.4 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 [Sudevalayam and Kulkarni 2011], bandwidth [Sonnek et al. 2010], or even memory page sharing [Wood et al. 2009]. Next, we discuss the relevant affinity related solutions for both Cloud and NFV environments.

Chen *et al.* [Chen et al. 2013] 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 hardware 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.

Oechsner and Ripke [Oechsner and Ripke 2015] propose a flexible placement mechanism for VNFs based on an adapted zone concept, using OpenStack. This solution focuses on ensuring a high availability and performance without unnecessary operator actions. Although this work does not strictly propose a way to identify affinity relations, they aim to minimize delays and maximize availability by using a zone tree mechanism

to properly place VNFs across multiple data centers. Further, that paper does not account for the services provided.

Yoshida *et al.* [Yoshida et al. 2014] 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.* [Franco et al. 2016] 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 [Yousaf and Taleb 2016] propose a 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 behavioral patterns for each VNF, which can be used by the NFV-MANO [Yousaf and Taleb 2016] 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 au-

thors, 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 Chapter, we present our affinity measurement solution, which tackles these issues.

Table 2.1: NSD overview.

Identifier	Type	Cardinality	Description
<i>Id</i>	Leaf	1	Identification of this NSD.
<i>vendor</i>	Leaf	1	Provider or vendor of the NS.
<i>version</i>	Leaf	1	Version of this NSD.
<i>vnfd</i>	Reference	1...N	VNF constituent of this NS.
<i>vnffgd</i>	Reference	0...N	VNF FG constituent of this NS.
<i>vld</i>	Reference	0...N	VL constituent of this NS.
<i>lifecycle_event</i>	Leaf	0...N	Defines NS functional scripts/workflows for lifecycle events.
<i>vnf_dependency</i>	Leaf	0...N	Describes dependencies between VNFs, in terms of source and target. If a target VNF depends on a source VNF, the source VNF shall be deployed and connected to the NS before the target VNF is instantiated.
<i>monitoring_parameter</i>	Leaf	0...N	Represents a monitoring parameter, such as call-per-second, to monitored for this NS.
<i>service_deployment_flavour</i>	Element	1...N	Represents the parameters and its requirements for each deployment flavours of this NS.
<i>auto_scale_policy</i>	Leaf	0...N	Represents the policy meta data, which may include the criteria parameter and action-type.
<i>connection_point</i>	Element	1...N	This element describes a connection point which acts as an endpoint of this NS.
<i>pnfd</i>	Reference	0...N	Physical NFs constituent of this NS.
<i>nsd_security</i>	Leaf	0...1	This is a signature of NSD to prevent tampering. The hash algorithm used to compute this signature, and corresponding cryptographic certificate to validate the signature should be included.

3 SOLUTION

In this Chapter, 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 solve VNF placement problems.

The proposed solution receives as input a list of weights for each affinity criterion (see Section 3.1) 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 VNFs 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 [Pfischer et al. 2016]. It is important to distinguish 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.

3.1 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 3.1: Overview of set of criteria.

Type	Scope	Criterion
Static	PM	Minimum CPU requirements
		Minimum memory requirements
		Minimum storage requirements
	FG	NFV conflicts
Dynamic	PM	CPU usage
		Memory usage
		Storage usage
	FG	Bandwidth usage
		Packet loss
		Latency

Table 3.1 presents an overview of the set of criteria considered by our solution, alongside the type and scope of each criterion. Below, we give a detailed description of each criterion, coupled with its affinity calculation equation. All affinity measures in Subsections below 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.

3.1.1 Static criteria

This Subsection presents all static criteria considered by our solution. The static type relates to all criteria which data can be fetched when the VNFs are not running. The affinity calculation for these criteria normally fetches information from static sources, such as the NSD or previously stored database info.

Physical Machine Criteria

Below, we introduce the static PM criteria considered in our solution. This scope relates to criteria that consider static PM resources, such as memory and CPU requirements. These criteria will only be calculated when the two VNFs being evaluated are hosted by the same PM.

Minimum CPU requirements This criterion is declared on the NSD, in MHz, and should be used when instantiating VNFs for an NS. 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 will have a lower affinity. Equation 3.1 presents the affinity calculation for this criterion.

$$\alpha = \begin{cases} 1 & \text{if } (cpu_{vnfa} \geq cpu_{NSD}) \wedge (cpu_{vnfb} \geq cpu_{NSD}), \\ (1 + \max(0.001, \frac{cpu_{vnfb}}{cpu_{NSD}})) \times 0.5 & \text{if } (cpu_{vnfa} \geq cpu_{NSD}) \wedge (cpu_{vnfb} < cpu_{NSD}), \\ (\max(0.001, \frac{cpu_{vnfa}}{cpu_{NSD}}) + 1) \times 0.5 & \text{if } (cpu_{vnfa} < cpu_{NSD}) \wedge (cpu_{vnfb} \geq cpu_{NSD}), \\ (\max(0.001, \frac{cpu_{vnfa}}{cpu_{NSD}}) + \max(0.001, \frac{cpu_{vnfb}}{cpu_{NSD}})) \times 0.5 & \text{otherwise.} \end{cases} \quad (3.1)$$

Where:

cpu_{vnfa} is the available CPU, in MHz, for VNF_a .

cpu_{vnfb} is the available CPU, in MHz, for VNF_b .

cpu_{NSD} is the CPU requirements, in MHz, specified on the NSD for each VNF.

Minimum memory requirements This criterion is also declared on the NSD, in MB, and should be used when instantiating VNFs for an NS. If these requirements are not met for the VNF being evaluated, these VNFs will have a lower affinity. Equation

3.2 presents the affinity calculation for this criterion.

$$\alpha = \begin{cases} 1 & \text{if } (mem_{vnfa} \geq mem_{NSD}) \wedge (mem_{vnfb} \geq mem_{NSD}), \\ (1 + \max(0.001, \frac{mem_{vnfb}}{mem_{NSD}})) \times 0.5 & \text{if } (mem_{vnfa} \geq mem_{NSD}) \wedge (mem_{vnfb} < mem_{NSD}), \\ (\max(0.001, \frac{mem_{vnfa}}{mem_{NSD}}) + 1) \times 0.5 & \text{if } (mem_{vnfa} < mem_{NSD}) \wedge (mem_{vnfb} \geq mem_{NSD}), \\ (\max(0.001, \frac{mem_{vnfa}}{mem_{NSD}}) + \max(0.001, \frac{mem_{vnfb}}{mem_{NSD}})) \times 0.5 & \text{otherwise.} \end{cases} \quad (3.2)$$

Where:

mem_{vnfa} is the available memory, in MB, for VNF_a .

mem_{vnfb} is the available memory, in MB, for VNF_b .

mem_{NSD} is the memory requirements, in MB, specified on the NSD for each VNF.

Minimum storage requirements This criterion is declared on the NSD, in IOPS, and should be used when instantiating VNFs for an NS. If these requirements are not met for the VNF being evaluated, these VNFs will have a lower affinity. Equation 3.3 presents the affinity calculation for this criterion.

$$\alpha = \begin{cases} 1 & \text{if } (sto_{vnfa} \geq sto_{NSD}) \wedge (sto_{vnfb} \geq sto_{NSD}), \\ (1 + \max(0.001, \frac{sto_{vnfb}}{sto_{NSD}})) \times 0.5 & \text{if } (sto_{vnfa} \geq sto_{NSD}) \wedge (sto_{vnfb} < sto_{NSD}), \\ (\max(0.001, \frac{sto_{vnfa}}{sto_{NSD}}) + 1) \times 0.5 & \text{if } (sto_{vnfa} < sto_{NSD}) \wedge (sto_{vnfb} \geq sto_{NSD}), \\ (\max(0.001, \frac{sto_{vnfa}}{sto_{NSD}}) + \max(0.001, \frac{sto_{vnfb}}{sto_{NSD}})) \times 0.5 & \text{otherwise.} \end{cases} \quad (3.3)$$

Where:

sto_{vnfa} is the available storage, in IOPS, for VNF_a .

sto_{vnf_b} is the available storage in IOPS, for VNF_b .

sto_{NSD} is the storage requirements, in IOPS, specified on the NSD for each VNF.

Forwarding Graph Criteria

Below, we introduce the static FG criteria considered in our solution. This scope relates to criteria that consider static FG resources, such as known VNFs conflicts or policies. These criteria will only be calculated when the two VNFs being evaluated are chained in the same 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. This criterion's calculation will return 1 if the conflicts are respected and 0.001 if not. For example, if a known conflict between a DPI and a Firewall exists, stating a DPI should not be chained before a Firewall, and an operator mistakenly performs that chaining, this criterion will yield 0.001 as result. Equation 3.4 presents the affinity calculation for this criterion.

$$\alpha = \begin{cases} 1 & \text{if the two VNFs respect conflicts,} \\ 0.001 & \text{otherwise.} \end{cases} \quad (3.4)$$

3.1.2 Dynamic criteria

This Subsection presents all dynamic criteria considered by our solution. The dynamic type relates to all criteria which data can only be fetched when the VNFs are deployed and running. The affinity calculation for these criteria fetches information from monitoring solutions running over the environment, such as memory consumption or latency between VNFs.

Physical Machine Criteria

In this subsection, we introduce the dynamic PM criteria considered in our solution. This scope relates to criteria that consider dynamic PM resources, such as memory and CPU consumption. These criteria shall only be calculated when the two VNFs being

evaluated are hosted by the same PM.

CPU usage This criterion 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 the CPU. If two VNFs were responsible for a large percentage of the PM CPU usage, then these VNFs would have a low affinity. Equation 3.5 presents the affinity calculation for this criterion.

$$\alpha = \max(0.001, 1 - (\frac{\%cpu_{vnf_a} + \%cpu_{vnf_b}}{100})) \quad (3.5)$$

Where:

$\%cpu_{vnf_a}$ is the PM CPU usage percentage VNF_a is responsible for.

$\%cpu_{vnf_b}$ is the PM CPU usage percentage VNF_b is responsible for.

Memory usage Just as CPU usage, this criterion 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. Equation 3.6 presents the affinity calculation for this criterion.

$$\alpha = \max(0.001, 1 - (\frac{\%mem_{vnf_a} + \%mem_{vnf_b}}{100})) \quad (3.6)$$

Where:

$\%mem_{vnf_a}$ is the PM memory usage percentage VNF_a is responsible for.

$\%mem_{vnf_b}$ is the PM memory usage percentage VNF_b is responsible for.

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. Equation 3.7 presents the affinity calculation for this criterion.

$$\alpha = \max(0.001, 1 - (\frac{\%sto_{vnf_a} + \%sto_{vnf_b}}{100})) \quad (3.7)$$

Where:

$\%mem_{vnf_a}$ is the PM storage usage percentage VNF_a is responsible for.

$\%mem_{vnf_b}$ is the PM storage usage percentage VNF_b is responsible for.

Forwarding Graph Criteria

Below, we introduce the dynamic FG criteria considered in our solution. This scope relates to criteria that consider dynamic PM resources, such as latency and bandwidth consumption. These criteria shall only be calculated when the two VNFs being evaluated are chained in the same FG.

Bandwidth usage This criterion 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. Equation 3.8 presents the affinity calculation for this criterion.

$$\alpha = \max(0.001, 1 - (\frac{\%bnd_{(vnf_a, vnf_b)}}{100})) \quad (3.8)$$

Where:

$\%bnd_{(vnf_a, vnf_b)}$ is the bandwidth usage percentage of the link between VNF_a and VNF_b .

Packet loss This criterion, 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. Equation 3.9 presents the affinity calculation for this criterion.

$$\alpha = \max(0.001, 1 - (\frac{\%pkt_loss_{(vnf_a, vnf_b)}}{100})) \quad (3.9)$$

Where:

$\%pkt_loss_{(vnf_a, vnf_b)}$ is the packet loss percentage of the link between VNF_a and VNF_b .

Latency This criterion 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. Equation 3.10 presents the affinity calculation for this criterion.

$$\alpha = \begin{cases} 1 & \text{if } 2 \times \text{lat}_{(vnf_a, vnf_b)} \leq \text{lat}_{SLA}, \\ \max(0.001, 1 - \frac{2 \times \text{lat}_{(vnf_a, vnf_b)} - \text{lat}_{SLA}}{\text{lat}_{SLA}}) & \text{otherwise.} \end{cases} \quad (3.10)$$

Where:

$\text{lat}_{(vnf_a, vnf_b)}$ is the latency, in ms, of the link between VNF_a and VNF_b .

lat_{SLA} is the latency, in ms, specified as SLA in the NSD.

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 Section (*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.

3.2 Affinity measurement

Our affinity measurement solution combines several equations into one. The affinity measure calculation between two VNFs, presented in Equation 3.11, is a harmonic mean of the static affinity (Equation 3.13) and dynamic affinity (Equation 3.14). However, whether or not dynamic affinity is considered in the mean depends on both VNFs being running. This behavior is represented by p (Equation 3.12). 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 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 yield a zero result, possibly masking any other higher values in the mean.

$$\alpha_{(vnf_a, vnf_b)} = \frac{1+p}{\frac{1}{\alpha_s} + \frac{p}{\alpha_d}}, \quad \forall fg \in \{vnf_a \cap vnf_b\} \quad (3.11)$$

$$p = \begin{cases} 1 & \text{if the two VNFs are running,} \\ 0 & \text{otherwise.} \end{cases} \quad (3.12)$$

The static affinity calculation (Equation 3.13) 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.

$$\alpha_s = \frac{2}{\frac{1}{\alpha_{s_{pm}}} + \frac{1}{\alpha_{s_{fg}}}} \quad (3.13)$$

The dynamic affinity calculation (Equation 3.14) 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 3.15) and y (Equation 3.16). 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.

$$\alpha_d = \frac{x+y}{\frac{x}{\alpha_{d_{pm}}} + \frac{y}{\alpha_{net}}} \quad (3.14)$$

$$x = \begin{cases} 1 & \text{if the two VNFs are hosted by the same PM,} \\ 0 & \text{otherwise.} \end{cases} \quad (3.15)$$

$$y = \begin{cases} 1 & \text{if the two VNFs are directly chained on the FG,} \\ 0 & \text{otherwise.} \end{cases} \quad (3.16)$$

The network affinity (Equation 3.17) 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.

$$\alpha_{net} = 0.5 + ((\alpha_{trf} / 2) \times (\alpha_{d_{fg}} - (1 - \alpha_{d_{fg}}))) \quad (3.17)$$

Figure 3.1 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. 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 3.18) 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.

$$\alpha_{trf} = \frac{trf_{(vnfa, vnfb)}}{hgst_trf_{fg}} \quad (3.18)$$

Finally, the static FG and PM affinity, as well as the dynamic FG and PM affinity, are presented in Equation 3.19. 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/she needs

Figure 3.1: Network affinity values by dynamic FG affinity and traffic affinity.

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
$\alpha_{trf} \backslash \alpha_{dfg}$	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	1.00

Source: Author

to provide zero as the criterion's weight.

$$\alpha_x = \frac{\sum_{i=1}^{n_x} w_i}{\sum_{i=1}^{n_x} \frac{w_i}{\alpha C_i}}, \quad \forall x \in \{s_{pm}, s_{fg}, d_{pm}, d_{fg}\} \quad (3.19)$$

n_x , number of criteria of x .

$w_i \in \mathbb{N}$, weight for criterion i .

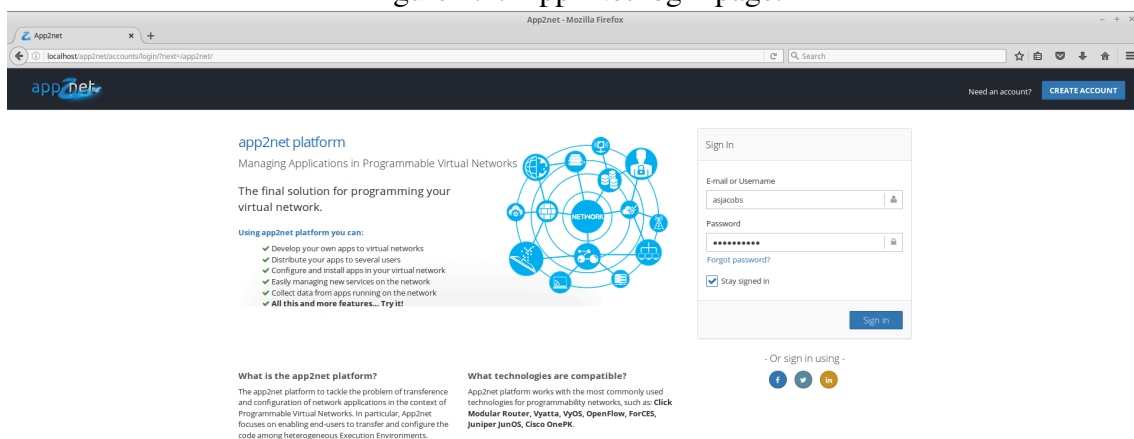
αC_i , affinity measure for criterion i .

4 IMPLEMENTATION

In this Chapter, we present AMNESiA, a Proof of Concept (PoC) implementation of the proposed affinity measurement, demonstrating a scenario in which our affinity model may be used with an operators' input. Nevertheless, as aforementioned, our affinity solution may be used without user intervention — coupled with an orchestrator, for example. We developed the affinity measurement solution as an affinity branch under App2Net [Santos et al. 2015], aiming to provide existing contexts and network environments to enhance the application.

App2Net is a platform to enable transferring and configuring Network Applications (NetApps) in Programmable Virtual Networks (PVNs) that use heterogeneous execution environments. A PVN is the denomination given to networks that support virtualization and programmability, such as NFV-enabled networks [Feamster, Rexford and Zegura 2014]. App2Net has several branches, such as visualization, policy and monitoring. Each branch has several associated NetApps. Hence, we developed our affinity solution as an App2Net branch, with three distinct NetApps, providing us with existing environment set-up to quickly deploy and install our application on NFV-environments. In addition, App2Net's existing system infrastructure already supported authentication and permission management, allowing us to focus on developing solely the proposed affinity measurement. Figure 4.1 displays the login page for App2Net.

Figure 4.1: App2Net login page.



4.1 AMNESiA

As we developed AMNESiA as an App2Net affinity branch, we had to use the same technologies as App2Net. Being so, we implemented the affinity solution as a Web application, using Python 2.7.6, the Django Web Framework 1.8.2 to provide back-end features, such as provisioning of views and database access, and HTML, Javascript and CSS for front-end capabilities. We also used a PostgreSQL Database 9.1.6 as storage. The prototype implementation was hosted and executed on a computer with four Intel Core i7-4217HQ CPUs with 2.30GHz clock speed, 8GB of RAM, and 200GB of storage, running Linux Mint 17.2 Cinnamon 64-bit.

In view of AMNESiA's as a affinity measurement solution only, we assume the data for VNFs, FGs and PMs have already been collected by any App2Net network monitoring solution, having stored such data on App2Net's database. Figure 4.2 presents the Entity-Relationship (ER) model that AMNESiA uses to retrieve data from an NFV-enabled network, as well as store the set of affinity criteria. It is important to point out that, since AMNESiA is developed in Python, the affinity equation for each criterion is also a Python function. However, we do not store the entire affinity equation function in the database, only the function's name. The body of the affinity function is stored on a specific Python file, which is imported by the platform as a module when measuring affinity.

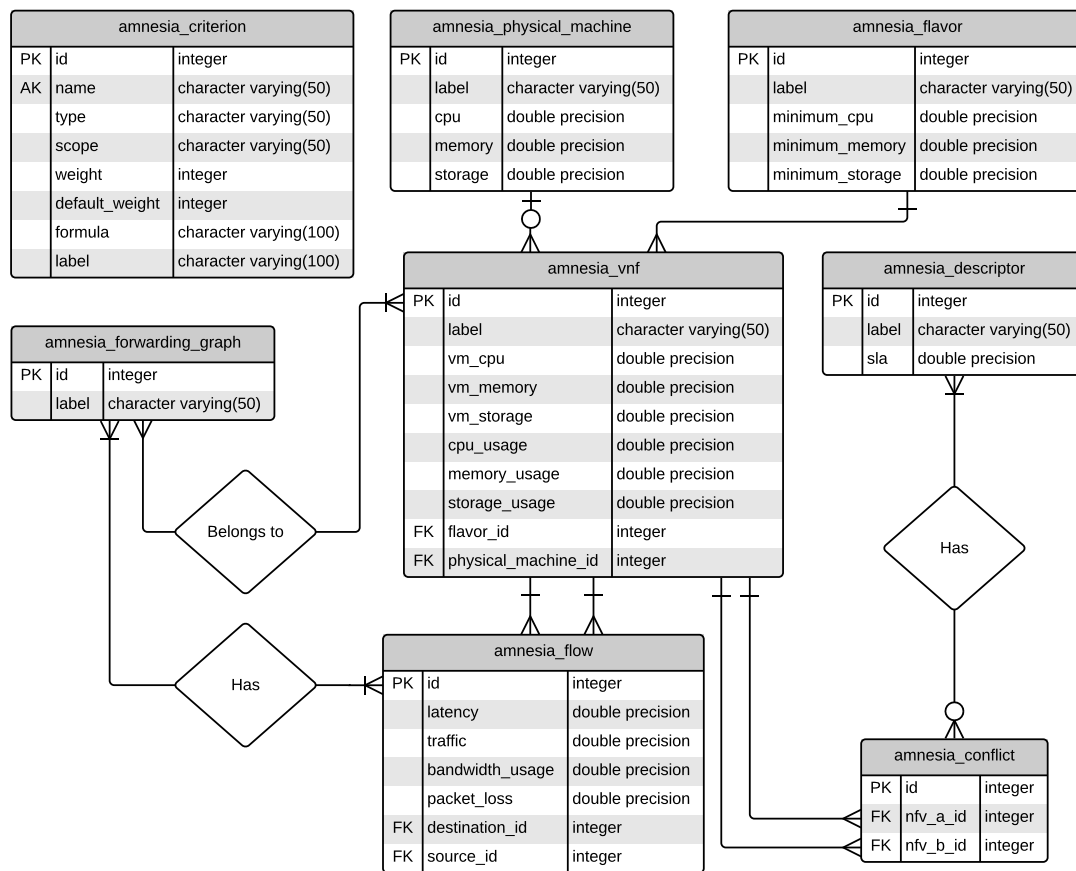
AMNESiA consists of three different NetApps: *(i)* Affinity Measurement; *(ii)* Affinity Report, and *(iii)* Criteria Manager. Each NetApp of the affinity measurement prototype is describe below, alongside execution instructions.

4.1.1 Affinity Measurement

The Affinity Measurement NetApp is a simple three-steps wizard to measure affinity between any two selected VNFs. On step one, the user — a network operator in this case — has to select two VNFs from a table list. We display on that table all the information stored in the database regarding each VNF: name; CPU, memory and storage specifications of the VM hosting the VNF; PM on which the VNF is hosted on; CPU, memory and storage usage percentage the VNF is responsible from the PM; flavor; and a list of FGs the VNF belong to.

Figure 4.3 presents the AMNESiA screen of the first step from the Affinity Mea-

Figure 4.2: AMNESiA ER model.



Source: Author

surement NetApp. It is important to point out that we only display minimal information about flavor, PM and FGS of the VNF, as we assume that flavor, PM and FG management are responsibilities of other NetApps inside App2Net. It is important to mention that the affinity measurement only works if exactly two VNFs are selected.

After selecting the VNFs on step one, the user is taken to step two: providing weights to each criterion. A network operator might give whatever weight he sees fit for each criterion. However, the weight of a criterion must be an integer larger or equal to zero. If an user wants to disregard any criterion, he/she has to provide a weight value of zero to that criterion. Just as VNFs on step one, we display the criteria on a table, coupled with the following criterion information: type, scope, label, affinity equation name; and a weight input column. We only display the affinity function name for the criterion, as that is the only information regarding the function we store on the database. Figure 4.4 presents the AMNESiA screen of the second step from the Affinity Measurement menu item.

Step three of the Affinity Measurement NetApp is simply a display of the affinity

Figure 4.3: First step of AMNESiA's Affinity Measurement NetApp.

The screenshot shows the 'Affinity Measurement' application in a browser. The main content area is titled 'Step 1 - Select two VNFs'. It features a progress indicator with three steps: 'Select VNFs' (1), 'Select criteria' (2), and 'Affinity results' (3). Below the progress indicator is a table of VNFs with the following columns: VNF, VM CPU, VM Memory, VM Storage, CPU Usage, Memory Usage, Storage Usage, Flavor, Physical Machine, and Forwarding Graphs.

VNF	VM CPU	VM Memory	VM Storage	CPU Usage	Memory Usage	Storage Usage	Flavor	Physical Machine	Forwarding Graphs
<input checked="" type="checkbox"/> Load Balancer	1000.0 MHz	1024.0 MB	2000.0 IOPS	40.0%	20.0%	30.0%	silver	host_1	[university_service][it_company_service]
<input checked="" type="checkbox"/> Firewall A	1000.0 MHz	1024.0 MB	2000.0 IOPS	10.0%	20.0%	10.0%	silver	host_1	[university_service][it_company_service]
<input type="checkbox"/> IDS	1000.0 MHz	1024.0 MB	2000.0 IOPS	35.0%	25.0%	20.0%	silver	host_1	[university_service]
<input type="checkbox"/> DPI	1000.0 MHz	1024.0 MB	2000.0 IOPS	60.0%	30.0%	20.0%	silver	host_2	[bank_service][it_company_service]
<input type="checkbox"/> Firewall B	1000.0 MHz	1024.0 MB	2000.0 IOPS	10.0%	20.0%	10.0%	silver	host_2	[university_service][bank_service][it_company_service]
<input type="checkbox"/> IPS	1000.0 MHz	1024.0 MB	2000.0 IOPS	20.0%	10.0%	15.0%	silver	host_2	[bank_service]
<input type="checkbox"/> Packet Sniffer	1000.0 MHz	1024.0 MB	2000.0 IOPS	20.0%	15.0%	20.0%	silver	host_3	[it_company_service]

Source: Author

Figure 4.4: Second step of AMNESiA's Affinity Measurement NetApp.

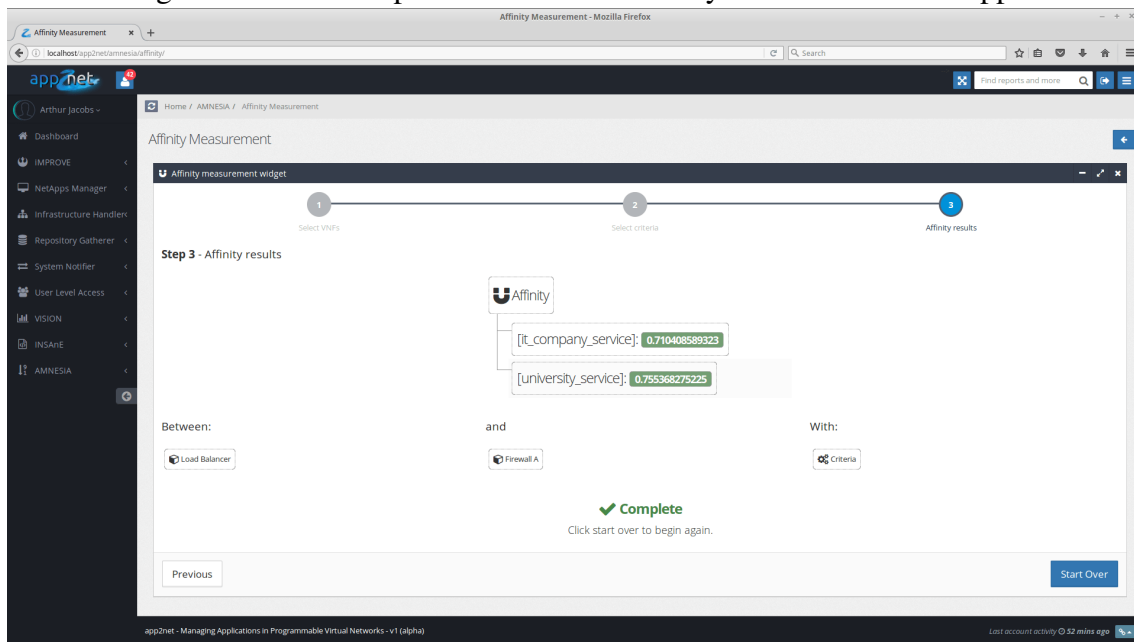
The screenshot shows the 'Affinity Measurement' application in a browser. The main content area is titled 'Step 2 - Select criteria'. It features a progress indicator with three steps: 'Select VNFs' (1), 'Select criteria' (2), and 'Affinity results' (3). Below the progress indicator is a table of criteria with the following columns: Type, Scope, Criterion, Formula, and Weight.

Type	Scope	Criterion	Formula	Weight
dynamic	FG	Bandwidth Usage	bwnd_usage_affinity	1
dynamic	FG	Latency	lat_affinity	1
dynamic	FG	Packet Loss	pkt_loss_affinity	1
dynamic	PM	CPU Usage	cpu_usage_affinity	1
dynamic	PM	Memory Usage	mem_usage_affinity	1
dynamic	PM	Storage Usage	sto_usage_affinity	1
static	FG	Conflicts Check	conflicts_affinity	1
static	PM	Minimum CPU Requirements	min_cpu_affinity	1
static	PM	Minimum Memory Requirements	min_mem_affinity	1
static	PM	Minimum Storage Requirements	min_sto_affinity	1

Source: Author

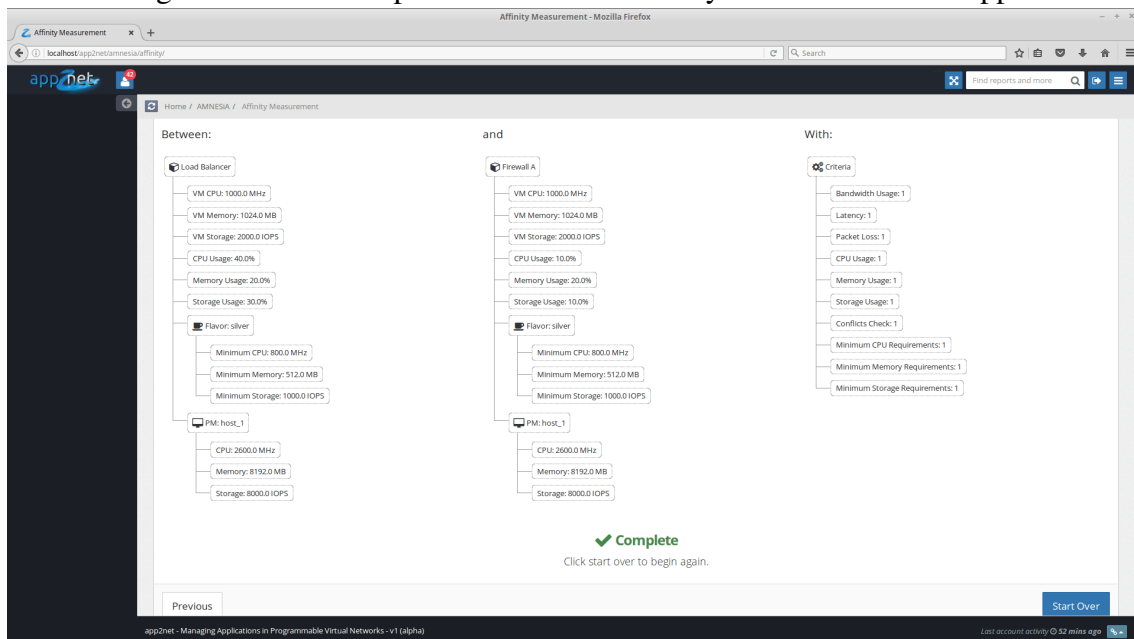
results. In this screen we present the affinity values for the pair of selected VNFs, with the given criteria weights. As described on Section 3.2, if the selected VNFs share more than one FG, the affinity measurement is calculated for each FG, since they may have different flows. In turn, if the VNFs do not share any FG, the affinity measurement is only calculated regarding PM criteria. Figures 4.5 and 4.6 present the affinity result screen when VNFs share multiple FGs.

Figure 4.5: Third step of AMNESiA's Affinity Measurement NetApp.



Source: Author

Figure 4.6: Third step of AMNESiA's Affinity Measurement NetApp.



Source: Author

4.1.2 Affinity Report

The Affinity Report NetApp is also a three-step wizard, likewise the Affinity Measurement NetApp. However, instead of selecting a pair of VNFs, the user must select an FG. This feature's purpose is to measure the affinity of every two VNFs combination of an FG, aiming to provide an overview of the overall state of the service's affinity. We

do understand this approach would not scale on a real-life NFV environment, with hundreds of VNFs, since finding every combination of pairs has a computational complexity of $O(n^2)$. This issue shall be considered on future work, possibly being addressed using graph search algorithms, which normally have a polynomial complexity of $O(n + m)$, proportional to number of vertices and edges of the graph. Nonetheless, for the scope of this prototype, the current binomial combination approach suffices. In addition, this feature only works if a single FG is selected.

Figure 4.7 display the AMNESiA screen of the first step from the Affinity Report NetApp. In this screen, we do not present information regarding VNFs. Instead, we provide usage metrics regarding every flow of each FG, such as latency, bandwidth usage, packet loss and traffic in every link (either virtual or physical). After selecting an FG, the user is taken to the same second step as in the Affinity Measurement NetApp, to provide weights for each criterion.

Figure 4.7: First step of AMNESiA's Affinity Report NetApp.

The screenshot shows the 'Affinity Report' interface in a browser. The main content area is titled 'Step 1 - Select one Forwarding Graph'. It features a table with the following columns: 'Forwarding Graph', 'Flow', 'Latency', 'Traffic', 'Bandwidth Usage', and 'Packet Loss'. The table is divided into three sections, each corresponding to a selected Forwarding Graph (FG):

- [university_service]**: Contains flows such as 'Internet/Tenant -> Load Balancer', 'Load Balancer -> Firewall A', 'Load Balancer -> Firewall B', 'Firewall A -> IDS', and 'Firewall B -> IDS'.
- [bank_service]**: Contains flows such as 'Internet/Tenant -> DPI', 'DPI -> Firewall B', 'Firewall B -> IPS', and 'IPS -> Internet/Tenant'.
- [it_company_service]**: Contains flows such as 'Internet/Tenant -> Load Balancer', 'Load Balancer -> Firewall A', 'Load Balancer -> Firewall B', 'Firewall A -> DPI', 'Firewall B -> DPI', and 'DPI -> Packet Sniffer'.

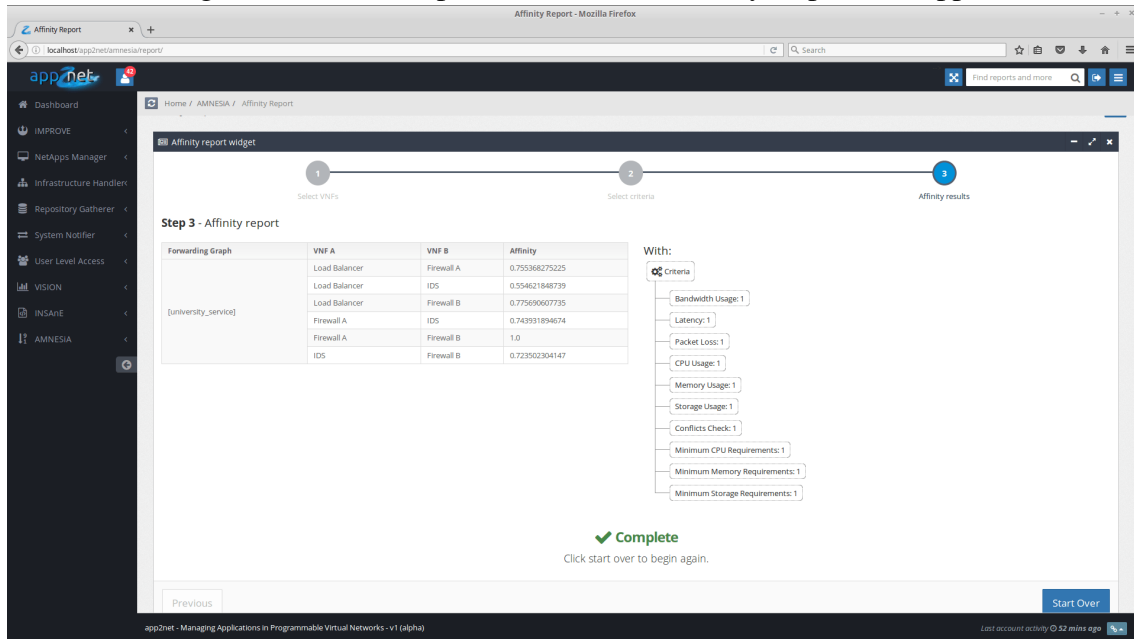
Each flow row provides specific metrics for Latency, Traffic, Bandwidth Usage, and Packet Loss. For example, the 'Internet/Tenant -> Load Balancer' flow for [university_service] has a latency of 10.0 ms, traffic of 500.0 Mbit/s, bandwidth usage of 50.0%, and a packet loss of 1.0%.

Source: Author

After stipulating the weights of the criteria, the user is then brought to the third and final step of the Affinity Report NetApp: the affinity results. In this screen, we display a table containing all the two VNFs combination of the selected FG, alongside the affinity result for that pair. Since in this feature the user already provides an FG to measure affinity with, we only consider connections between VNFs regarding the selected FG. Once again, if two VNFs are not directly connected in the FG, then solely the PM criteria are considered by the measurement. Figure 4.8 displays the affinity results of the Affinity

Report NetApp.

Figure 4.8: Third step of AMNESiA's Affinity Report NetApp.



Source: Author

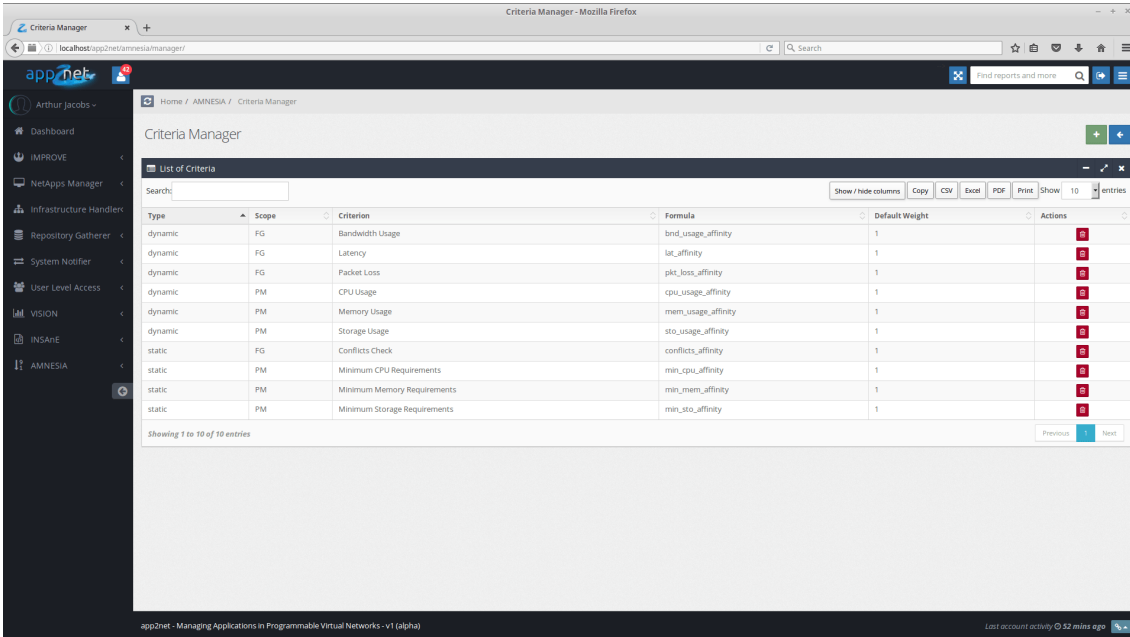
4.1.3 Criteria Manager

Finally, the last NetApp in the AMNESiA branch is the Criteria Manager. This feature is designed to enable network operators to add or remove criteria as they wish. With that purpose in mind, we present the user a table containing list of all the criteria in the application, as shown in Figure 4.9. Every row of the table has a delete action, which the operator might use to remove the criterion from the system.

To add a new criterion, the user must press the green button with a plus icon, on the top-right corner of the screen. This button opens up a new view, containing a form to fill the new criterion's information, as shown in Figure 4.10. The form's information include: criterion's name (which must be a unique identifier, containing no spaces), label (name that is actually presented on screen), type and scope, the default weight (value is initially displayed to the user when providing weights to criteria), and the criterion's affinity equation.

The criterion's affinity equation must be a Python function, receiving as input the objects of both VNFs being evaluated, the FG object to consider, and the NSD object; and returning a numerical value between 0.001 and 1. The text area for the affinity equation

Figure 4.9: AMNESiA's Criteria Manager NetApp.



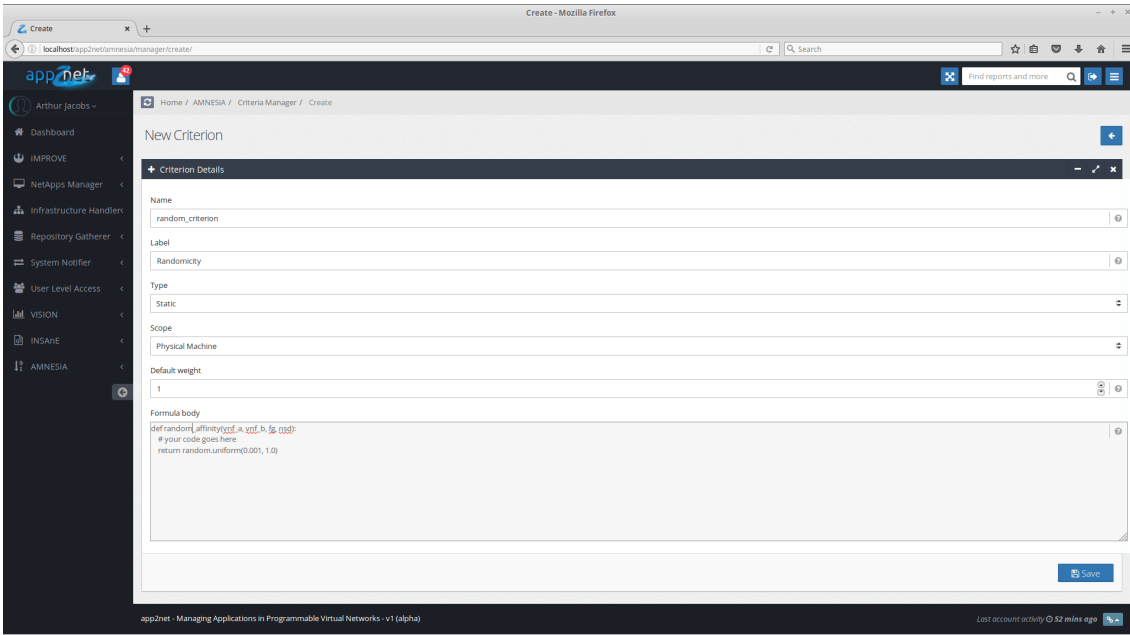
The screenshot shows the 'Criteria Manager' interface in a web browser. The main content is a table titled 'List of Criteria' with the following data:

Type	Scope	Criterion	Formula	Default Weight	Actions
dynamic	FG	Bandwidth Usage	bnd_usage_affinity	1	[+]
dynamic	FG	Latency	lat_affinity	1	[+]
dynamic	FG	Packet Loss	pkt_loss_affinity	1	[+]
dynamic	PM	CPU Usage	cpu_usage_affinity	1	[+]
dynamic	PM	Memory Usage	mem_usage_affinity	1	[+]
dynamic	PM	Storage Usage	sto_usage_affinity	1	[+]
static	FG	Conflicts Check	conflicts_affinity	1	[+]
static	PM	Minimum CPU Requirements	min_cpu_affinity	1	[+]
static	PM	Minimum Memory Requirements	min_mem_affinity	1	[+]
static	PM	Minimum Storage Requirements	min_sto_affinity	1	[+]

The interface also includes a search bar, a 'Showing 1 to 10 of 10 entries' indicator, and navigation buttons for 'Previous' and 'Next'.

Source: Author

Figure 4.10: AMNESiA's screen to add new criterion.



The screenshot shows the 'New Criterion' form in the AMNESiA interface. The form fields are as follows:

- Name:** random_criterion
- Label:** Randomicity
- Type:** Static
- Scope:** Physical Machine
- Default weight:** 1
- Formula body:**

```
def random_affinity(v):
    # your code goes here
    return random.uniform(0.001, 1.0)
```

The form includes a 'Save' button at the bottom right.

Source: Author

has an initial value of a function skeleton, providing some guidance for the user to fill the form. A clear shortcoming of this approach is that, for a network operator to include a new criterion, internal knowledge of the application's development is required, not to mention Python skills. This issue might be addressed by developing a graphical interface to build equations. However, that type of user interaction is out of the scope of the implementation, as AMNESiA is a prototype application. Further information on the implementation, such

as objects description and usage, is provided on Appendix A.

After filling out the criterion inclusion form, and clicking on the button to save, the system parses the affinity equation, to fetch the function's name, and then stores the new criterion in the database. The body of affinity function is appended on the equations Python file, alongside the other criteria affinity functions. The included criterion can now be selected among existing criteria when measuring affinity.

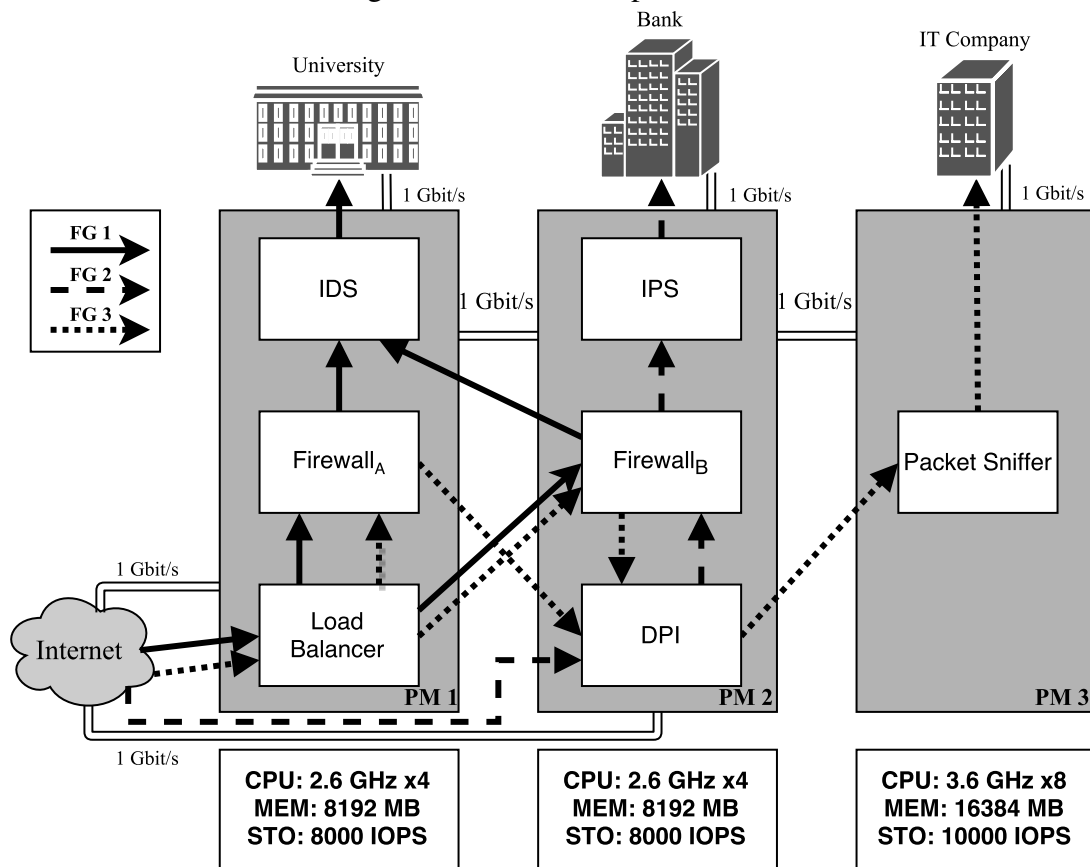
5 CASE STUDIES

To evaluate our affinity model, we analyze a VNF as a Service [Xilouris et al. 2015] (VNFaaS) scenario with multiple tenants sharing the same infrastructure and VNFs. To facilitate the affinity measurement of VNFs, we rely on features provided by AMNESiA, specially on the Affinity Report. Further, 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 5.1 illustrates our evaluation scenario, built inside AMNESiA, 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 5.1 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 5.1 includes the bandwidth capacity of physical links between PMs, PMs to the Internet, and PMs to tenants. In AMNESiA, FGs 1, 2 and 3 have been given the names [university_service], [bank_service] and [it_company_service], respectively.

Table 5.1 shows a snapshot of the current resource usage for each PM: CPU, memory and IOPS. In addition, Table 5.1 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 80% of CPU usage, from which the load balancer is responsible for 40%. Meanwhile, Table 5.2 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.

Figure 5.1: NFV example scenario.



Source: Author

5.1 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. Figure 5.2 presents the resulting affinities of AMNESiA's Affinity Report on FG 1, which includes all VNFs hosted in PM 1, given that all the criteria have the same weight 1.

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

Table 5.1: PMs resources usage of example scenario.

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%

more focused on execution stats, he/she can provide input weights with zero value for the static criteria. Figure 5.3 produces the affinity results directed to the dynamic behavior in this scenario. Therefore, all static criteria receive zero as input weights.

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.

5.2 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

Table 5.2: FGs resources usage of example scenario.

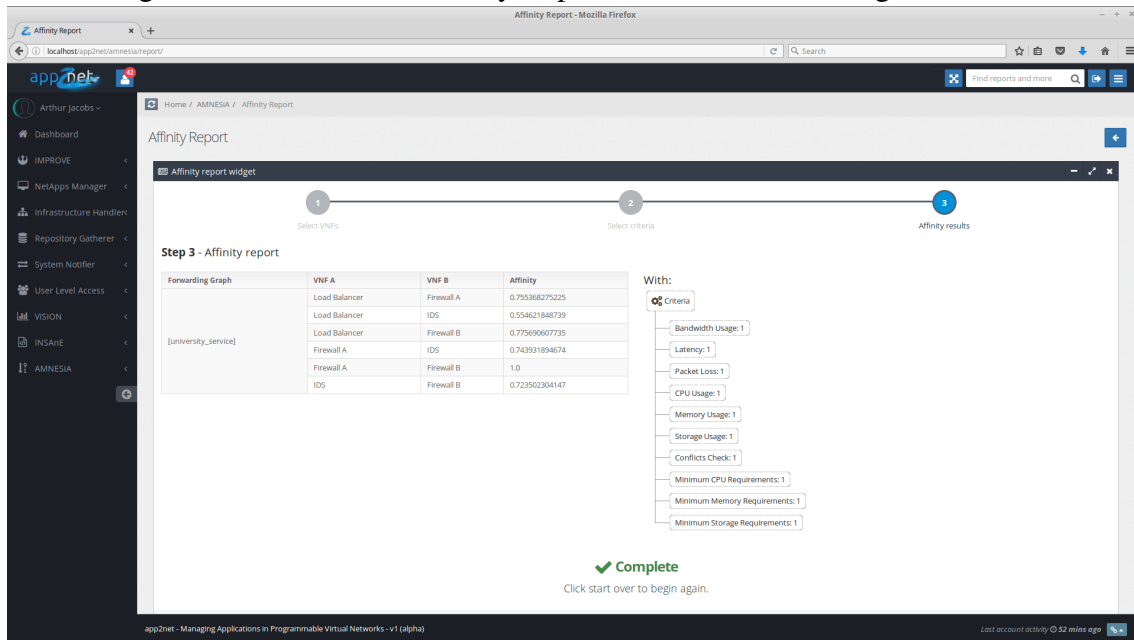
FG	Flow	Traffic	Bandwidth Usage	Packet Loss	Latency
1	<i>Internet</i> → <i>Load Balancer</i>	500 Mbit/s	50%	1%	10 ms
	<i>Load Balancer</i> → <i>Firewall_A</i>	200 Mbit/s	25%	1%	1 ms
	<i>Load Balancer</i> → <i>Firewall_B</i>	200 Mbit/s	37%	1%	5 ms
	<i>Firewall_A</i> → <i>IDS</i>	100 Mbit/s	10%	1%	1 ms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

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

With these values exposed, 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. Figure 5.4 exposes the affinity measure from our model in this case.

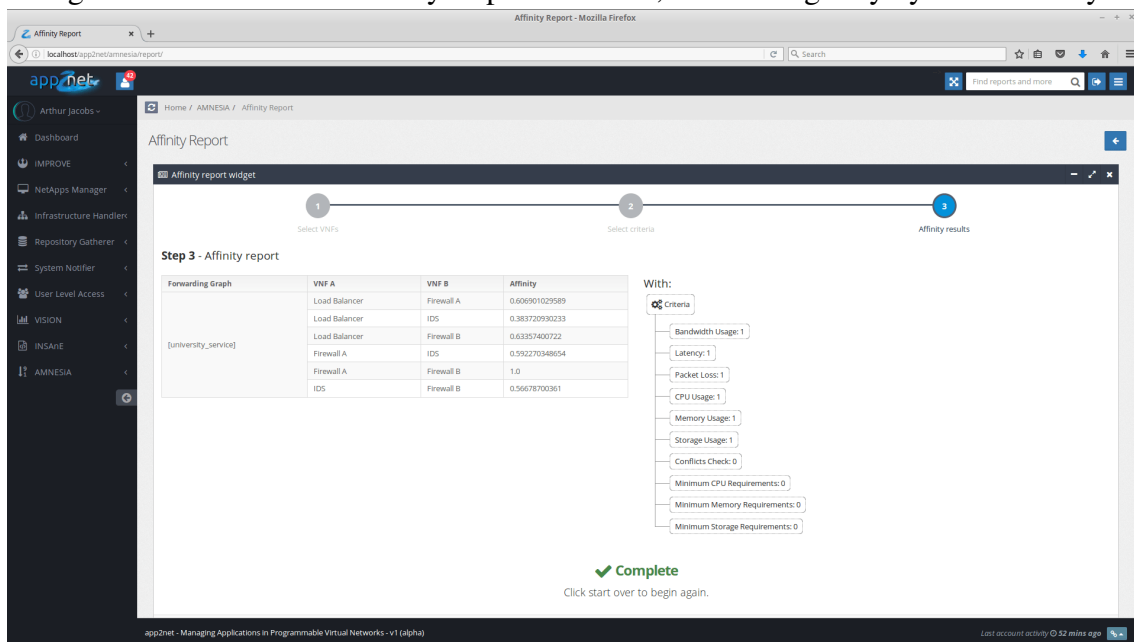
These values reveal an affinity relationship between any pair of VNFs in the FG.

Figure 5.2: AMNESiA Affinity Report for FG 1, considering all criteria.



Source: Author

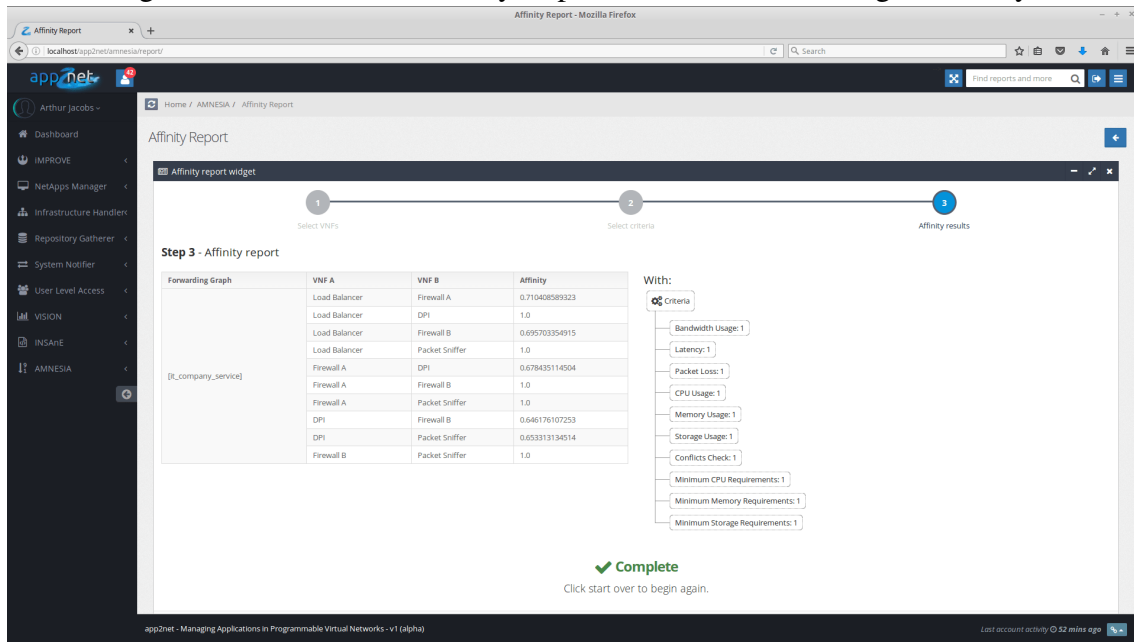
Figure 5.3: AMNESiA Affinity Report for FG 1, considering only dynamic affinity.



Source: Author

This occurs because the combination of small traffic and high latency reduces the network affinity in Equation 3.17. Thus, since the static criteria affinity equals to 1, the total affinity results in a measure larger than 0.5. In summary, the analysis of these values can prevent the operator from misplacing VNFs based only on latency observations.

Figure 5.4: AMNESiA Affinity Report for FG 3, considering all affinity.



Source: Author

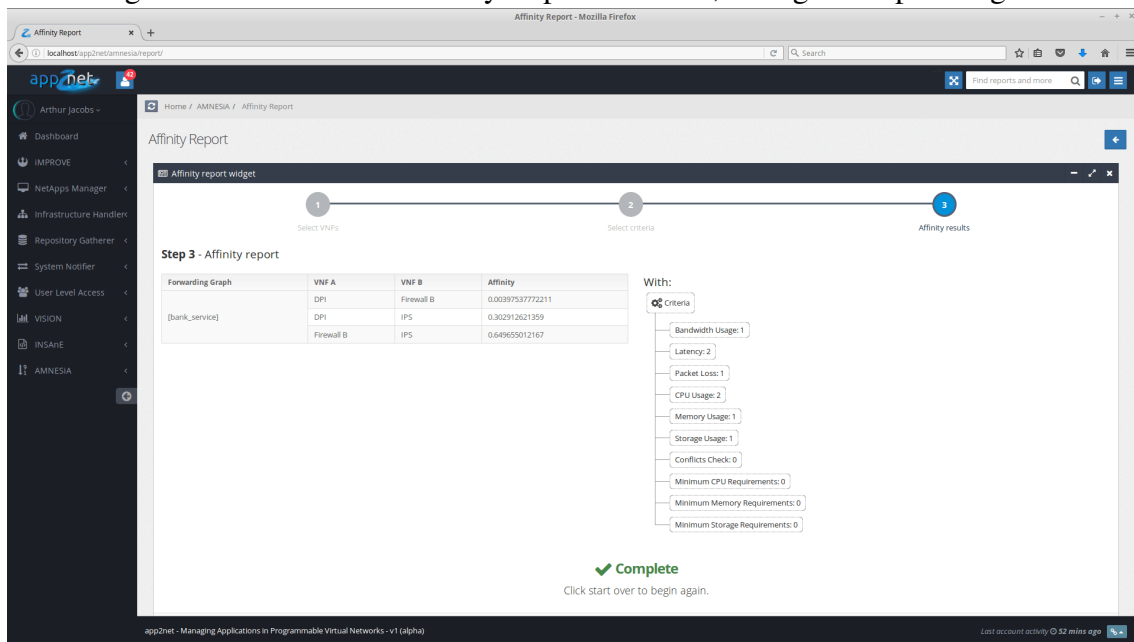
5.3 Case Study #3

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. Figure 5.5 presents the resultant affinities for the VNFs on FG 2.

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. Figure 5.6 presents the

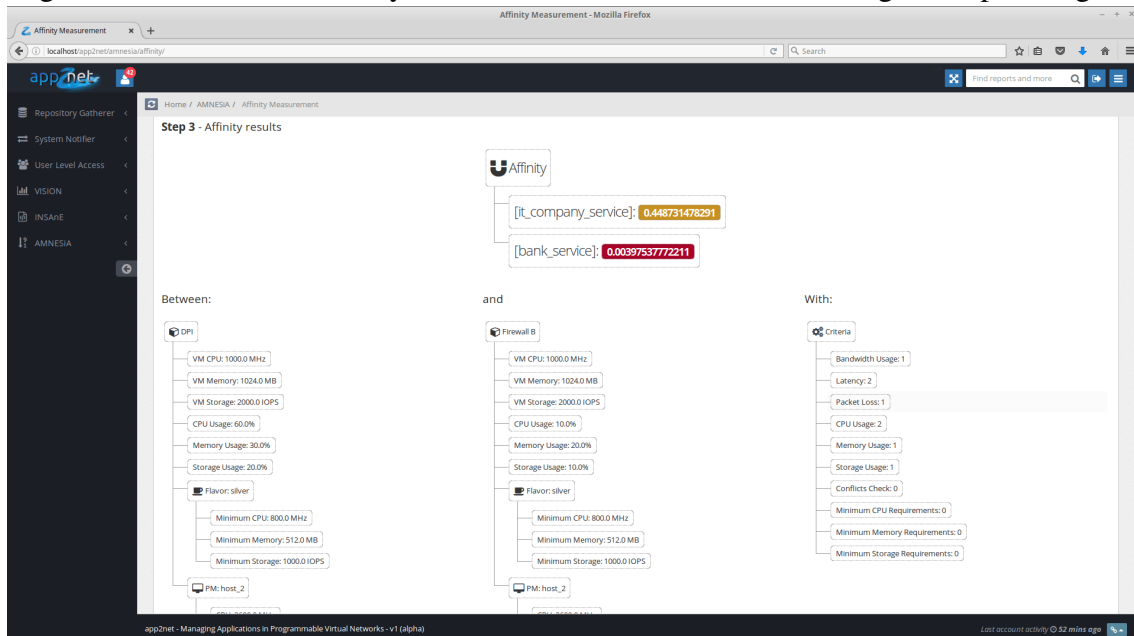
Figure 5.5: AMNESiA Affinity Report for FG 2, with given input weights.



Source: Author

affinity measure for the DPI and the firewall B for both FG 2 and FG 3, considering the same weights as in Figure 5.5. The resulting affinity for FG 2 is low due to PM resource contention, but since the FG criteria are normalized for FG 3, the measure is much higher.

Figure 5.6: AMNESiA affinity results for Firewall B and DPI, with given input weights.



Source: Author

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.

6 CONCLUSION AND FUTURE WORK

In this document, 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 placement and migration. In summary, we *(i)* provided an affinity relationship definition, *(ii)* defined an extendable set of affinity criteria, *(iii)* described a mathematical model to measure the affinity between any pair of VNFs; and *(iv)* presented AMNESiA, a PoC implementation of the proposed affinity model.

AMNESiA was developed in Python, using the Django Web Framework, and incorporated as an affinity branch inside App2Net [Santos et al. 2015]. AMNESiA serves the purpose of demonstrating that the implementation of the proposed affinity model is feasible, helping us identify several issues on the experimental scenario. In addition, because of the complexity of the proposed affinity measurement, with many distinct equations, AMNESiA helped us evaluate the affinity measurement on the experimental scenario much easier and efficiently than evaluating those equations manually.

We analyzed our affinity measure, relying on AMNESiA, 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.

Developing the proposed affinity model was an insightful experience. Taking a grasp of the problem we intended to solve required extensive research, on both the NFV and affinity concepts, as these topics are not covered in undergraduation courses. Moreover, making this affinity measurement solution both well-rounded and flexible at the

same time was very challenging, since the requirements for affinity may vary on a case to case basis. Nevertheless, using mathematical concepts learned throughout the undergraduate course, we were able to combine several criteria to provide an extensible solution that fits distinct scenarios. Still, the carried-out research allowed us to have insights on many different real-world scenarios and related issues, which are imperative to achieve innovative technologies such as 5G, that relies heavily on NFV.

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 an IPS). Another foreseeable improvement to the affinity measure is to extend it to consider more than one pair of VNFs, possibly relying on set theory concepts. In addition, we plan to store every affinity measurement made on AMNESiA, aiming to provide historical data for each pair of VNFs, and use it as input to retroactively improve the affinity measures. Moreover, AMNESiA's Affinity Report feature might be extended to include such historical data on VNFs, as well as improved to achieve a computational complexity fit for real-world scenarios. Finally, we aim to integrate AMNESiA with VISION [Franco et al. 2016], incorporating the affinity measurements on a forwarding graphs visualizations to aid network operators on having insights about the network.

REFERENCES

- ALCATEL-LUCENT. **Network Functions Virtualization - Challenges and Solutions**. [S.l.], 2013.
- CHEN, J. et al. AAGA: Affinity-Aware Grouping for Allocation of Virtual Machines. In: **Advanced Information Networking and Applications (AINA), 2013 IEEE 27th International Conference on**. [S.l.: s.n.], 2013. p. 235–242. ISSN 1550-445X.
- FEAMSTER, N.; REXFORD, J.; ZEGURA, E. The Road to SDN: An Intellectual History of Programmable Networks. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 44, n. 2, p. 87–98, abr. 2014. ISSN 0146-4833.
- FRANCO, M. et al. 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: [s.n.], 2016.
- HAWILO, H. et al. NFV: state of the art, challenges, and implementation in next generation mobile networks (vEPC). **IEEE Network**, v. 28, n. 6, p. 18–26, Nov 2014. ISSN 0890-8044.
- ISG, E. N. **Network Functions Virtualisation**. [S.l.], 2012. Available from Internet: <http://portal.etsi.org/NFV/NFV_White_Paper.pdf>.
- ISG, E. N. **Network Functions Virtualisation: Architectural Framework**. [S.l.], 2013. Available from Internet: <http://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf>.
- ISG, E. N. **Network Functions Virtualisation: Use Cases**. [S.l.], 2013. Available from Internet: <http://www.etsi.org/deliver/etsi_gs/nfv/001_099/001/01.01.01_60/gs_nfv001v010101p.pdf>.
- ISG, E. N. **Network Functions Virtualisation: Management and Orchestration**. [S.l.], 2014. Available from Internet: <http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf>.
- ISG, E. N. **Network Functions Virtualisation: Terminology for Main Concepts in NFV**. [S.l.], 2014. Available from Internet: <http://www.etsi.org/deliver/etsi_gs/NFV/001_099/003/01.02.01_60/gs_nfv003v010201p.pdf>.
- MIJUMBI, R. et al. Network Function Virtualization: State-of-the-art and Research Challenges. **CoRR**, abs/1509.07675, 2015. Available from Internet: <<http://arxiv.org/abs/1509.07675>>.
- OECHSNER, S.; RIPKE, A. Flexible support of VNF placement functions in OpenStack. In: **Network Softwarization (NetSoft), 2015 1st IEEE Conference on**. [S.l.: s.n.], 2015. p. 1–6.
- PFITSCHER, R. J. et al. DReAM-a distributed result-aware monitor for Network Functions Virtualization. In: **IEEE. 2016 IEEE Symposium on Computers and Communication (ISCC)**. [S.l.], 2016. p. 663–668.

SANTOS, R. L. dos et al. App2net: A platform to transfer and configure applications on programmable virtual networks. In: **2015 IEEE Symposium on Computers and Communication (ISCC)**. [S.l.: s.n.], 2015. p. 315–322.

SONNEK, J. et al. Starling: Minimizing Communication Overhead in Virtualized Computing Platforms Using Decentralized Affinity-Aware Migration. In: **2010 39th International Conference on Parallel Processing**. [S.l.: s.n.], 2010. p. 228–237. ISSN 0190-3918.

SUDEVALAYAM, S.; KULKARNI, P. Affinity-Aware Modeling of CPU Usage for Provisioning Virtualized Applications. In: **Cloud Computing (CLOUD), 2011 IEEE International Conference on**. [S.l.: s.n.], 2011. p. 139–146. ISSN 2159-6182.

WOOD, T. et al. 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**. New York, NY, USA: ACM, 2009. (VEE '09), p. 31–40. ISBN 978-1-60558-375-4. Available from Internet: <<http://doi.acm.org/10.1145/1508293.1508299>>.

XILOURIS, G. et al. T-NOVA: Network functions as-a-service over virtualised infrastructures. In: **Network Function Virtualization and Software Defined Network (NFV-SDN), 2015 IEEE Conference on**. [S.l.: s.n.], 2015. p. 13–14.

YOSHIDA, M. et al. MORSA: A multi-objective resource scheduling algorithm for NFV infrastructure. In: **Network Operations and Management Symposium (APNOMS), 2014 16th Asia-Pacific**. [S.l.: s.n.], 2014. p. 1–6.

YOUSAF, F. Z.; TALEB, T. Fine-grained resource-aware virtual network function management for 5G carrier cloud. **IEEE Network**, v. 30, n. 2, p. 110–115, March 2016. ISSN 0890-8044.

APPENDIX A — AMNESIA REFERENCE

In this Appendix, we present AMNESiA implementation details, providing model references for all objects in the system. Relying on this reference, any user with Python skills would be able to include a new criterion in the system, properly using the affinity formula function input variables.

All object classes in AMNESiA are Django models, which are responsible for creating and managing related database tables and entries, in addition to memory objects. To facilitate and standardize model definitions, we first create a base model, named `BaseModel`, which all other models extend. Hence, before presenting any other AMNESiA models, we introduce below the class definition of the Base model.

```
from django.db import models

class BaseModel(models.Model):
    class Meta:
        abstract = True
        app_label = 'amnesia'
```

Having defined the Base model, we now able to extend it to create other model classes. Below, we present classes for the VNF model, the `PhysicalMachine` model, the `Flavor` model, the `ForwardingGraph` and `Flow` models, the `Descriptor` and `Conflicts` models, and the `Criterion` model.

A.1 VNF

The VNF model contains the most information, as it is a main point of reference of the system. All the properties included in this model can be used on when providing an affinity formula function.

```
from base import *
from flavor import *
from physical_machine import *
from forwarding_graph import *
```

```

class VNF(BaseModel):
    label = models.CharField(max_length=50)
    vm_cpu = models.FloatField()
    vm_memory = models.FloatField()
    vm_storage = models.FloatField()
    cpu_usage = models.FloatField()
    memory_usage = models.FloatField()
    storage_usage = models.FloatField()

    flavor = models.ForeignKey(Flavor)
    physical_machine = models.ForeignKey(PhysicalMachine)
    forwarding_graphs = models.ManyToManyField(ForwardingGraph)

    def __unicode__(self):
        return self.label

```

A.2 Flavor

The Flavor model contains the requirement specification of each VNF that has that flavor. As mentioned before, management and registration of flavors are out of the scope of AMNESiA. Nonetheless, this information may be used during affinity measurement.

```

from base import *

class Flavor(BaseModel):
    label = models.CharField(max_length=50)
    minimum_cpu = models.FloatField()
    minimum_memory = models.FloatField()
    minimum_storage = models.FloatField()

    def __unicode__(self):
        return self.label

```


A.3 Physical Machine

The `PhysicalMachine` model holds information of the hardware hosting VNF. One `PhysicalMachine` may be connected to many VNFs.

```
from base import *

class PhysicalMachine(BaseModel):
    label = models.CharField(max_length=50)
    cpu = models.FloatField()
    memory = models.FloatField()
    storage = models.FloatField()

    class Meta():
        db_table = 'amnesia_physical_machine'

    def __unicode__(self):
        return self.label
```

A.4 Forwarding Graph and Flow

The `ForwardingGraph` and `Flow` models contains information regarding services provided by chained VNFs. Each `ForwardingGraph` has one or many `Flow` related objects. In turn, each `Flow` has two related VNFs, a source and a destination of the flow. In addition, the `Flow` model holds all information of links, physical or virtual, connecting the source and destination VNFs.

```
from base import *

class Flow(BaseModel):
    source = models.ForeignKey(
        "VNF",
        related_name="source_vnf"
    )
    destination = models.ForeignKey(
```

```

        "VNF",
        related_name="destination_vnf"
    )
    latency = models.FloatField()
    traffic = models.FloatField()
    bandwidth_usage = models.FloatField()
    packet_loss = models.FloatField()

    def __unicode__(self):
        return self.src.label + " -> " + self.dst.label

class ForwardingGraph(BaseModel):
    label = models.CharField(max_length=50)
    flows = models.ManyToManyField(Flow)

    class Meta:
        db_table = "amnesia_forwarding_graph"

    def __unicode__(self):
        return self.label

```

A.5 Descriptor and Conflict

The `Descriptor` model was built to represent any possible descriptor (*e.g.*, NSD, VNFD, VNFFGD, etc.) on an NFV scenario. However, for the scope of AMNESiA, we only populated the model with the necessary information to measure affinity for the presented criteria. To enhance the application, a logical first step would be to include in the `Descriptor` model all the fields of the descriptors described in [ISG 2014]. In addition, we added to the `Descriptor` model a list known conflicts of VNFs, representing all VNFs that should not be directly linked on the same FG.

```

from base import *
from conflict import *

```

```

class Descriptor(BaseModel):
    label = models.CharField(max_length=50)
    sla = models.FloatField()
    conflicts = models.ManyToManyField(Conflict)

    def __unicode__(self):
        return self.label

```

A.6 Criterion

Finally, the Criterion model holds all the information provided when including a new criterion in AMNESiA. In addition, this model includes the weight property, in which the last used weight to measure affinity is stored into.

```

from base import *

```

```

class Criterion(BaseModel):
    CRITERIA_TYPES = (
        ('static', 'Static'),
        ('dynamic', 'Dynamic'),
    )
    CRITERIA_SCOPE = (
        ('PM', 'Physical Machine'),
        ('FG', 'Forwarding Graph'),
    )
    name = models.CharField(max_length=50, unique=True)
    label = models.CharField(max_length=100, default="Criterion")
    type = models.CharField(max_length=50, choices=CRITERIA_TYPES)
    scope = models.CharField(max_length=50, choices=CRITERIA_SCOPE)
    weight = models.IntegerField(default=0)
    default_weight = models.IntegerField(default=1)
    formula = models.CharField(max_length=100, default="")

    def __unicode__(self):
        return self.name

```

APPENDIX B — ARTICLE — IM 2017

In this appendix, we present the paper entitled "Affinity Measurement for NFV-enabled Networks: A Criteria-based Approach". This paper presents, on a reputed scientific conference (IFIP/IEEE International Symposium on Integrated Network Management), the affinity measurement proposed in this document.

- **Title:**

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

- **Conference:**

15th IFIP/IEEE International Symposium on Integrated Network Management (IM 2017)

- **URL:**

<<http://im2017.ieee-im.org/>>

- **Date:**

8-12 May 2017

- **Local:**

Lisbon, Portugal

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	$\alpha = \begin{cases} 1 & \text{if } (cpu_{vnfa} \geq cpu_{NSD}) \wedge (cpu_{vnfb} \geq cpu_{NSD}), \\ (1 + \max(0.001, \frac{cpu_{vnfb}}{cpu_{NSD}})) \times 0.5 & \text{if } (cpu_{vnfa} \geq cpu_{NSD}) \wedge (cpu_{vnfb} < cpu_{NSD}), \\ (\max(0.001, \frac{cpu_{vnfa}}{cpu_{NSD}}) + 1) \times 0.5 & \text{if } (cpu_{vnfa} < cpu_{NSD}) \wedge (cpu_{vnfb} \geq cpu_{NSD}), \\ (\max(0.001, \frac{cpu_{vnfa}}{cpu_{NSD}}) + \max(0.001, \frac{cpu_{vnfb}}{cpu_{NSD}})) \times 0.5 & \text{otherwise.} \end{cases}$
Minimum memory	$\alpha = \begin{cases} 1 & \text{if } (mem_{vnfa} \geq mem_{NSD}) \wedge (mem_{vnfb} \geq mem_{NSD}), \\ (1 + \max(0.001, \frac{mem_{vnfb}}{mem_{NSD}})) \times 0.5 & \text{if } (mem_{vnfa} \geq mem_{NSD}) \wedge (mem_{vnfb} < mem_{NSD}), \\ (\max(0.001, \frac{mem_{vnfa}}{mem_{NSD}}) + 1) \times 0.5 & \text{if } (mem_{vnfa} < mem_{NSD}) \wedge (mem_{vnfb} \geq mem_{NSD}), \\ (\max(0.001, \frac{mem_{vnfa}}{mem_{NSD}}) + \max(0.001, \frac{mem_{vnfb}}{mem_{NSD}})) \times 0.5 & \text{otherwise.} \end{cases}$
Minimum storage	$\alpha = \begin{cases} 1 & \text{if } (sto_{vnfa} \geq sto_{NSD}) \wedge (sto_{vnfb} \geq sto_{NSD}), \\ (1 + \max(0.001, \frac{sto_{vnfb}}{sto_{NSD}})) \times 0.5 & \text{if } (sto_{vnfa} \geq sto_{NSD}) \wedge (sto_{vnfb} < sto_{NSD}), \\ (\max(0.001, \frac{sto_{vnfa}}{sto_{NSD}}) + 1) \times 0.5 & \text{if } (sto_{vnfa} < sto_{NSD}) \wedge (sto_{vnfb} \geq sto_{NSD}), \\ (\max(0.001, \frac{sto_{vnfa}}{sto_{NSD}}) + \max(0.001, \frac{sto_{vnfb}}{sto_{NSD}})) \times 0.5 & \text{otherwise.} \end{cases}$
NFV conflicts	$\alpha = \begin{cases} 1 & \text{if the two VNFs respect conflicts,} \\ 0.001 & \text{otherwise.} \end{cases}$
CPU usage	$\alpha = \max(0.001, 1 - (\frac{\%cpu_{vnfa} + \%cpu_{vnfb}}{100}))$
Memory usage	$\alpha = \max(0.001, 1 - (\frac{\%mem_{vnfa} + \%mem_{vnfb}}{100}))$
Storage usage	$\alpha = \max(0.001, 1 - (\frac{\%sto_{vnfa} + \%sto_{vnfb}}{100}))$
Bandwidth usage	$\alpha = \max(0.001, 1 - (\frac{\%bnd_{(vnfa, vnfb)}}{100}))$
Packet loss	$\alpha = \max(0.001, 1 - (\frac{\%pkt_loss_{(vnfa, vnfb)}}{100}))$
Latency	$\alpha = \begin{cases} 1 & \text{if } 2 \times lat_{(vnfa, vnfb)} \leq lat_{SLA}, \\ \max(0.001, 1 - \frac{2 \times lat_{(vnfa, vnfb)} - lat_{SLA}}{lat_{SLA}}) & \text{otherwise.} \end{cases}$

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.

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

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

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.

$$\alpha_s = \frac{2}{\frac{1}{\alpha_{spm}} + \frac{1}{\alpha_{sfg}}} \quad (3)$$

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.

$$\alpha_d = \frac{x + y}{\frac{x}{\alpha_{dpm}} + \frac{y}{\alpha_{net}}} \quad (4)$$

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

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

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.

$$\alpha_{net} = 0.5 + ((\alpha_{trf}/2) \times (\alpha_{dfg} - (1 - \alpha_{dfg}))) \quad (7)$$

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} α_{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.

$$\alpha_{trf} = \frac{trf_{(vnfa, vnfb)}}{hgst_trf_{fg}} \quad (8)$$

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.

$$\alpha_x = \frac{\sum_{i=1}^{n_x} w_i}{\sum_{i=1}^{n_x} \alpha C_i}, \quad \forall x \in \{spm, sfg, dpm, dfg\} \quad (9)$$

n_x , number of criteria of x .
 $w_i \in \mathbb{N}$, weight for criterion i .
 αC_i , 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.

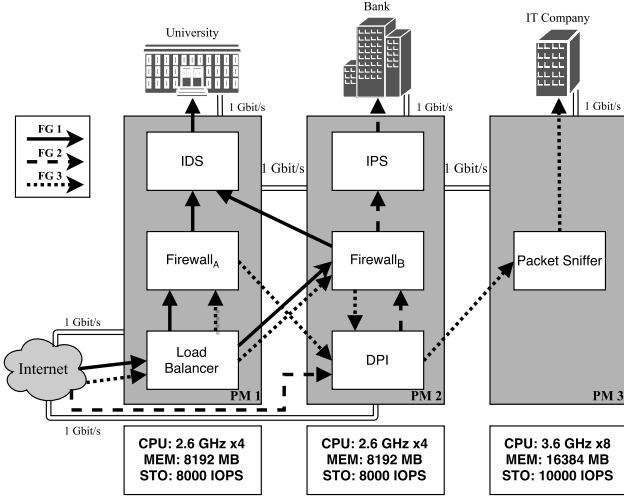


Figure 2: NFV example scenario.

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.

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

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.

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

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	<i>Internet</i> → <i>Load Balancer</i>	500 Mbit/s	50%	1%	10 ms
	<i>Load Balancer</i> → <i>Firewall_A</i>	200 Mbit/s	25%	1%	1 ms
	<i>Load Balancer</i> → <i>Firewall_B</i>	200 Mbit/s	37%	1%	5 ms
	<i>Firewall_A</i> → <i>IDS</i>	100 Mbit/s	10%	1%	1 ms
	<i>Firewall_B</i> → <i>IDS</i>	100 Mbit/s	37%	1%	5 ms
	<i>IDS</i> → <i>University</i>	200 Mbit/s	20%	1%	10 ms
2	<i>Internet</i> → <i>DPI</i>	200 Mbit/s	20%	1%	10 ms
	<i>DPI</i> → <i>Firewall_B</i>	200 Mbit/s	22%	1%	35 ms
	<i>Firewall_B</i> → <i>IPS</i>	40 Mbit/s	4%	1%	1 ms
	<i>IPS</i> → <i>Bank</i>	40 Mbit/s	4%	1%	10 ms
3	<i>Internet</i> → <i>Load Balancer</i>	500 Mbit/s	50%	1%	10 ms
	<i>Load Balancer</i> → <i>Firewall_A</i>	50 Mbit/s	25%	1%	1 ms
	<i>Load Balancer</i> → <i>Firewall_B</i>	50 Mbit/s	37%	1%	5 ms
	<i>Firewall_A</i> → <i>DPI</i>	20 Mbit/s	37%	1%	5 ms
	<i>Firewall_B</i> → <i>DPI</i>	20 Mbit/s	22%	1%	1 ms
	<i>DPI</i> → <i>Packet Sniffer</i>	40 Mbit/s	4%	1%	28 ms
	<i>Packet Sniffer</i> → <i>IT Company</i>	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 occurs 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 these 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.

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

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.

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

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.

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 - Challenges 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.