This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2017.2681025, IEEE Transactions on Network and Service Management

Semantically Enhanced Mapping Algorithm for Affinity-Constrained Service Function Chain Requests

Niels Bouten*, Rashid Mijumbi†, Joan Serrat‡, Jeroen Famaey§, Steven Latré§, Filip De Turck*
*Department of Information Technology, Ghent University - iMinds, Technologiepark-Zwijnaarde 15, B-9052 Ghent, Belgium
†Telecommunications Software and Systems Group, Waterford Institute of Technology, Ireland
‡Network Engineering Department, Universitat Politècnica de Catalunya, 08034 Barcelona, Spain
§Department of Mathematics and Computer Science, University of Antwerp, Middelheimlaan 1, B-2020 Antwerp, Belgium
email: niels.bouten@intec.ugent.be

***Abstract*—Network Function Virtualization (NFV) and Software Defined Networking (SDN) have been proposed to increase the cost-efficiency, flexibility and innovation in network service provisioning. This is achieved by leveraging IT virtualization techniques and combining them with programmable networks. By doing so, NFV and SDN are able to decouple the network functionality from the physical devices on which they are deployed. Service Function Chains (SFCs) composed out of Virtual Network Functions (VNFs) can now be deployed on top of the virtualized infrastructure to create new value-added services. Current NFV approaches are limited to mapping the different VNFs to the physical substrate subject to resource capacity constraints. They do not provide the possibility to define location requirements with a certain granularity and constraints on the colocation of VNFs and virtual edges. Nevertheless, many scenarios can be envisioned in which a Service Provider (SP) would like to attach placement constraints for efficiency, resilience, legislative, privacy and economic reasons. Therefore, we propose a set of affinity and anti-affinity constraints, which can be used by SPs to define such placement restrictions. Furthermore, a semantic SFC validation framework is proposed that allows the Virtual Network Function Infrastructure Provider (VNFInP) to check the validity of a set of constraints and provide feedback to the SPs. This allows the VNFInP to filter out any non-valid SFC requests before sending them to the mapping algorithm, significantly reducing the mapping time.**

I. Introduction

In traditional telecommunications networks, network functionality is strongly tied to the physical network device it runs on. To adapt or create network services, dedicated network appliances need to be deployed and interconnected in a strict order. Together with the ever increasing requirements for high quality and stability, this has led to long product cycles, limited service agility and substantial dependence on specialized hardware. The Network Function Virtualization (NFV) paradigm [1], [2] has been introduced to alleviate the aforementioned issues by leveraging IT virtualization technology to decouple the network functionality from the physical infrastructure. Furthermore, by taking advantage of Software Defined Networking (SDN), the network control and forwarding functions can be decoupled to allow an automated network configuration to interconnect the various Virtual Network Functions (VNFs) in a flexible way. To fully exploit the benefits of both paradigms, the mapping of VNFs and the allocation of network resources interconnecting them, should be considered together. For example, when virtualizing the Customer Premises Equipment (CPE), not only the placement of the various functions (e.g., decoding function) needs to be optimized, also the allocation of network resources interconnecting these functions should be considered (e.g., minimum throughput, maximum latency) to be able to deliver an end-to-end service.

The concepts of NFV and SDN have introduced new business opportunities for Virtual Network Function Infrastructure Providers (VNFInPs), which act as brokers between Infrastructure Providers (InPs) and Service Providers (SPs) [3]. The VNFInPs lease the infrastructure provided by different InPs and deploy, orchestrate and interconnect VNFs to create end-to-end Service Function Chains (SFCs) [5]. Together with these new opportunities and stakeholders, a set of new interactions arises as well. For example, the SPs need a way to express their SFC requests and requirements to the VNFInP. In traditional network embedding approaches [6], only node and link restrictions can be specified. However, many scenarios can be envisioned where a SP might want to attach more detailed constraints concerning the placement and routing between Network Functions (NFs) as well as constraints on their colocation. For example, to increase efficiency, the SP may require the embedding of VNFs at a certain granularity (e.g., within the same datacenter or even on the same host). Other reasons for more detailed affinity and anti-affinity constraints could be resilience, economic, legislative and privacy issues. In this paper, a set of affinity and anti-affinity constraints is proposed that increases the control of SPs on the embedding of their SFC requests.

With this newfound ability to add custom constraints, the possibility arises that conflicting constraints are introduced by SPs in their requested SFCs. Therefore, the VNFInP requires a method to check the consistency of SFC requests and inform

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2017.2681025, IEEE Transactions on Network and Service Management

the SP on potential conflicts. For example, if the SP states that two VNFs of certain types need to be embedded on the same host, but also requires that any instances of functions with those VNF types are required to be embedded in distinct datacenters, a conflict arises. Since SFC requests can contain many VNFs, virtual edges and constraints, detecting conflicts within these requests is not a straightforward task, neither for human operators, nor for computer systems. Since conflicts can arise between sets of constraints, pairwise detection will not suffice. Therefore, this chapter proposes to take advantage of semantic modelling to define an ontology and rule set, which can be enriched with individuals based on the specific SFC request. Using a semantic reasoner, the consistency can be determined and subsequently the validity of the SFC request can be assessed. Furthermore, the proposed framework can also be used by SPs to validate the SFC embeddings made by the VNFInP with respect to the imposed affinity requirements.

Existing VNF and Virtual Network (VN) embedding approaches focus on the resource constraints posed by the SFC requests or take into account other types of constraints, such as forwarding latency, processor and resource sharing, VNF precedence constraints, etc. These approaches do not provide the opportunity to add placement constraints to both VNF placement and virtual edge allocation. Therefore, this paper focuses on the the affinity-constrained SFC placement problem. First, the resource and flow conservation constraints are modeled for the concurrent mapping of SFCs sets. Second, the affinity and anti-affinity constraints are added to the model. Finally, to allow scalable SFC mapping, a heuristic procedure is proposed. This heuristic approach takes advantage of additional knowledge of the SFC set to sort the individual SFCs in a way that optimizes the objective. Both the optimal and heuristic approaches are evaluated considering different metrics and use cases, such as the number of VNFs in an SFC, the size of the physical infrastructure, the number of SFCs in a set and the impact of applying semantic filtering to the SFC request set prior to embedding.

The contributions of this paper are fourfold. First, a set of affinity and anti-affinity constraints that can be attached to an SFC request by the SP are defined. Second, we propose and evaluate a semantic conflict detection mechanism that can be employed by the VNFInP to check the validity of SFC requests. In this way, SFCs containing inconsistent constraints can be filtered out by the VNFInP before the embedding step. Third, we propose a mathematical formulation of the affinity-constrained SFC embedding problem and a set of algorithms to solve them. Finally, a heuristic approach to tackle the computationally expensive task of embedding SFC sets is proposed and evaluated.

II. Related Work

NFV has been proposed as a paradigm that allows more flexible service deployment by leveraging IT virtualization technology in combination with programmable networks [7], [8]. A recent survey on NFV by co-authors of this paper, identifies the decoupling of NFs from hardware, flexible NF deployment and dynamic scaling as the main differences between network service provisioning in NFV compared to current practice [9].

Affinity and anti-affinity restrictions have previously been studied in the context of grid and cloud computing. Many argued that the lack of influence on the placement of workflow or service components is a hindrance for the adoption of the technology [10], [11]. Even though performance and economical benefits of cloud computing are clear, potential users hesitate to deploy the technology because legal, privacy, efficiency and resilience aspects are completely out of their control. Also, recent media coverage shows an increased concern by end-users, companies and governments about their data privacy, raising the need for SPs to take into account these issues when deploying and offering their services[1,2]. These concerns also arise for NFV when deploying VNFs at certain locations and transferring data between them over virtual paths. Therefore, we argue that also in NFV, mechanisms should be designed to allow SPs to add constraints concerning locality and affinity, both to VNFs as well as the interconnecting paths.

The solutions proposed in the affinity and anti-affinity context for cloud computing mostly relate to two aspects: developing models to describe affinity rules and developing service placement algorithms that can work under the constraints of these rules. *Konstanteli et al.* present a set of affinity rules for cloud computing applications which are added to a Mixed-Integer Non-Linear Programming (MINLP) [12]. The authors define constraints that require allocating components and services in the same subnet or physical node, or prevent services from being federated. *Larsson et al.* and *Espling et al.* propose a model for defining Virtual Machine (VM) placement in cloud computing supporting a set of affinity and anti-affinity constraints [13], [14]. We extend this approach by defining affinity and anti-affinity restrictions for SFCs. To this end we add support for specification of constraints on the path between network functions and furthermore, a more expressive syntax that allows constraints to apply to specific VNFs, VNF types, locations and location types. A semantic framework is proposed which allows to check the validity of these constraints.

One of the benefits of NFV is that it supports automated deployment and orchestration of services. To achieve this, a number of descriptions are needed for everything that was configured manually in the past, including VNFs and network requirements [3]. Also, Service Level Agreement (SLA)-related parameters such as affinity and anti-affinity rules should be transformed into machine-readable description formats [15]. Huawei mentions the generation of affinity and anti-affinity policies as a mechanism for fault prevention [16]. In the definition of *Service Quality Metrics* by ETSI, special

[1]No Safe Harbor: How NSA Spying Undermined U.S. Tech and Europeans' Privacy – https://www.eff.org/nl/deeplinks/2015/10/europes-court-justice-nsa-surveilance

[2]Facebook case may force European firms to change data storage practices – http://www.theguardian.com/us-news/2015/sep/23/us-intelligence-services-surveillance-privacy

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2017.2681025, IEEE Transactions on Network and Service Management

attention is brought to the enforcement of NFV customer anti-affinity rules which can improve the availability mechanisms [17]. The automatically generated affinity rules for VNFs in combination with user-specific affinity requirements could lead to conflicting constraints. In this paper, a machine-readable format for affinity and anti-affinity constraints is proposed. Furthermore, we establish an automated way to detect conflicting constraints based on ontologies. The proposed conflict detection is applicable to both user-generated as well as automatically generated affinity constraint sets. Furthermore, the proposed framework can also be used to validate SFC embeddings with respect to the imposed affinity requirements.

To attain the gains promised by NFV, the VNFs and interconnecting virtual links should be efficiently mapped onto the physical substrate. To achieve this, several placement algorithms have been proposed in the related fields of Virtual Network Embedding (VNE) [6] and Virtual Data Center Network Embedding (VDCNE) [18], [19], as well as for NFV [9].

In the field of NFV, *Mehraghdam et al.* apply Mixed Integer Quadratically Constrained Programming (MIQCP) to solve the placement problem and conclude that to obtain efficient use of resources, the placement of functions should be different according to the desired objective [20]. *Beck et al.* propose a coordinated allocation heuristic based on the backtracking concept in which they take into account that the structure of SFCs can be flexible [21]. This allows to reorder some of the VNFs in coordination with the allocation of the SFC. Also *Mehraghdam et al.* take into account such flexible structures [22]. However, they propose a two-step approach without coordination between the SFC composition and embedding. *Moens et al.* propose an Integer Linear Programming (ILP) based solution in which hybrid scenarios are considered where part of the functions are provided by dedicated physical hardware and part of them by virtualized instances [23]. *Luizellie et al.* formalize the NF placement and chaining problem and propose an ILP model to solve it [24]. *Xia et al.* propose a Binary Integer Programming (BIP) model for optimal VNF placement subject to minimizing the expensive optical/electronic/optical conversions for NFV chaining in packet/optical data centers [25]. *Sahhaf et al.* propose an algorithm for mapping SFCs to the network infrastructure while taking into account decompositions of NFs [26]. *Yoshida et al.* propose a multi-objective resource scheduling algorithm simultaneously optimizing possibly conflicting objectives with multifaceted constraints [27]. *Vaishnavi et al.* tackle the problem of inter-domain virtual network provisioning by abstracting the resources of a domain to appear as a single node [28]. In previous work, we formulated the online virtual function mapping and scheduling problem and proposed a set of algorithms for solving it [29]. Other research focuses on specific NFV uses cases such as virtualizing the CPE, Evolved Packet Core (EPC) or Deep Packet Inspection (DPI) [30], [31], [32], [33]. For an extended overview of resource allocation in NFV, we refer to a survey by *Herrera et al.* [34]. None of the aforementioned approaches offers support for attaching affinity or anti-affinity constraints to the SFCs nor do they take into account such constraints when evaluating the embeddings.

This paper is an extension of previous work [35] in which the affinity and anti-affinity constraints were first proposed. The previous paper proposed a first version of the semantic framework for affinity-constrained SFCs validation. In the current paper, the aforementioned framework was further fine-tuned and extended with embedding algorithms for affinity-constrained SFCs. Furthermore, using the mathematical model proposed in this paper, the semantic filtering is validated. Also the impact of applying semantic filtering prior to the embedding is evaluated in this paper.

III. Affinity and Anti-Affinity Constraints

Currently, in an NFV context, SPs have limited control over the mapping of VNFs to physical hosts or SFC edges to physical paths. Nevertheless, many situations can be envisioned where an SP might want to attach constraints to the placement of certain functions or on the routing of traffic, such as:

- **Efficiency:** VNFs that exchange a lot of data may want to be positioned close to one another (e.g., within the same datacenter, or even on the same physical host).
- **Resilience:** The SP might want to spread instances of the same VNF across multiple datacenters in order to improve resilience in case a failure occurs in one of the datacenters.
- **Legislation:** The SP might want to avoid hosting VNFs in certain countries due to legislative restrictions on the location of the data that is processed or transferred.
- **Privacy:** SPs or their customers might not want the traffic to pass through certain domains due to privacy concerns.
- **Economic:** SPs might have economic reasons (e.g., peering agreements) to place their functions in or route their traffic through certain domains.

However, currently there is no way to specify or model such requirements in an SFC template. In this section, the set of affinity and anti-affinity constraints for VNFs and their interconnecting paths are discussed. The affinity constraints apply to a set of physical locations P, a set of VNF instances V and a set of edges E interconnecting them. There are different location granularities $g \in G$ that can be considered (e.g., countries, network domains, datacenters), leading to a hierarchical structure of locations. Two hosts in a single datacenter represent different locations at the granularity of hosts, but have the same location at the granularity of datacenters. $P^g \subseteq P$ is the set of locations at a certain granularity g. Furthermore, each VNF instance has an associated VNF type (e.g., firewall, DPI), forming subsets $V^t \subseteq V$ of VNFs with type $t \in T$. Finally, each virtual edge $e = (a, b) \in E$ connects two VNFs $a \in V$ and $b \in V$ and maps to a single (or path of) physical network links. We propose and define four groups of constraints which model location and colocation constraints, both for VNFs and virtual edges, each consisting of pairs of affinity and anti-affinity constraints. Figure 1 shows a graphical illustration of a subset of the constraints defined below:

- VNF Location Constraints:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2017.2681025, IEEE Transactions on Network and Service Management

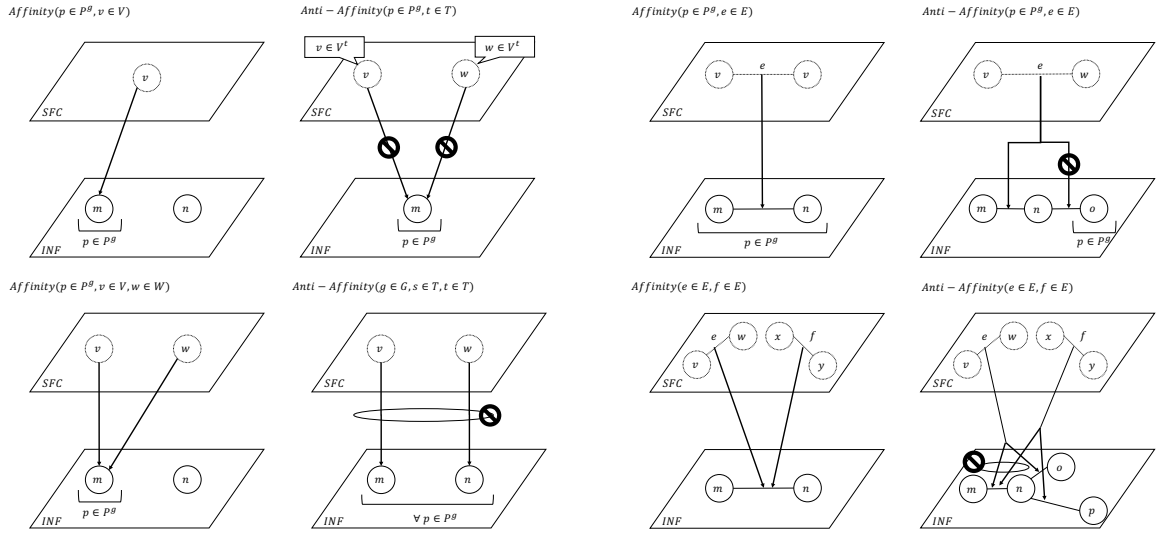

Fig. 1: Graphical illustration of the various affinity and anti-affinity constraints

- ∗ Affinity($p \in P^g, v \in V$ or $t \in T$): a specific instance v or all instances $v \in V^t$ of type $t \in T$ must be located at a specific location p with granularity g.
- ∗ Anti-Affinity($p \in P^g, v \in V$ or $t \in T$): a specific instance v or all instances $v \in V^t$ of type $t \in T$ must not be located at a specific location p with granularity g.

- VNF Colocation Constraints:
  - ∗ Affinity($p \in P^g$ or $g \in G, v \in V$ or $s \in T, w \in V$ or $t \in T$): a specific instance v or all instances $v \in V^s$ of type $s \in T$ must be placed together with a specific instance w or all instances $w \in V^t$ of type $t \in T$ at a specific location $p \in P^g$ or at the same location at a specific granularity $g \in G$.
  - ∗ Anti-Affinity($p \in P^g$ or $g \in G, v \in V$ or $s \in T, w \in V$ or $t \in T$): a specific instance v or any instances $v \in V^s$ of type $s \in T$ must not be placed together with a specific instance w or any instances $w \in V^t$ of type $t \in T$ at a specific location $p \in P^g$ or at the same location at a specific granularity $g \in G$.
- Virtual Edge Location Constraints:
  - ∗ Affinity($p \in P^g, e \in E$): a virtual edge $e \in E$ must be fully embedded at a specific location $p \in P^g$ with a granularity $g \in G$. It does not suffice to model this as location constraints on the endpoints, since physical nodes along the virtual path can still be located in other locations $q \in P^g \setminus \{p\}$.
  - ∗ Anti-Affinity($p \in P^g, e \in E$): the physical links comprising the virtual edge $e \in E$ must not pass through a specific location $p \in P^g$ with a granularity $g \in G$.
- Virtual Edge Colocation Constraints:
  - ∗ Affinity($e \in E, f \in E$): two virtual edges $e \in E$ and $f \in E$ must overlap (i.e. all physical links comprising the edge e must be the same as those comprising edge f).
  - ∗ Anti-Affinity($e \in E, f \in E$): two virtual edges $e \in E$ and $f \in E$ must not overlap (i.e. none of the physical links comprising the virtual edges shall be part of both e and f). Modelling this as an anti-affinity constraint for the VNFs of the endpoints of e and f does not achieve this, since the physical paths can still have overlapping links even though the endpoints are mapped to distinct physical nodes.

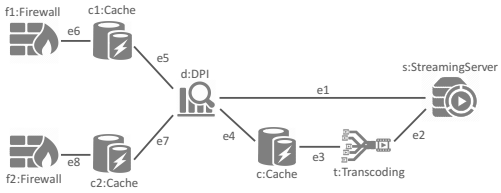

Fig. 2: An example SFC.

To further clarify the presented constraint formulations, an example of an SFC request with both affinity and anti-affinity constraints is given here. Given a set of location types {*Autonomous System (AS), Datacenter (DC), Host*} and a set of network function types {*Firewall, DPI, Cache, Transcoding, Streaming Server*}. An example SFC is depicted in Figure 2, where a *Streaming Server* is connected via a DPI function (e.g., for tagging data packets) to a content cache. The DPI functions may either directly forward requests to the streaming server or may send them via the *Transcoder* for transcoding and packaging (e.g., for delivery to mobile devices). The transcoding is preceded by a *Cache* in the chain to store the transcoded content. The end-users are connected via a firewall to the service. Suppose a SP wants to offer a Video on Demand (VoD) service in Belgium where two infrastructure providers are active: Telenet (*AS6848*) and Proximus (*AS6774*). Further-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2017.2681025, IEEE Transactions on Network and Service Management

more, the SP wants to deploy part of the service in an Amazon DC identified by *DC_AWS*. Let us consider the following set of affinity and anti-affinity constraints:

- *Affinity(AS6848, c1)*
- *Affinity(AS6774, c2)*
- *Affinity(DC, Transcoder, Streaming Server)*
- *Affinity(DC_AWS, e2)*
- *Affinity(Host, t, c)*
- *Anti-Affinity(e1, e3)*

Specifically, the first two constraints state that the *Caches c1* and *c2* need to be located in the Telenet and Proximus AS respectively (e.g., because they should be close to the end user and limit uplink traffic through other networks). The third constraint states that, for efficiency reasons and for reducing the network traffic, all *Transcoder* functions should be colocated with the *Streaming Server* functions. The next constraint states that the virtual edge *e2* should be fully embedded within the scope of the *DC* identified by *DC_AWS*. The fifth constraint assures that the *Transcoder t* and the *Cache c* should be located at the same *Host*. Finally, the last constraints dictates that none of the physical links that are used for the embedding of the virtual edges *e1* and *e3* are allowed to overlap.

IV. Semantic SFC Request Checker

Since SPs are now free to specify their custom constraints during the SFC request, it is possible that conflicting constraints are introduced. Extending the example from the previous section and adding the constraints *Affinity(Host, c1, f1)* (i.e. specifying that *c1* and *f1* should be colocated in the same Host) and *AntiAffinity(DC, Cache, Firewall)* (i.e. specifying that a VNF of type Cache cannot be colocated with a VNF of type Firewall in the scope of a *DC*) leads to a conflicting constraint set. Also more complex conflicts can appear when multiple constraints are involved in the conflicting set that can only be detected as a conflict when considering the full set. For example, returning to the base example from the previous section and adding the constraints *Affinity(DC, DPI, Cache)* and *Anti-Affinity(DC_AWS, d)* would lead to a conflict set {*Affinity(Host, t, c), Anti-Affinity(DC_AWS, d), Affinity(DC_AWS, e2), Affinity(DC, DPI, Cache)*}. Since *d* and *c* should be colocated in the same *DC* due to the VNF type affinity constraint and *t* and *c* are colocated at the *Host* level, *d* and *t* should be colocated at the *DC* level as well. Furthermore, since *e2* should be fully embedded in *DC_AWS*, *t* should also be located in *DC_AWS*. This means that *d* should be located in *DC_AWS* as well. However, this inferred knowledge conflicts with the defined constraint *Anti-Affinity(DC_AWS, d)*.

When the VNFInP tries to deploy the requested SFC, none of the resulting embedding configurations will lead to a feasible realisation of the SFC request. The VNFInP should however be able to differentiate between a non-acceptance of the SFC request caused by a shortage of appropriate resources and conflicting request constraints, in order to inform the SP on the reason why the SFC deployment failed. The previous example shows the need for the VNFInP to check the validity of an SFC request upon reception in order to exclude any conflicting constraints when trying to provision the requested SFC.

In the proposed architecture, the *SFC Embedding Algorithm* (which will be discussed in the next section) is responsible for assigning resources to the SFC requests. Concretely, it decides on which VNFs should be deployed on which hosts and how many resources should be assigned to them. The *Cloud Manager* performs the management of deployed VNFs and server resources. Moreover, the algorithm selects the forwarding paths interconnecting the VNFs and assigns network resources to them through the *SDN Controller*. Before the SFC request is forwarded to the *SFC Embedding Algorithm* it needs to be checked by the *SFC Request Checker* to confirm the validity.

In previous work we proposed to exploit ontology representations for the purpose of modeling the physical substrate, the SFC requests and defining a set of rules that can be used to infer additional information [35]. Figure 3 represents the proposed semantic model. The SFC requests are modeled as a set of *VNFs* with a certain *VNFType* and *VirtualEdges* containing an ingress and egress *VNF*. The physical resources are modeled at the granularity level of *Hosts*, *DCs* and *ASs*. Each of these *Locations* has a certain *LocationType* (i.e., AS, DC or Host). The hierarchical relations between these *Locations* and *LocationTypes* are modeled by isSubLocationOf and isSubLocationTypeOf respectively. To model affinity (respectively anti-affinity) constraints for single VNFs and edges, positive (respectively negative) object property assertions of the type isVNFEmbeddedOn, isEdgeEmbeddedOn and isEdgeFullyEmbeddedOn are attached to *VNFs* and *VirtualEdges*.

To be able to model more complex affinity and anti-affinity relationships between two *VNFs*, two *VNFTypes* or between a *VNF* and *VNFType*, the additional concepts *VNFVNFRestriction*, *VNFVNFTypeRestriction* and *VNFTypeVNFTypeRestriction* were added to the ontology. By adding the respective positive (respectively negative) property isVNFVNFEmbeddedOn or isVNFVNFEmbeddedOnType, one is able to model affinity (respectively anti-affinity) restrictions for more complex constraints on the *Location* or *LocationType*. By using the isEdgeEmbeddedWithEdge relationship, affinity and anti-affinity constraints between edges can be modeled.

To infer new information out of existing knowledge, a set of rules is defined. For example, Rule (1) stipulates that if a certain *VNF x* is embedded on a *Location y* and if *y* is a sublocation of *z*, this *VNF* is also embedded on *Location z*. When a *VNFType y* is embedded on a *Location z*, each *VNF x* of that *VNFType y* needs to be embedded at the *Location z* (Rule (2)). If a *VirtualEdge z* contains a *VNF x* embedded at *Location a*, this *VNF z* is embedded at *Location a* as well (Rule (3)). For a full description of the rules, we refer to previous work [35] which focuses on the semantic validation of SFC requests.

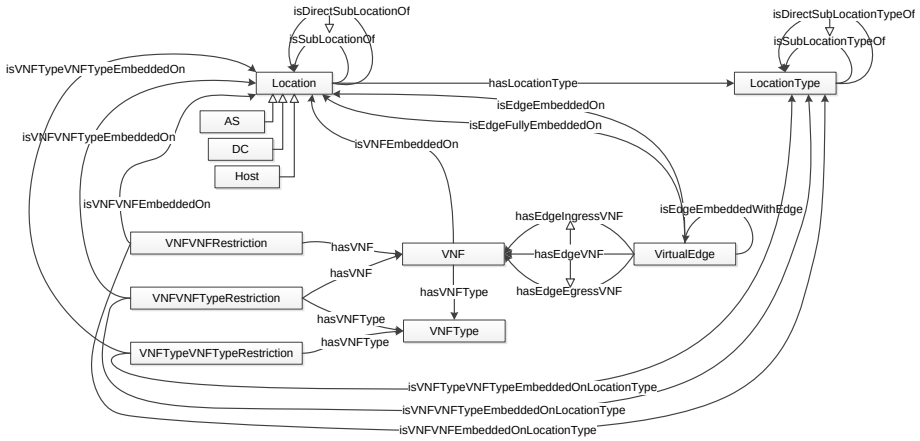

Fig. 3: Graphical representation of ontology.

$$isVNFEmbeddedOn(x, y) \wedge isSubLocOf(y, z) \rightarrow isVNFEmbeddedOn(x, z) \quad (1)$$

$$hasVNFType(x, y) \wedge isVNFTypeEmbeddedOn(y, z) \rightarrow isVNFEmbeddedOn(x, z) \quad (2)$$

$$hasEdgeVNF(z, x) \wedge isVNFEmbeddedOn(x, a) \rightarrow isEdgeEmbeddedOn(z, a) \quad (3)$$

When a new SFC request arrives at the VNFInP, this request is parsed and the set of VNFs and edges are added as individuals to the Web Ontology Language (OWL) ontology. Next, the set of affinity and anti-affinity constraints are also added by either creating new individuals (i.e. VNFVNFRestriction), adding property assertions (i.e. isVNFEmbeddedOn) or both. For example, the affinity restriction *Affinity(AS6848, c1)* concerning *VNF c1* and *AS AS6848*, is modeled as a positive property assertion $\mathrm{isVNFEmbeddedOn(c1, AS6848)}$. On the other hand, the negative restriction *Anti-Affinity(e1, e3)* concerning *VirtualEdge e1* and *VirtualEdge e3*, is modeled as a negative property assertion $\mathrm{isEdgeEmbeddedWithEdge(e1, e3)}$.

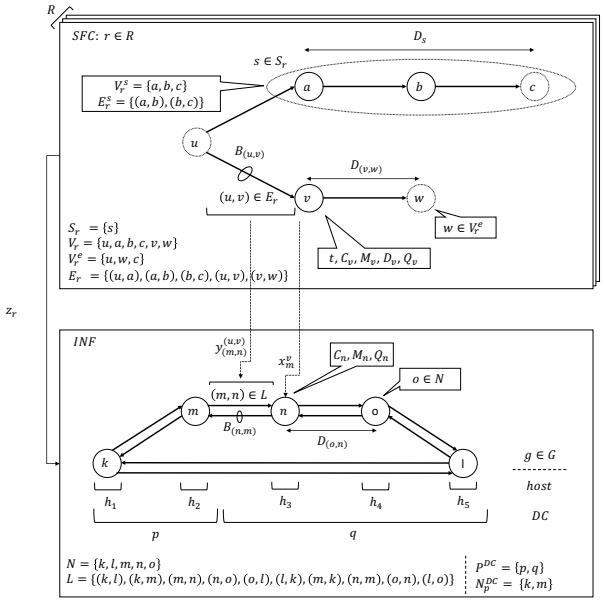

Fig. 4: Graphical representation of the SFC placement model.

V. Model

In this section the affinity-constrained SFC placement problem is described. First, the notations for the inputs and variables used throughout the chapter are presented. Second, the general constraints involved in SFC placement are introduced. Third, the constraints related to affinity and anti-affinity based placement restrictions are explained. Finally, the placement objectives are introduced. Figure 4 shows a graphical representation of the variables and their relationships.

A. NFV Infrastructure model

The topology of an NFV infrastructure can be described as a weighted directed graph $INF = \{N, L\}$, where N is the set of substrate nodes in the infrastructure and L the set of links interconnecting them. Each substrate node $n \in N$ represents a location where a network function could be deployed. Each substrate node n is characterized by its available processing capacity C_n, available memory capacity M_n and its location $p \in P$. This geographic location represents a hierarchical structure where each level of this structure represents a certain geographic granularity (e.g., host, rack, datacenter, network domain, etc.). The subset of geographic locations $P^g \subseteq P$ represents the set of locations at a particular granularity $g \in G$ (e.g., P^{DC} represents the set of all datacenter locations). Similarly, N_p^g represents the set of all substrate nodes at a geographic location $p \in P^g$ at a certain granularity g (e.g., N_p^{DC} represents the set of all substrate nodes located at a datacenter p). It is possible that a certain network node n is not involved directly in mapping VNFs for an SFC but is along the path between VNFs and thus participates in forwarding the traffic for the SFC. Therefore we need to take into account the packet processing capacity of such a node: Q_n when deploying

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2017.2681025, IEEE Transactions on Network and Service Management

SFCs on the substrate network.

Each link $(m,n) \in L$ represents a unidirectional link between two network nodes $m,n \in N$. Bidirectional links are represented as a pair of unidirectional links (i.e. $(m,n),(n,m)$). This allows us to model a variety of network links with different up/download characteristics. Each link is characterized by an available bandwidth capacity $B_{(m,n)}$ and latency $D_{(m,n)}$. Similar to network nodes, also the links are characterized by a geographic location which is the set of locations of their endpoints $\{P_m, P_n\}$.

B. SFC model

Similarly to the VNF infrastructure model, an SFC request $r \in R$ can be described as a weighted directed graph $SFC_r = \{V_r, E_r\}$, where V_r is the set of VNFs and E_r the set of edges interconnecting them. An SFC typically has two or multiple endpoint VNFs which terminate the service chain. These are modeled as endpoint-VNFs $V_r^e \subseteq V_r$. T_r represents the set of VNF types (e.g., firewall, gateway, router). Each VNF $v \in V_r$ has an associated VNF type $t \in T_r$. V_r^t forms the set of VNFs in the SFC that have a type t associated with them. Each VNF v requires a certain processing capacity C_v and a specific memory capacity M_v. Furthermore, each VNF v has an associated processing delay D_v.

Each edge $(u,v) \in E_r$ represents a unidirectional virtual edge between two VNFs $u,v \in V_r$. These are modeled as unidirectional connections since SFCs can have different capacity requirements for up-and downstream respectively (e.g., a video streaming SFC will have higher downstream requirements). An SFC edge can be composed of one or more substrate links or could be an internal link when both u and v are mapped onto the same substrate node n. Link aggregation, where a virtual edge is composed out of multiple distinct underlying paths, is not considered here. Each edge has a required capacity $B_{(u,v)}$ between the VNFs, this capacity is also used to account for the packet processing requirements of the forwarding nodes (i.e. Q_n). A section or path $s \in S_r$ is a tuple composed of a set of ordered VNFs V_r^s and the ordered set of consecutive virtual edges interconnecting these VNFs E_r^s. An SFC request r can associate a maximum allowed latency D_s for the end-to-end flow processing of such sections. A_r denotes the affinity and anti-affinity constraints that are defined in the SFC request r. A_r^a and A_r^{aa} denote the set of affinity constraint and anti-affinity constraints respectively.

C. Assignment variables

For every SFC request $r \in R$, a set of node and link mapping variables is defined. The binary variable z_r defines whether the SFC $r \in R$ is mapped or not as defined in Expression (4). The binary variable x is used in the mapping of the VNFs to network nodes and is defined in Expression (5), where $x_n^v \rightarrow N \cdot V_r$. Similarly, the binary variable y is used as defined in Expression (6), where $y_{(m,n)}^{(u,v)} \rightarrow L \cdot E_r$.

$$z_r = \begin{cases} 1, & \text{if } r \in R \text{ is mapped.} \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

$$\forall r \in R : x_n^v = \begin{cases} 1, & \text{if } v \in V_r \text{ is mapped to } n \in N. \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

$$\forall r \in R : y_{(m,n)}^{(u,v)} = \begin{cases} 1, & \text{if } (u,v) \in E_r \text{ is} \\ & \text{mapped to } (m,n) \in L. \\ 0, & \text{otherwise.} \end{cases} \tag{6}$$

D. General constraints

In order to assure a correct mapping of the requested SFC, a set of constraints can be defined. Equations (7) and (8) prevent that VNFs and virtual edges respectively are mapped when the corresponding SFC to which they belong is not mapped. Equation (9) ensures that if an SFC request r is mapped ($z_r = 1$), each VNF is assigned and that it is assigned to exactly one node. On the other hand, if the SFC request r is not mapped ($z_r = 0$), none of the VNFs should be mapped. Equations (10) and (11) respectively assure that the available CPU and memory capacity of each individual node is not exceeded by the capacity requirements of all VNFs $v \in V$ in the request set R ($V = \bigcup_{r \in R} V_r$).

$$\forall r \in R : \forall v \in V_r : \forall n \in N : x_n^v \leq z_r \tag{7}$$

$$\forall r \in R : \forall (u,v) \in E_r : \forall (m,n) \in L : y_{(m,n)}^{(u,v)} \leq z_r \tag{8}$$

$$\forall r \in R : \forall v \in V_r : \sum_{n \in N} x_n^v = z_r \tag{9}$$

$$\forall n \in N : \sum_{v \in V} (x_n^v \cdot C_v) \leq C_n \tag{10}$$

$$\forall n \in N : \sum_{v \in V} (x_n^v \cdot M_v) \leq M_n \tag{11}$$

$$\forall (u,v) \in E, m \in N : \sum_{n \in N} y_{(m,n)}^{(u,v)} - \sum_{n \in N} y_{(n,m)}^{(u,v)} = x_m^u - x_m^v \tag{12}$$

$$\forall (m,n) \in L : \sum_{(u,v) \in E} \left(y_{(m,n)}^{(u,v)} \cdot B_{(u,v)} \right) \leq B_{(m,n)} \tag{13}$$

$$\forall n \in N : \sum_{u \in V} (x_n^u \cdot Q_u) + \sum_{(u,v) \in E} \left(2 \cdot x_n^u \cdot x_n^v \cdot B_{(u,v)} \right) + \sum_{(u,v) \in E} \sum_{\substack{(m,n) \in L \\ (n,o) \in L}} \left(\left(y_{(m,n)}^{(u,v)} + y_{(n,o)}^{(u,v)} \right) \cdot B_{(u,v)} \right) \leq Q_n \tag{14}$$

$$\forall r \in R : \forall s \in S_r : \sum_{u \in V_r^s} \sum_{n \in N} (x_n^u \cdot D_u) + \sum_{(u,v) \in E_r^s} \sum_{(m,n) \in L} \left(y_{(m,n)}^{(u,v)} \cdot D_{(m,n)} \right) \leq D_s \tag{15}$$

The multi-commodity flow conservation is assured by Equation (12). This ensures that there exist virtual paths between the requested VNFs. Modeling that no capacity is required for a certain virtual edge (u,v), while a virtual edge (v,u) exists with non-zero capacity requirements can be achieved

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2017.2681025, IEEE Transactions on Network and Service Management

by setting $B_{(u,v)} = 0$ and $B_{(v,u)} > 0$ respectively. Using the constraint defined in Equation (13), we ensure that each substrate link has enough capacity to serve all virtual paths $(u, v) \in R$ mapped on it for a request set R ($E = \bigcup_{r \in R} E_r$). Equation (14) guarantees that the required processing capacity is available for each of the substrate nodes. The first term sums up the required capacity B_u by each VNF $u \in V$. The second term assures that if a virtual edge is mapped internally within a single substrate node, the required processing capacity for this edge is added. The last term sums up the processing capacity of each ingress and egress links of a substrate node. Finally, Equation (15) ensures that the end-to-end latency incurred by the substrate links comprising the virtual paths $s \in S_r$ and the processing delays at the VNFs adhere to the maximum allowed latency D_s specified by the SFC request $r \in R$.

E. Affinity and Anti-Affinity constraints

To take into account the required geographic and colocation requirements for a specific VNF request r, these are modeled as constraints for the optimization problem. Equation (16) stipulates that all VNFs of a certain type t need to be located at a location p with granularity g, while Equation (17) prevents the embedding of any VNF of type t at this location p. To model the affinity of certain VNF types s and t at a any location with granularity g, Equation (18) guarantees that every VNF v of type s is embedded with all VNFs of type t (=$|V^t|$) at a single location p with granularity g. Equation (19) prevents a VNF of type s to be embedded with a VNF of type t at the same location p with granularity g. Similar constraints were added for the other affinity and anti-affinity constraints involving VNFs and VNF types defined in Section III.

Next to VNF affinity constraints, link affinity constraints are also modeled. Equation (20) assures that every substrate link (m, n) on which a virtual edge (v, w) is mapped, is located at a location $p \in P^g$. Furthermore, since edges can be mapped internally (i.e. both VNFs v and w are mapped onto the same substrate node), Equation (21) ensures that also the VNFs v and w are mapped onto a host at that location $n \in N_p^g$. To prevent the mapping virtual edges (u, v) at a certain location $p \in P^g$, Equation (22) ensures that none of the substrate links (m, n) for which one of the endpoints is located at that location ($m \vee n \in N_p^g$) is used. Similarly, Equation (23 prevents the internal embedding of the virtual edge (v, w) at that location.

Equation (25) stipulates that two virtual edges need to be mapped onto the same virtual path. Equation (24) ensures that the corresponding endpoints of the virtual edge are mapped onto the same host. This constraint takes care of the cases where the virtual edges are mapped internally. Equation (26) prevents two virtual edge mappings of sharing a link. Furthermore, none of the endpoints of these respective virtual edges should be colocated at the same host (Equation (27))

$$\text{Affinity}(p \in P^g, t \in T_r): \quad \forall v \in V_r^t : \sum_{n \in N_p^g} x_n^v = z_r \tag{16}$$

$$\text{Anti-Affinity}(p \in P^g, t \in T_r): \quad \forall v \in V_r^t : \sum_{n \in N_p^g} x_n^v = 0 \tag{17}$$

$$\text{Affinity}(g \in G, s \in T_r, t \in T_r): \quad \forall v \in V_r^s : \forall p \in P^g : \sum_{n \in N_p^g} \left(x_n^v \cdot \left|V_r^t\right|\right) = \sum_{w \in V_r^t} \sum_{n \in N_p^g} x_n^w \tag{18}$$

$$\text{Anti-Affinity}(g \in G, s \in T_r, t \in T_r): \quad \sum_{p \in P^g} \left(\sum_{n \in N_p^g} \sum_{v \in V_r^s} x_n^v \cdot \left(\sum_{n \in N_p^g} \sum_{w \in V_r^s} x_n^w \right) \right) = 0 \tag{19}$$

$$\text{Affinity}(p \in P^g, e = (v, w) \in E_r): \quad \sum_{\substack{(m,n) \in L \\ m \wedge n \in N_p^g}} y_{(m,n)}^{(v,w)} \geq \sum_{(m,n) \in L} y_{(m,n)}^{(v,w)} \tag{20}$$

$$\left(\sum_{n \in N_p^g} x_n^v = z_r \right) \wedge \left(\sum_{n \in N_p^g} x_n^w = z_r \right) \tag{21}$$

$$\text{Anti-Affinity}(p \in P^g, e = (v, w) \in E_r): \quad \sum_{\substack{(m,n) \in L \\ m \vee n \in N_p^g}} y_{(m,n)}^{(v,w)} = 0 \tag{22}$$

$$\left(\sum_{n \in N_p^g} x_n^v = 0 \right) \wedge \left(\sum_{n \in N_p^g} x_n^w = 0 \right) \tag{23}$$

$$\text{Affinity}(e = (v, w) \in E_r, f = (u, z) \in E_r): \quad \left(\sum_{n \in N} (x_n^v \cdot x_n^u) = z_r \right) \wedge \left(\sum_{n \in N} (x_n^w \cdot x_n^z) = z_r \right) \tag{24}$$

$$\begin{gathered} \sum_{(m,n) \in L} \left(y_{(m,n)}^{(v,w)} \cdot y_{(m,n)}^{(u,z)} \right) \geq \sum_{(m,n) \in L} y_{(m,n)}^{(v,w)} \\ \sum_{(m,n) \in L} \left(y_{(m,n)}^{(v,w)} \cdot y_{(m,n)}^{(u,z)} \right) \geq \sum_{(m,n) \in L} y_{(m,n)}^{(u,z)} \end{gathered} \tag{25}$$

$$\text{Anti-Affinity}(e = (v, w) \in E_r, f = (u, z) \in E_r): \quad \sum_{(m,n) \in L} \left(y_{(m,n)}^{(v,w)} \cdot y_{(m,n)}^{(u,z)} \right) = 0 \tag{26}$$

$$\begin{gathered} \left(\sum_{n \in N} (x_n^v \cdot x_n^u) = 0 \right) \wedge \left(\sum_{n \in N} (x_n^w \cdot x_n^z) = 0 \right) \\ \left(\sum_{n \in N} (x_n^v \cdot x_n^z) = 0 \right) \wedge \left(\sum_{n \in N} (x_n^w \cdot x_n^u) = 0 \right) \end{gathered} \tag{27}$$

F. Objective functions

When embedding SFC request set R, there are multiple objectives that could be considered. To maximize the revenue of the VNFInP, the number of accepted SFC requests should be maximized as shown in Equation (28). On the other hand, to efficiently manage the infrastructure resources, the InP could have other objectives, such as minimizing the number

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2017.2681025, IEEE Transactions on Network and Service Management

of substrate nodes and links that are used, minimizing the overall traffic or load balancing the traffic over the full infrastructure. To achieve this, the problem is first solved by using the acceptance maximization as an objective function. Afterwards, the solution Z_{sol}, which characterizes the number of embedded SFC requests is added as an additional constraint by Equation (29).

$$\max \sum_{r \in R} z_r \tag{28}$$

$$\sum_{r \in R} z_r \geq Z_{sol} \tag{29}$$

Afterwards the adapted optimization problem is solved again, this time with the objective specified by the InP. Equation (30) minimizes the number of substrate nodes $n \in N$ that are used for embedding the requests. The total bandwidth consumption is minimized by the objective defined in Equation (31). Similar objectives can be defined for other resource types as well.

$$\min \sum_{r \in R} \sum_{v \in V_r} \sum_{n \in N} x_n^v \tag{30}$$

$$\min \sum_{r \in R} \sum_{(u,v) \in E_r} \sum_{(m,n) \in L} \left(y_{(m,n)}^{(u,v)} \cdot B_{(u,v)} \right) \tag{31}$$

To balance the bandwidth consumption over the full infrastructure, the difference between the maximum and minimum load over all links is minimized as shown in Equation (32). To model this, two additional continuous decision variables $Load_{min} \in [0.0, 1.0]$ and $Load_{max} \in [0.0, 1.0]$ are added, as well as constraints stating that the load of each link should be smaller, respectively larger, than the maximum and minimum load. The objective is then to minimize the difference $Load_{max} - Load_{min}$. The same could be achieved with other resources than bandwidth as well.

$$\min \left(\max_{(m,n) \in L} \sum_{r \in R} \sum_{(u,v) \in E_r} \left(\frac{y_{(m,n)}^{(u,v)} \cdot B_{(u,v)}}{B_{(m,n)}} \right) - \min_{(m,n) \in L} \sum_{r \in R} \sum_{(u,v) \in E_r} \left(\frac{y_{(m,n)}^{(u,v)} \cdot B_{(u,v)}}{B_{(m,n)}} \right) \right) \tag{32}$$

VI. Heuristic approach

Solving the SFC embedding problem can be computationally expensive, as will be shown later on during the evaluations. Therefore, a heuristic approach is proposed in which the SFCs are ordered according to certain criteria and the embedding algorithm proceeds to map them individually to the infrastructure taking into account different objective functions. The mathematical model that was presented before was adapted to be able to support such individual mapping of ordered SFC sets. To this end, the decision variable z_r, indicating if SFC request $r \in R$ should be mapped is left out of the model. This ensures that the optimization process tries to embed the request r if there exists a feasible solution. Similar objectives can be defined as before, however also the resource usage incurred by previous individual embeddings should now be taken into account when evaluating the objective functions.

The ordering of the SFCs in the set can be achieved in many ways. A first ordering criterion could be to order the SFCs based on the number of affinity constraints that they contain, resulting in the set R_{nc} (Equation (33)). Another option is to order the SFC requests based on the requested bandwidth resources as shown in Equation (34).

$$R_{nc} : \left\{ r \in R, i \leq j : |A_{r_i}| \leq \left|A_{r_j}\right| \right\} \tag{33}$$

$$R_{bw} : \left\{ r \in R, i \leq j : \sum_{(u,v) \in E_i} B_{(u,v)} \leq \sum_{(u,v) \in E_j} B_{(u,v)} \right\} \tag{34}$$

VII. Evaluation

In this section, the simulation framework that was developed to evaluate the affinity-constrained SFC embedding is discussed. The implementation details of the semantic validation module are discussed, as well as the implementation of the mathematical model. Furthermore, the generation of substrate topologies and SFCs is discussed. The first set of experiments evaluates the impact of the infrastructure size, the requested SFC size and the number of affinity and anti-affinity constraints on the scalability of the semantic validation. Afterwards, the performance gain of the semantic validation on the SFC embedding is discussed.

A. Simulation Framework

The simulation framework is implemented in Java 8[3] and allows to generate topologies, generate random SFCs, validate these SFCs and map them onto the substrate. The models presented in the previous sections are implemented using CPLEX 12.6[4]. The topologies are generated using the BRITE topology generator[5]. The ASs and their interconnections are generated using BRITE after which a number of DCs are added to each of the ASs. The DC topologies that are generated for the evaluations are two-level fat-tree topologies [36].

The SFCs and their respective constraints are generated randomly. First a set of VNFs and interconnecting virtual edges is generated. The respective VNF types and capacity constraints for both VNFs and edges are uniformly distributed between the configured ranges. Furthermore, end-to-end segments are assigned with a maximum delay, uniformly distributed within the configured ranges. Second, random affinity/anti-affinity constraints are added and the required VNFs, virtual edges and VNF types are randomly selected from the ones that are present in the SFC. When generating these constraints, it is checked whether the same restriction or its counterpart (affinity or anti-affinity restriction with the same parameters) is not already present in the set. Finally, the locations are randomly selected from the respective location sets and added to the

[3]Java 8 – https://java.com/en/download/faq/java8.xml

[4]IBM CPLEX – http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

[5]BRITE – www.cs.bu.edu/brite/

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2017.2681025, IEEE Transactions on Network and Service Management

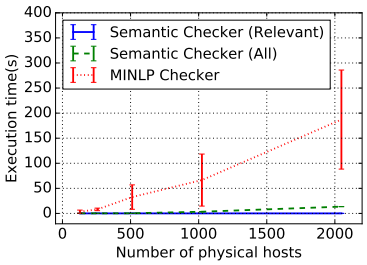

Fig. 5: Impact of the infrastructure size on MINLP based and semantic validation.

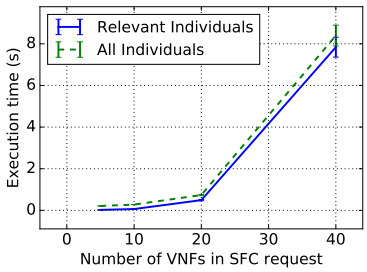

Fig. 6: Impact of the number of requested VNFs on the semantic validation.

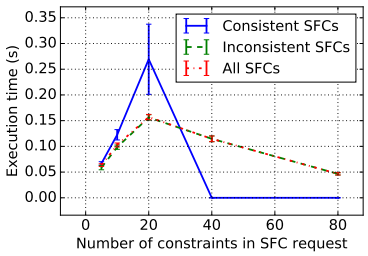

Fig. 7: Impact of the number of constraints per SFC on the semantic validation.

constraints. To select these locations, a discrete probability distribution with the following probability mass function is used for affinity restrictions (AS: 0.6, DC: 0.3, host: 0.1) and anti-affinity restrictions (AS: 0.1, DC: 0.3, host: 0.6). The rationale behind this is that affinity constraints apply to more general location restrictions while for anti-affinity constraints, more granular specification of locations apply. The type of the constraints is uniformly distributed among the constraints defined in Section III. Table I lists the set ($[x, y]$) or ranges ($[x - y]$) for the various parameters that are set during the evaluations.

The Protégé editor[6] was used to develop the SFC request modelling ontology using the OWL API[7]. Semantic Web Rule Language (SWRL)[8] was used to express the aforementioned rules using concepts from the ontology defined in Section IV. The Protégé editor was also used to define the rules using the Manchester syntax[9]. The HermiT OWL Reasoner[10] was used to check the consistency and the classification of the ontology. HermiT is a semantic reasoner for ontologies written in OWL. It is able to determine whether or not the ontology is consistent, identify subsumption relationships between classes, etc. The reasoner is based on a hypertableau calculus which provides efficient reasoning. The output of the reasoning process allows us to determine whether the SFC request at hand is valid or not. In the case of an invalid request, this is communicated to the requesting SP, otherwise the request is passed on to the embedding engine.

The evaluations were carried out using the STEVIN Supercomputer Infrastructure at Ghent University[11]. The nodes are equipped with 2 Intel Xeon CPU E5-2680 v3 12-core processors and 32GB of physical memory. For the experiments, a single 2.5GHz core and 16GB of memory were requested. All evaluations were repeated 20 times with varying seed values, the graphs show the average values and 95% confidence intervals.

[6]Protégé – http://protege.stanford.edu/
[7]OWL2 – http://www.w3.org/TR/owl-features/
[8]SWRL – http://www.w3.org/Submission/SWRL/
[9]Manchester – http://www.w3.org/2007/OWL/wiki/ManchesterSyntax
[10]HermiT OWL Reasoner – http://hermit-reasoner.com
[11]HPC UGent – http://www.ugent.be/hpc

Type	Parameter	Range
BRITE Topology	#AS	[2,3,4,5,8,16,32,64,128,256]
	#Neighbour-AS	[2]
	#AS-Routers	[8,16,32]
	#Neighbour-Routers	[3]
	#DC	[1,2,4]
	Intra-AS BW	[5Gbps ... 50Gbps]
	Inter-AS BW	[1Gbps ... 10Gbps]
	Intra-AS delay	[5ms ... 10ms]
	Inter-AS delay	[1ms ... 5ms]
Fat-tree Topology	#Core switches	[2,3]
	#Pods	[1 ... 4]
	#Servers per pod	[5 ... 10]
	Link BW	[5Mbps ... 10Mbps]
	Link delay	[1ms]
SFC	#VNF	[2,5,10,20,30,40]
	#Affinity constraints	[5,10,20,40,60,80]
	Link BW	[50Mbps ... 500Mbps]
	Segment delay	[5ms ... 100ms]
	Processing delay	[0ms ... 5ms]

TABLE I: Scenario parameters.

B. Scalability of Semantic SFC Validation

The mathematical formulation proposed in Section V can be adapted to validate affinity-constrained SFC requests by removing all resource constraints from the model. However, as indicated by the results in Figure 8, this approach scales poorly with increasing infrastructure sizes. The main reason for this is the increasing number of decision variables and constraints when modeling the validation using MINLP based solution. These experiments use a fixed number of 10 VNFs per SFC and 5 randomly generated affinity and anti-affinity restrictions. Furthermore, the exhaustive search that is performed when solving such a problem leads to execution times that increase exponentially with increasing infrastructure sizes. Modeling the problem using ontologies avoids having to check every possible placement, but instead allows to reason on the various constraints and their mutual impact. Nonetheless, also semantic reasoning times are known to increase exponentially with the number of individuals in the ontology. In Section IV, a semantic SFC request validation framework was discussed which loads both the substrate topology and requested SFC into an ontology. As shown in Figure 8, also the semantic validations suffers from exponentially increasing execution times when the infrastructure size grows. The semantic reasoning times can be significantly reduced when only including the relevant

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2017.2681025, IEEE Transactions on Network and Service Management

parts of the infrastructure in the ontology. To determine these relevant parts, all physical locations of the substrate topology that are included in the SFC constraints are modeled. Furthermore, also the parent locations are subsequently modeled. This set is typically much smaller than the complete infrastructure and can be considered constant for SFCs of the same size. When only loading relevant individuals, execution times show a constant trend at about $62ms$, even when the infrastructure size increases.

Leaving out irrelevant parts of the infrastructure could have an impact on the accuracy and recall of identifying the inconsistent SFCs. Therefore, additional experiments are conducted taking the MINLP based solution as the ground truth and comparing the filtering results with those of the semantic request checker that only models relevant individuals. In the context of SFC validation, a false positive (FP) is an SFC that is considered to be invalid by the framework while in fact it is consistent, while a false negative (FN) is an SFC that is considered to be valid by the framework while in fact it is not. Filtering out valid SFCs, causing a high number of FPs, would be an unacceptable side-effect of the semantic filtering step. The proposed semantic solution based on OWL uses an Open World Assumption (OWA), meaning that in general no single agent or observer has complete knowledge, and therefore limits the kinds of inference and deductions it makes to those that follow from statements that are known to be true. Statements about knowledge that is not included in or inferred from the knowledge explicitly recorded in the system may be considered unknown, rather than wrong or false. Therefore, the semantic SFC checker will not consider an SFC to be invalid based on the absence of information about the topology. And thus, noFPs will be caused by filtering out irrelevant parts of the infrastructure. Validating 1000 SFC requests using the output of the MINLP based solution yields 371 true positives (TPs), 620 true negatives (TNs), 0 FPs and 9 FNs. As expected none of the SFCs are falsely marked as inconsistent by the semantic validation, leading to a precision of 100%. An example of constraints leading to a FN is a request requiring two VNFs to be located in a certain AS, while at the same time requesting them to be embedded in different DCs, when only a single DC is available in that AS. Due to the OWA, the semantic validation framework will not assume that the absence of such a DC in the ontology means that it does not exist, and therefore not mark the SFC as inconsistent.

Figure 6 shows the impact of increasing the requested SFC size on the semantic validation time. As can be expected, the execution time increases exponentially with an increasing number of VNFs per SFC. Validating SFC requests with 20 VNFs takes about 500 ms. Considering that we use a standard Hermit reasoner without optimizations on standard off-the-shelve hardware, this could be considered a reasonable processing overhead. As can be seen from the graph, only including the relevant substrate nodes has only a limited impact in this configuration, since we are considering substrate topologies with 512 nodes and 5 affinity restrictions per SFC.

To evaluate the impact of the number of affinity constraints on the semantic validation time, substrate topologies with 512 nodes were created and SFC requests with 10 VNFs instances were used. The number of randomly generated affinity constraints was varied from 5 to 80. With an increasing number of constraints, the probability of inconsistencies in the SFC requests increases. Therefore, the average validation times for both groups of inconsistent and consistent SFC requests are shown separately. Figure 7 shows that the average validation times increase up to a certain point, after which they drop again. This behavior can be accounted to the fact that the probability of easy-to-detect conflicts (e.g., two contradictory constraints) increases when the number of constraints increases, quickly terminating the reasoning process. When the number of constraints is lower, the variety of constraint types and subjects on which they are posing these constraints is larger, leading to more complex conflicts that only appear when more information is inferred from the ontology, causing the validation to take more time. None of the SFCs with over 40 constraints is consistent, leading to an execution time of $0s$ for this subgroup.

C. Impact of Semantic Validation on Mapping Time

In this set of experiments we evaluate the impact on the mapping time when performing semantic validation on SFC sets prior to running the embedding algorithm. Furthermore, the share of the semantic validation step on the total execution time is evaluated. To this end, SFC request sets are generated where a certain percentage of SFCs has conflicting affinity constraints. The substrate topologies under consideration contain 512 nodes, while the request sets contain 20 SFCs, of which each SFC contains 10 VNFs and 5 affinity restrictions. During this set of evaluations, the bandwidth minimization objective defined in Equation (31) is used.

Figure 8 shows the positive impact on the total mapping time per SFC request when semantically filtering the inconsistent SFC requests out of the request set prior to performing the mapping step. The SFC requests are mapped as a set and contain 20 SFC requests per set. By validating the SFC request set prior to mapping it, the total execution time can be reduced by 59% on average. Even when only 10% of the SFC requests are inconsistent, the execution times can be reduced with more than 25%. The relative execution time reduction approximately shows a logarithmic trend between 0% and 100% with an increasing number of inconsistent requests. These results show that it is beneficial to perform the semantic matching although it yields a small additional execution overhead when no inconsistent SFCs are present.

Figure 9 shows the same graph, but for the individual mapping of 100 SFC requests. On average, semantically filtering the request set reduces the mapping time with 50%. When no inconsistent SFC requests are present, there is an overhead incurred by the semantic matching step of 0.4%, which can be considered negligible compared to the total mapping time. For a 10% fraction of inconsistent SFCs per request set, the performance gain is about 11%.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2017.2681025, IEEE Transactions on Network and Service Management

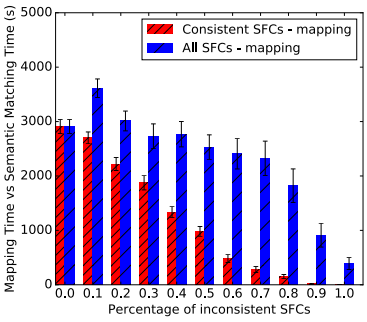

Fig. 8: Share of semantic matching and mapping on total execution time for SFC set mapping.

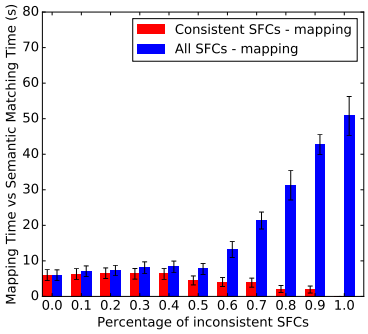

Fig. 9: Share of semantic matching and mapping on total execution time for individual SFC mapping.

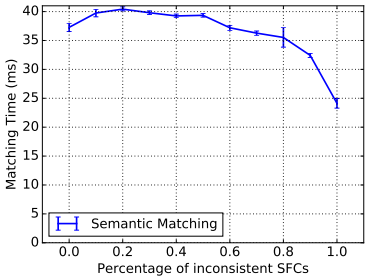

Fig. 10: Semantic matching execution time.

The previous graphs shows both the mapping and matching times. Figure 10 shows the matching time per SFC separately, note that the y-axis of this graph shows the time in ms instead of s. As can be seen from the graph, the semantic filtering takes about $40ms$ but yields a benefit which is 28000 times greater in the case of set mapping and 355 times greater for individual request mapping.

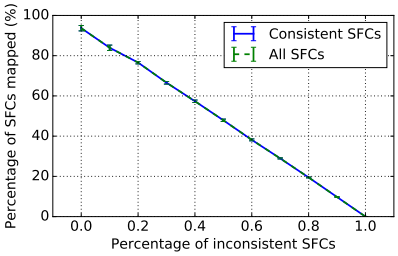

Fig. 11: Percentage of SFC requests that are mapped.

To assess the validity of the proposed semantic SFC validation framework, also the number of mapped SFC requests was tracked when filtering out inconsistent SFCs, as well as when the complete SFC request set is considered for mapping. Figure 11 shows that every inconsistent SFC that was filtered out during the semantic matching process, was indeed impossible to map during the embedding phase due to inconsistencies in the request.

D. Performance of Heuristic Approach

Mapping the SFC request sets in a single step yields an increase in total execution time of the mapping process due to an increasing number of decision variables and constraints in the mathematical model. To alleviate these problems, a heuristic mapping procedure is proposed in which the different SFCs in the set are first ordered using a certain criterion, after which they are individually mapped onto the infrastructure. Figure 12 shows the impact of performing the heuristic procedure on the total execution time. During the experiments, both approaches are using the bandwidth minimization objective defined in Equation (31), the heuristic approach orders the SFCs according to the requested bandwidth criterion defined in Equation (34). The total execution time can be reduced with a factor 3378 on average when performing the heuristic approach. The evaluations were performed using SFC request sets containing 20 SFCs and each SFC containing on average 5 VNFs and 3 affinity constraints. Mapping the SFC requests individually takes on average $0.85s$, including the semantic matching.

The heuristic approach comes at a cost in optimality, since each request is considered individually, the global optimum is not achieved. Figure 13 shows the allocated bandwidth as a percentage of the total bandwidth available in the infrastructure when minimizing the bandwidth usage. On average, the heuristic approach allocates 2.6% more bandwidth than the optimal solution. Also in terms of number of mapped SFC requests, the heuristic approach is outperformed by the optimal set mapping approach. Figure 14 shows that in some cases only 97.5% of the feasible SFC requests are mapped when applying the heuristic approach. This also implies that the resource consumption for the heuristic approach would be higher if the same amount of requests could be mapped.

E. Impact of Optimization Objective

Multiple optimization objectives were proposed in Section V. We compare the bandwidth minimization objective (MBW) proposed in Equation (31) with the load balancing objective (LBBW) of Equation (32). To compare the objectives, the maximum requested bandwidth for the virtual edges is varied. A set of 40 consistent SFC requests is generated and mapped onto the infrastructure resources. Figure 15 shows the impact of the objectives on the total resource usage. It is obvious that the LBBW optimization is outperformed in terms of total resource usage by MBW. Looking at the difference between the maximum and minimum load, shown in Figure 16, it is clear that the proposed balancing objective allows a better spread of the load compared to pure bandwidth optimization. Figure 17 shows the percentage of nodes that is used for the mapping, showing a higher node usage for LBBW. This graph also confirms that the load is spread across the infrastructure. The decrease in node usage with increasing bandwidth can be attributed to a reduced number of SFCs that can be mapped on the infrastructure due to capacity constraints.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2017.2681025, IEEE Transactions on Network and Service Management

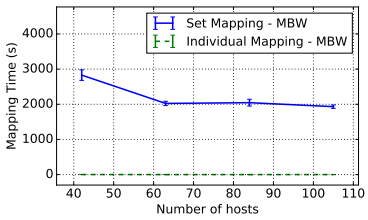

Fig. 12: Total mapping time of Set mapping compared to Individual mapping for increasing infrastructure size.

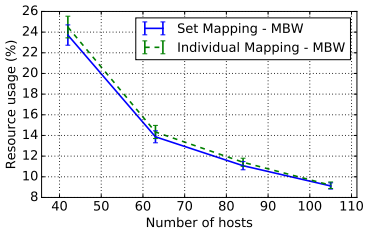

Fig. 13: Objective of Set mapping compared to Individual mapping for increasing infrastructure sizes.

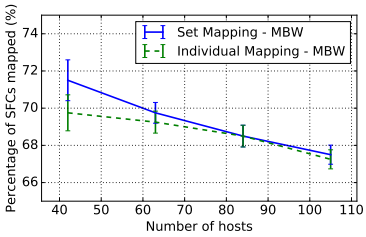

Fig. 14: Percentage of mapped SFCs when applying Set mapping compared to Individual mapping for increasing infrastructure sizes.

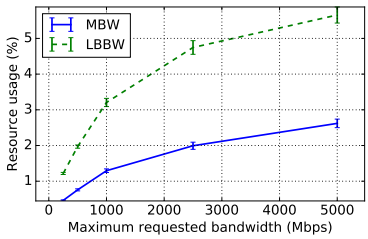

Fig. 15: Total link bandwidth usage for various optimization objectives.

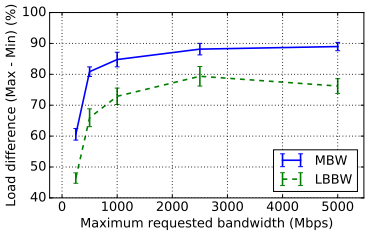

Fig. 16: Difference between the maximum and minimum link usage for various optimization objectives.

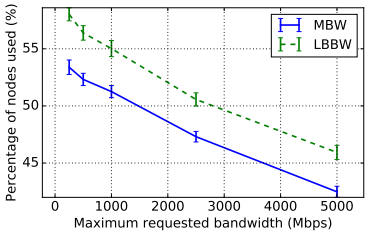

Fig. 17: Percentage of used nodes for various optimization objectives.

F. Impact of Ordering Criterion

To assess the impact of the ordering criteria proposed in Section VI, the experiments of the previous section were repeated for different criteria. The objective used is the bandwidth minimization objective. Each of the SFC requests has the same amount of affinity constraints attached to them. Figure 18 shows the difference in acceptance rate when ordering based on the requested bandwidth, compared to ordering on the number of constraints. In this experiment, the number of constraints is constant, so ordering based on number of constraints falls back to an online processing behavior, where demands are taken one after another. Mapping the SFCs with the lowest bandwidth requirements first, increases the acceptance rate significantly, since the load on the infrastructure is minimized.

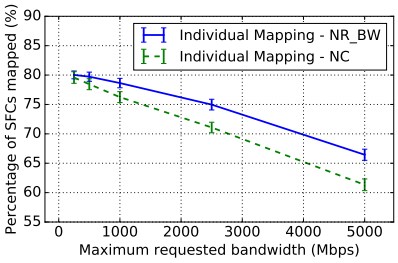

Fig. 18: Impact of SFC ordering on acceptance rate.

VIII. Conclusion and Future Work

This paper proposes a way for Service Providers (SPs) to attach location and colocation constraints to the mapping of Service Function Chains (SFCs) onto the substrate. These affinity constraints can be used to increase efficiency and resilience, to adhere to legislative and privacy restrictions or for economic reasons. First, the different sets of affinity and anti-affinity constraints are formalized. Second, a semantic validation framework is proposed, which allows the Virtual Network Function Infrastructure Provider (VNFInP) to check the consistency of the constraints posed by the SFC requests. To this end, the substrate and the SFC request are modeled using an ontology of which the consistency is checked using a semantic reasoner. Finally, the SFC embedding problem subject to affinity constraints is formalized and an Integer Linear Programming (ILP) formulation for both set-based SFC and individual SFC mapping is proposed. The semantic validation and different mapping algorithms were evaluated thoroughly. By only loading the parts of the infrastructure that are relevant for the SFC request into the ontology, the number of individuals and thus the semantic reasoning time can be significantly reduced. Furthermore, by filtering out inconsistent SFC requests before mapping, the total execution time of the embedding algorithm can be reduced with more than 50% for the considered scenarios. Next to the set embedding formulation, also a heuristic approach is proposed in which the SFCs requests are embedded individually. To this end, the SFC requests are ordered based on certain criteria, after which the embedding is performed by solving the ILP.

In future work, alternative techniques for modeling the SFC mapping will be studied. For example, modeling it as a combined capacitated facility location-flow routing problem allows greatly reducing the complexity compared to the modeling approach taken in this paper. By combining the proposed semantic filtering approach with more efficient mapping algorithms could potentially allow VNFInPs to find

optimal SFC mappings in reasonable time.

ACKNOWLEDGMENT

Niels Bouten is funded by a Ph.D. grant of the Agency for Innovation by Science and Technology in Flandres (IWT). This work was partially funded by the FWO project 11385 "Service-centric management of a virtualized future Internet". This work was partially funded by Flamingo, a Network of Excellence project (318488) supported by the European Commission under its Seventh Framework Programme. The computational resources (Stevin Supercomputer Infrastructure) and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by Ghent University, the Hercules Foundation and the Flemish Government - department EWI.

REFERENCES

[1] ETSI, "Network functions virtualization: An introduction, benefits, enablers, challenges and call for action," *Cloud Computing, IEEE Transactions on*, October 2012. [Online]. Available: http://portal.etsi.org/NFV/NFV_White_Paper.pdf

[2] ——, "Network functions virtualization: Network operator perspectives on industry progress," *Cloud Computing, IEEE Transactions on*, October 2013.

[3] S. Latre, J. Famaey, F. De Turck, and P. Demeester, "The fluid internet: service-centric management of a virtualized future internet," *Communications Magazine, IEEE*, vol. 52, no. 1, pp. 140–148, January 2014.

[4] R. Mijumbi, "On the energy efficiency prospects of network function virtualization," *arXiv preprint arXiv:1512.00215*, 2015.

[5] S. Boucadair, D. Lopez, I. Telefonica, D. Guichard, and C. Pignataro, "Service function chaining: Framework & architecture draft-boucadair-sfc-framework-00," 2014.

[6] A. Fischer, J. Botero, M. Till Beck, H. de Meer, and X. Hesselbach, "Virtual network embedding: A survey," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 4, pp. 1888–1906, Fourth 2013.

[7] M. Chiosi, D. Clarke, P. Willis, A. Reid, J. Feger, M. Bugenhagen, W. Khan, M. Fargano, C. Cui, H. Denf *et al.*, "Network functions virtualisation: An introduction, benefits, enablers, challenges and call for action," in *SDN and OpenFlow World Congress*, 2012, pp. 22–24.

[8] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *Communications Magazine, IEEE*, vol. 53, no. 2, pp. 90–97, 2015.

[9] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *Communications Surveys Tutorials, IEEE*, vol. PP, no. 99, pp. 1–1, 2015.

[10] S. Benkner and C. GEMSS, "Report on cots security technologies and authorisation services," Project Report, February 2004. [Online]. Available: http://eprints.cs.univie.ac.at/3311/

[11] P. Massonet, S. Naqvi, C. Ponsard, J. Latanicki, B. Rochwerger, and M. Villari, "A monitoring and audit logging architecture for data location compliance in federated cloud infrastructures," in *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on*, May 2011, pp. 1510–1517.

[12] K. Konstanteli, T. Cucinotta, K. Psychas, and T. Varvarigou, "Admission control for elastic cloud services," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, June 2012, pp. 41–48.

[13] L. Larsson, D. Henriksson, and E. Elmroth, "Scheduling and monitoring of internally structured services in cloud federations," in *Computers and Communications (ISCC), 2011 IEEE Symposium on*, June 2011, pp. 173–178.

[14] D. Espling, L. Larsson, W. Li, J. Tordsson, and E. Elmroth, "Modeling and placement of cloud services with internal structure," *Cloud Computing, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2014.

[15] H. Basilier, M. Darula, and J. Wilke, "Virtualizing network services–the telecom cloud," *Ericsson Review*, 2014. [Online]. Available: http://www.ericsson.com/res/thecompany/docs/publications/ericsson_review/2014/er-telecom-cloud.pdf

[16] H. Technologies, "White paper - huawei observation to nfv," 2014. [Online]. Available: www.huawei.com/ilink/en/download/HW_399662

[17] E. I. S. G. I. NFV, "Network functions virtualisation (nfv); service quality metrics," 2014. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-INF/001_099/010/01.01.01_60/gs_nfv-inf010v010101p.pdf

[18] M. Bari, R. Boutaba, R. Esteves, L. Granville, M. Podlesny, M. Rabbani, Q. Zhang, and M. Zhani, "Data center network virtualization: A survey," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 2, pp. 909–928, Second 2013.

[19] E. Correa, L. Fletscher, and J. Botero, "Virtual data center embedding: A survey," *Latin America Transactions, IEEE (Revista IEEE America Latina)*, vol. 13, no. 5, pp. 1661–1670, May 2015.

[20] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," *CoRR*, vol. abs/1406.1058, 2014.

[21] M. T. Beck and J. F. Botero, "Coordinated allocation of service function chains," in *2015 IEEE Global Communications Conference (GLOBECOM)*, Dec 2015, pp. 1–6.

[22] S. Mehraghdam and H. Karl, "Placement of services with flexible structures specified by a yang data model," in *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, June 2016, pp. 184–192.

[23] H. Moens and F. De Turck, "Vnf-p: A model for efficient placement of virtualized network functions," in *Network and Service Management (CNSM), 2014 10th International Conference on*, Nov 2014, pp. 418–423.

[24] M. Luizelli, L. Bays, L. Buriol, M. Barcellos, and L. Gaspary, "Piecing together the nfv provisioning puzzle: Efficient placement and chaining of virtual network functions," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, May 2015, pp. 98–106.

[25] M. Xia, M. Shirazipour, Y. Zhang, H. Green, and A. Takacs, "Network function placement for nfv chaining in packet/optical datacenters," *J. Lightwave Technol.*, vol. 33, no. 8, pp. 1565–1570, Apr 2015.

[26] S. Sahhaf, W. Tavernier, D. Colle, and M. Pickavet, "Network service chaining with efficient network function mapping based on service decompositions," in *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, April 2015, pp. 1–5.

[27] 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*, Sept 2014, pp. 1–6.

[28] I. Vaishnavi, R. Guerzoni, and R. Trivisonno, "Recursive, hierarchical embedding of virtual infrastructure in multi-domain substrates," in *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, April 2015, pp. 1–9.

[29] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. D. Turck, and S. Davy, "Design and evaluation of algorithms for mapping and scheduling of virtual network functions," in *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, April 2015, pp. 1–9.

[30] A. Basta, W. Kellerer, M. Hoffmann, H. J. Morper, and K. Hoffmann, "Applying nfv and sdn to lte mobile core gateways, the functions placement problem," in *Proceedings of the 4th Workshop on All Things Cellular: Operations, Applications, & Challenges*, ser. AllThings-Cellular '14. ACM, 2014, pp. 33–38.

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

[32] A. Baumgartner, V. Reddy, and T. Bauschert, "Mobile core network virtualization: A model for combined virtual core network function placement and topology optimization," in *Network Softwarization (NetSoft), 2015 1st IEEE Conference on*, April 2015, pp. 1–9.

[33] M. Bouet, J. Leguay, T. Combe, and V. Conan, "Cost-based placement of vdpi functions in nfv infrastructures," *International Journal of Network Management*, vol. 25, no. 6, pp. 490–506, 2015.

[34] J. G. Herrera and J. F. Botero, "Resource allocation in nfv: A comprehensive survey," *IEEE Transactions on Network and Service Management*, vol. PP, no. 99, pp. 1–1, 2016.

[35] N. Bouten, M. Claeys, R. Mijumbi, J. Serrat, J. Famaey, S. Latré, and F. De Turck, "Semantic validation of affinity constrained service function chain requests," in *Network Softwarization (NetSoft), 2016 2nd IEEE Conference on*, June 2016, pp. 1–9.

[36] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 63–74, 2008.