Semantic Validation of Affinity Constrained Service Function Chain Requests

Niels Bouten*, Maxim Claeys*, Rashid Mijumbi†, Joan Serrat‡, Jeroen Famaey§, Steven Latré§, Filip De Turck*
*Department of Information Technology, Ghent University - iMinds, Gaston Crommenlaan 8/201, B-9050 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) has been proposed as a paradigm to increase the cost-efficiency, flexibility and innovation in network service provisioning. By leveraging IT virtualization techniques in combination with programmable networks, NFV is able to decouple network functionality from the physical devices on which they are deployed. This opens up new business opportunities for both Infrastructure Providers (InPs) as well as Service Providers (SPs), where the SP can request to deploy a chain of Virtual Network Functions (VNFs) on top of which its service can run. However, current NFV approaches lack the possibility for SPs to define location requirements and constraints on the mapping of virtual functions and paths onto physical hosts and links. Nevertheless, many scenarios can be envisioned in which the 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 framework is proposed that allows the InP to check the validity of a set of constraints and provide feedback to the SP. To achieve this, the Service Function Chain (SFC) request and relevant information on the physical topology are modeled as an ontology of which the consistency can be checked using a semantic reasoner.**

I. Introduction

In the traditional telecommunications networking approach, functionality of a network node is strongly tied with the physical network device it runs on. Typically, the network operator needs to deploy a dedicated network appliance for each Network Function (NF) (e.g., Deep Packet Inspection (DPI), Firewall). In addition, NFs have a strict chaining that must be adhered to when deploying a specific service. Thus, service deployments are tightly coupled to the underlaying network topology. These reasons, together with the ever increasing requirements for high quality and stability have led to long product cycles, limited service agility and considerable dependence on specialized hardware. To be able to compete with Over-The-Top (OTT) service providers, which typically have much shorter product development cycles, and to limit the Capital Expenditures (CAPEX) and Operational Expenditures (OPEX) involved with physical network expansions, the network operators need to devise novel and less expensive ways to meet the increased capacity requirements and at the same time reduce the time to market of new services.

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. This allows NFs to run on standard high volume servers, storage devices and switches. The advantages are manyfold. First there potentially is a significant reduction in total costs through more efficient maintenance which can be performed remotely. In addition, thanks to the increased flexibility offered by virtualization, resources can be shared and used more efficiently. Finally, NFV has the potential to allow network operators to deploy novel services cheaper and faster with higher service agility.

The concepts of NFV open up new business opportunities in the form of Virtual Network Function Infrastructure Providers (VNFInPs), acting as brokers between Infrastructure Providers (InPs) and Service Providers (SPs). These VNFInPs lease the infrastructure offered by different InPs and deploy, orchestrate and interconnect Virtual Network Functions (VNFs) to create Service Function Chains (SFCs) [3], that are run by SPs to offer value-added services to their customers. InPs can profit by maximizing resource utilization and optimizing energy usage by offering their virtualized infrastructure to remote parties. SPs benefit from the proposed model since it allows rapid deployment and testing in a real network environment, thus leveraging faster time to market of new services. The offered services benefit from the dynamic nature of the network, computing and storage resources offered by the Virtual Network (VN), which allows them to scale dynamically based on service requirements and user mobility.

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, 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 NFs as well as constraints on their affinity. For example, to increase efficiency, the SP might want to require the embedding of VNFs 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 SFC requests. Therefore, the VNFInP needs to be provided with a means to check the validity of SFC requests and inform the SP on potential conflicts. 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 paper 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 of this entire ontology can be determined and subsequently the validity of the SFC request can be assessed.

The contributions of this paper are threefold. First, the sets set of affinity and anti-affinity constraints are defined that can be attached to a SFC request by the SP. Second, we extend an existing virtualization description language to support these constraints. Finally, we propose and evaluate a semantic conflict detection mechanism that can be employed by the VNFInP to check the validity of SFC requests.

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 [4], [5]. 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 [6] and virtual datacenter embedding [7], as well as for NFV [8]. A placement algorithm can be formulated as an optimization problem with a particular objective such as load balancing, resource utilization, acceptance ratio, etc. *Basta et al.* propose a model for placing virtualized Evolved Packet Core (EPC) functions in a way that minimizes the network overhead introduced by Software Defined Networking (SDN) control plane interactions [9]. *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 placment of functions should be different according to the desired objective [10]. *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 [11]. Others propose a heuristic approach to deal with the intractability of the aforementioned optimization approaches. *Xia et al.* propose a greedy heuristic which sorts VNFs according to the resource demands and embeds the resource-demanding VNFs with highest priority [12]. *Yoshida et al.* propose a multi-objective resource scheduling algorithm which optimizes simultaneously possibly conflicting objectives with multifaceted costraints [13]. 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.

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 [14], [15]. Even though performance and economical benefits of cloud computing are clear, potential users hesitate to use 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 about their data privacy, raising the need for SPs to take into account privacy and legal issues when offering their services. 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 affinity and anti-affinity context in 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) [16]. The authors define constraints that require allocating components/services in the same subnet or physical node or prevent services to be federated. *Espling et al.* propose a model for defining Virtual Machine (VM) placement in cloud computing supporting a set of affinity and anti-affinity constraints [17], [18]. 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. Furthermore, 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 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. Also Service Level Agreement (SLA)-related parameters such as affinity and anti-affinity rules should be transformed into machine-readable description formats [19]. Huawei mentions the generation of affinity and anti-affinity policies as a mechanism for fault prevention [20]. In the definition of *Service Quality Metrics* by ETSI, special attention is brought to the enforcement of NFV customer anti-affinity rules which can improve the availability mechanisms [21]. 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 for both user-generated as well as automatically generated affinity constraint sets.

III. Affinity and Anti-Affinity Constraint Model

In an NFV context, SPs have no 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.
- **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, a set of affinity and anti-affinity constraints for VNFs and their interconnecting paths are proposed. The affinity constraints apply to a set of physical locations P, a set of VNF instances V and a set of edges interconnecting them E. There are different location granularities $g \in G$ that can be considered (e.g., network domains, datacenters, hosts), 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 datacenter level. $P^g \subset P$ is the set of locations at a certain granularity g. Furthermore, each VNF instance has an associated VNF type $t \in T$ (e.g., firewall, DPI), forming subsets $V^t \subseteq V$ of VNFs with type 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 the following 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 $t \in T$ may not be located at a specific location p with granularity g.
- 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 $v \in V^s$ must be placed together with a specific instance w or all instances $w \in V^t$ 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 all instances $v \in V^s$ may not be placed together with a specific instance w or all instances $w \in V^t$ at a specific location $p \in P^g$ or at the same location at a specific granularity $g \in G$.
- Affinity($p \in P^g, e \in E[, c \in (0, 1]]$): A virtual edge $e \in E$ must pass through a specific location $p \in P^g$ with a granularity $g \in G$. The parameter c is an optional percentage, defining the portion of links comprising the physical path associated with the virtual edge that must at least be part of $p \in P^g$. If $c = 1$, all links must belong to $p \in P^g$. If c is omitted, at least one link must be part of $p \in P^g$.
- Anti-Affinity($p \in P^g, e \in E$): The physical links comprising the virtual edge $e \in E$ may not pass through a specific location $p \in P^g$ with a granularity $g \in G$.
- Affinity($e \in E, f \in E$): Two virtual edges $e \in E$ and $f \in E$ must overlap (i.e. one or more physical links comprising the virtual edges must be part of both e and f).
- Anti-Affinity($e \in E, f \in E$): Two virtual edges $e \in E$ and $f \in E$ may not overlap (i.e. none of the physical links comprising the virtual edges may be part of both e and f).

A wide range of languages could be used to define the constraints outlined above, depending on the language used to define the SFC template. As an SFC is generally a directed acyclic graph structure, the specification language should be capable of modelling this. To model the constraints, we use an extension of the DTMF Open Virtualization Format (OVF) version 2.1.1[1]. The OVF descriptor is an XML-based language for annotating software to be run in virtual machines, such as product details, virtual hardware requirements and licensing. *Espling et al.* [18] expanded the OVF descriptor with additional constructs for defining constraints in structured cloud services. We expanded upon this work to additionally model the affinity and anti-affinity constraints for both node and edge mapping for SFCs. Figure 1 shows how the affinity constraint subtypes could be defined.

To further clarify the presented constraint formulations and syntax, an example of an SFC request with both affinity and anti-affinity constraints will be presented next. Given a set of location types {*Autonomous System (AS), Datacenter (DC), Host*} and a set of network function types {*Firewall, DPI, Cache, StreamingServer*}. An example SFC is depicted in Figure 2, where a streaming server is connected to two DPI functions (for tagging data packets), which in turn are connected to a firewall (for filtering) and a content cache. The DPI functions may either directly forward content to the cache or may send it to the firewall for filtering. Suppose a SP wants to offer a Video on Demand (VoD) service in Belgium where two major telecom providers are active: Telenet (*AS6848*) and Proximus (*AS6774*). Let us consider the following set of affinity and anti-affinity constraints:

- *Affinity(AS6848, c1)*
- *Affinity(AS6774, c2)*

[1]DMTF - OVF Specification - https://www.dmtf.org/sites/default/files/standards/documents/DSP0243_2.1.1.pdf

```
<xs:element name="Affinity" type="Constraint">
   <xs:complexType>
      <xs:choice>
         <xs:sequence>
            <xs:choice>
               <xs:element name="locType" type="LocationType" />
               <xs:element name="loc" type="Location" />
            </xs:choice>
            <xs:choice>
               <xs:element name="funcTypeA" type="FunctionType" />
               <xs:element name="funcA" type="NetworkFunction" />
            </xs:choice>
            <xs:choice minOccurs="0">
               <xs:element name="funcTypeB" type="FunctionType" />
               <xs:element name="funcB" type="NetworkFunction" />
            </xs:choice>
         </xs:sequence>
         <xs:sequence>
            <xs:element name="connC" type="Connection" />
            <xs:choice>
               <xs:element name="connD" type="Connection" />
               <xs:choice>
                  <xs:element name="locType" type="LocationType" />
                  <xs:element name="loc" type="Location" />
               </xs:choice>
            </xs:choice>
            <xs:element name="perc" type="Percentage" minOccurs="0" />
         </xs:sequence>
      </xs:choice>
   </xs:complexType>
</xs:element>
```

Fig. 1: OVF specification extension for modelling Affinity node and link constraints.

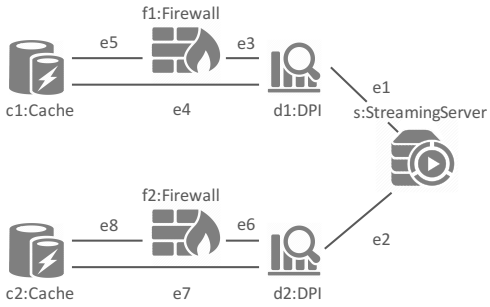

Fig. 2: An example SFC.

- *Affinity(DC, e3, 1)*
- *Affinity(DC, e6, 1)*
- *AntiAffinity(e1,e2)*
- *AntiAffinity(DC, DPI, DPI)*

Specifically, the first two constraints state that the caches 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 and fourth constraint define that the edges *e3* and *e6* between firewall and DPI functions should be completely embedded within a single DC, automatically forcing the DPI and connected firewall to be deployed within that DC. Finally, to improve fault tolerance, it is stated that edges *e1* and *e2* can not have any links in common and that functions with type DPI should not be deployed within the same DC. Using the XSD schema defined above, the constraints can be represented as shown in Figure 3.

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. For example, extending the previous example and adding the constraints *Affinity(DC, c1, f1)* (i.e. specifying that *c1* and *f1* should be colocated in the same

```
<Affinity>
   <location>AS6848</location>
   <functionA>c1</functionA>
</Affinity>
<Affinity>
   <location>AS6774</location>
   <functionA>c2</functionA>
</Affinity>
<Affinity>
   <locationType>DataCenter</locationType>
   <connectionC>e3</connectionC>
   <percentage>1</percentage>
</Affinity>
<Affinity>
   <locationType>DataCenter</locationType>
   <connectionC>e6</connectionC>
   <percentage>1</percentage>
</Affinity>
<AntiAffinity>
   <connectionC>e1</connectionC>
   <connectionD>e2</connectionD>
</AntiAffinity>
<AntiAffinity>
   <locationType>DataCenter</locationType>
   <functionTypeA>DPI</functionTypeA>
   <functionTypeB>DPI</functionTypeB>
</AntiAffinity>
```

Fig. 3: A simplified constraint specification for the example SFC

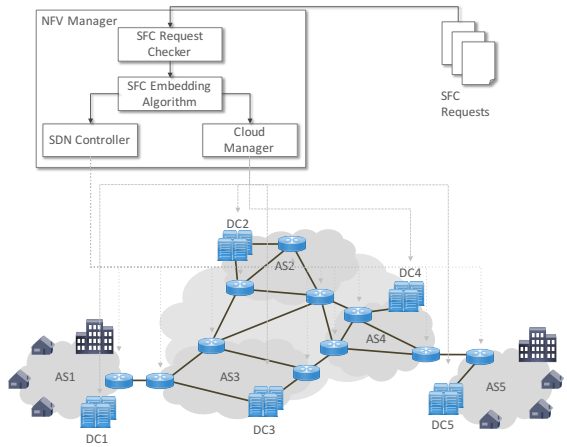

Fig. 4: An overview of the NFV architecture with support for semantic SFC request checking.

DC) and *AntiAffinity(DC, Cache, Firewall)* (i.e. specifying that a VNF of type Cache can not 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(AS, c1, f1)* and *Anti-Affinity(AS6848, d1)* would lead to a conflict set {*Affinity(DC, c1, f1), Anti-Affinity(AS6848, d1), Affinity(AS6848, c1), Affinity(DC, e3, 100)*}. Since *d1* and *f1* should be colocated in the same DC due to the link affinity constraint and *f1* and *c1* are colocated at the DC level, *d1* and *c1* should be colocated at the AS level as well. Furthermore, since *c1* should be located in *AS6848*, *d1* should be located in the same AS. However, this inferred constraint conflicts with the defined constraint *Anti-Affinity(AS6848, d1)*.

When the VNFInP tries to deploy the requested SFC, none of the resulting embedding configurations will lead to

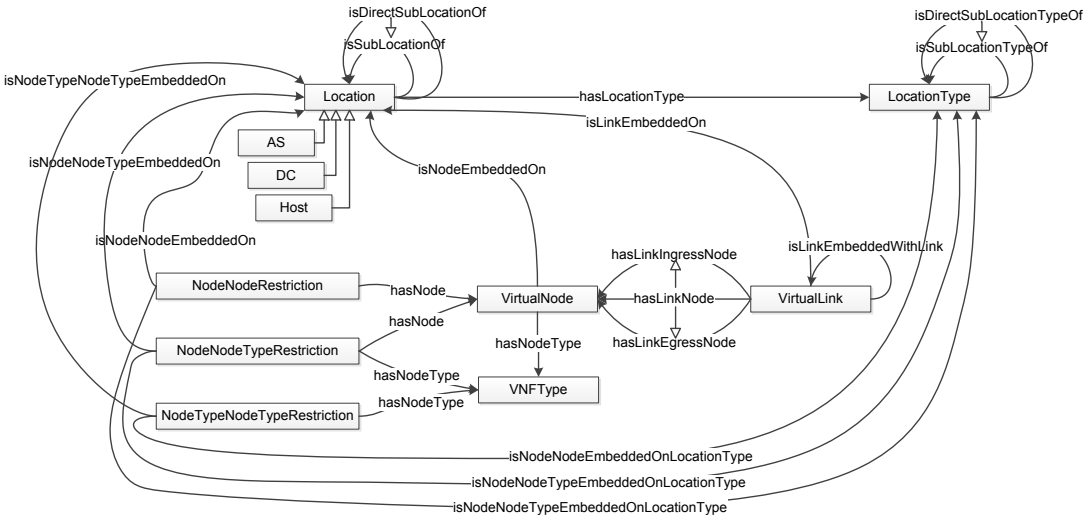

Fig. 5: Graphical representation of ontology.

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.

A. NFV Architecture for SFC Request Checking

Figure 4 depicts how the SFC request checking system could be integrated into the *NFV Manager*. In this architecture, the *SFC Embedding Algorithm* is responsible for assigning physical hardware and resources to the SFC requests. Concretely, it decides on which VNFs should be deployed on which physical 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.

B. Ontology for SFC Request Modelling

This paper proposes to exploit ontology representations for the purpose of modelling the physical substrate, the SFC request and defining a set of rules that can be used to infer additional information. Figure 5 represents the proposed semantic model. The SFC request is modelled as a set of *VirtualNodes* with a certain *VNFType* and *VirtualLinks* containing an ingress and egress *VirtualNode*. 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 virtual nodes and edges, positive (respectively negative) object property assertions of the type *isNodeEmbeddedOn* and *isLinkEmbeddedOn* are attached to *VirtualNodes* and *VirtualLinks* respectively.

To be able to model more complex affinity and anti-affinity relationships between two *VirtualNodes*, two *VNFTypes* or between a *VirtualNode* and *VNFType*, the additional concepts *NodeNodeRestriction*, *NodeNodeTypeRestriction* and *NodeTypeNodeTypeRestriction* were added to the ontology. By adding the respective positive (respectively negative) property *isNodeNodeEmbeddedOn* or *isNodeNodeEmbeddedOnType*, one is able to model affinity (respectively anti-affinity) restrictions for more complex constraints on the *Location* or *LocationType*. By using the *isLinkEmbeddedWith* relationship, affinity and anti-affinity constraints between links can be modeled.

The Protégé editor[2] was used to develop the SFC request modelling ontology using the Web Ontology Language (OWL)[3].

C. Rules

To be able to infer new information out of existing knowledge, a set of rules is defined. For example, a new relationship is defined in Rule (1) and (2), which is called *isSubLocOrEqualOf* that checks if a certain *Location a* is either equivalent to *b* or a *isSubLocationOf b* is valid. This relationship will be used later on to infer knowledge on affinity and anti-affinity characteristics at a certain *LocationType*.

$$SameAs(a,b) \rightarrow SubLocOrEqualOf(a,b) \quad (1)$$

$$SubLocOf(a,b) \rightarrow SubLocOrEqualOf(a,b) \quad (2)$$

Rule (3) stipulates that if a certain *VirtualNode x* is embedded on a *Location y* and if *y* is a sublocation of *z*, this node is also embedded on *Location z*. When a *VirtualNode x* of a certain *VNFType y* is embedded on a *Location z*, the *VNFType y* is also embedded on that *Location z* (Rule (4)). Rule (5) determines that if two *VirtualNodes x* and *y* are both

[2]Protégé - http://protege.stanford.edu/

[3]OWL2 - http://www.w3.org/TR/owl-features/

embedded on a *Location a*, then the *NodeNodeRestriction z* containing *x* and *y* is embedded on the same *Location a*. Similar rules can be determined for *NodeNodeTypeRestriction* and *NodeTypeNodeTypeRestriction*, which are omitted due to space restrictions. Rule (6) stipulates the opposite: if a *NodeNodeRestriction z* containing *VirtualNodes x* and *y* is embedded on *Location a*, then both *VirtualNode x* and *y* are embedded on *Location a*. Similar rules can be determined for *NodeNodeTypeRestriction* and *NodeTypeNodeTypeRestriction*. Rule (7) determines that if *VirtualNodes x* and *y* are embedded on *Location a* and *Location b* respectively and if both *a* and *b* are either sublocations of or equal to *Location c* with *LocationType l* and *x* is not equal to *y*, then the *NodeNodeRestriction z* containing both *x* and *y* is embedded on *LocationType l*. Similar rules can be determined for *NodeNodeTypeRestriction* and *NodeTypeNodeTypeRestriction*. If a *VirtualLink z* contains a *VirtualNode x* embedded at *Location a*, this *VirtualLink z* is embedded at *Location a* as well (Rule (8)). Rule (9) states that if two *VirtualLink x* and *y* have a *VirtualNode v* in common, they are overlapping.

$$isNodeEmbeddedOn(x,y) \wedge isSubLocOf(y,z) \rightarrow isNodeEmbeddedOn(x,z) \quad (3)$$

$$hasNodeType(x,y) \wedge isNodeEmbeddedOn(x,z) \rightarrow isNodeTypeEmbeddedOn(y,z) \quad (4)$$

$$hasNode(z,x) \wedge isNodeEmbeddedOn(x,a) \wedge hasNode(z,y) \wedge isNodeEmbeddedOn(y,a) \wedge Different(x,y) \rightarrow isNodeNodeEmbeddedOn(z,a) \quad (5)$$

$$isNodeNodeEmbeddedOn(z,a) \wedge hasNode(z,x) \wedge hasNode(z,y) \rightarrow isNodeEmbeddedOn(x,a) \wedge isNodeEmbeddedOn(y,a) \quad (6)$$

$$hasNode(z,x) \wedge isNodeEmbeddedOn(x,a) \wedge hasNode(z,y) \wedge isNodeEmbeddedOn(y,b) \wedge isSubLocOrEqualOf(a,c) \wedge isSubLocOrEqualOf(b,c) \wedge hasLevel(c,l) \wedge Different(x,y) \rightarrow isNodeNodeEmbeddedOnType(z,l) \quad (7)$$

$$hasLinkNode(z,x) \wedge isNodeEmbeddedOn(x,a) \rightarrow isLinkEmbeddedOn(z,a) \quad (8)$$

$$hasLinkIngressNode(x,v) \wedge hasLinkEgressNode(y,v) \rightarrow isLinkEmbeddedWithLink(x,y) \quad (9)$$

Semantic Web Rule Language (SWRL)[4] was used to express the aforementioned rules using concepts from the ontology defined in Section IV-B. The Protégé editor was also used to define the rules using the Manchester syntax[5].

D. Conflict Detection

When a new SFC request arrives at the VNFInP, this request is parsed and the set of virtual nodes and links are added as individuals to the OWL ontology using the OWL API[6]. Next, the set of affinity and anti-affinity constraints are also added by either creating new individuals (i.e. *NodeNodeRestriction*), adding property assertions (i.e. *isNodeEmbeddedOn*) or both.

[4]SWRL - http://www.w3.org/Submission/SWRL/

[5]Manchester Syntax - http://www.w3.org/2007/OWL/wiki/ManchesterSyntax

[6]OWL API - http://owlapi.sourceforge.net

TABLE I: Overview of the evaluation parameters.

		Network Size			# Affinity and Anti-Affinity Constraints	# VNFs
		#AS	#DC	#Hosts		
All	Individuals	1	4	100	5	5
		2	8	200	10	10
		4	16	400	20	20
		8	32	800	40	40
		16	64	1600	80	80
		32	128	3200	160	160
Relevant	Individuals	5	50	25000	5	5
		10	100	50000	10	10
		20	200	100000	20	20
		40	400	200000	40	40
		80	800	400000	80	80
		160	1600	800000	160	160
		320	3200	1600000	320	320

The HermiT OWL Reasoner[7] 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.

V. Evaluation

We created a Java-based simulation framework in which we are able to model the physical substrate and the SFC requests. The components contained in the physical infrastructure are added as individuals to the ontology using the OWL API. Upon reception of an SFC request, the required individuals for the VNFs and their interconnecting links are added to a copy of this ontology. The affinity constraints defined in the SFC request are parsed and the necessary individuals and rules are instantiated. Using the HermiT OWL Reasoner, the consistency of the resulting ontology is checked. The execution time and resulting consistency are logged. We evaluate two implementations of the validation component. In the first version, all locations of the physical topology are instantiated in the ontology. In the second version, rather than including all physical locations into the ontology, only the physical locations that occur in the SFC request are added to the ontology as individuals, as well as the hierarchy of their parent locations. This is done so to reduce the size of the ontology as it is known that semantic approaches suffer from scalability issues when ontology sizes increase. The evaluations were performed using the Flemish Supercomputer Center (VSC) which contains nodes with 2x8-core Intel E5-2670 (Sandy Bridge @ 2.6 GHz) processors and 32 GB RAM.

To evaluate the performance of the proposed semantic validation framework, a number of random network topologies and SFC requests are generated. To generate the network topology, multiple interconnected hierarchical structures are generated where each host is part of a DC which is in turn

[7]HermiT OWL Reasoner - http://hermit-reasoner.com

part of an AS. Each of the hosts has a specific physical location that can be seen as a hierarchical combination of AS, DC and host identification. The total number of hosts, DCs and ASs are listed in Table I for both the case where we include all individuals and where only relevant individuals are considered. The number of interconnections for each node are uniformly distributed between 2 and 10. The SFC requests are randomly generated as well where the number of VNFs and the number of Affinity and Anti-Affinity constraints is varied. The type of the constraints is uniformly distributed among the constraints defined in Section III. The parameter values for both the number of VNFs and the number of constraints per SFC request are shown in Table I. For each parameter configuration, 1000 random SFC requests are generated and fed to the semantic request checker. The execution times are averaged and the 95% confidence intervals are shown in the graphs.

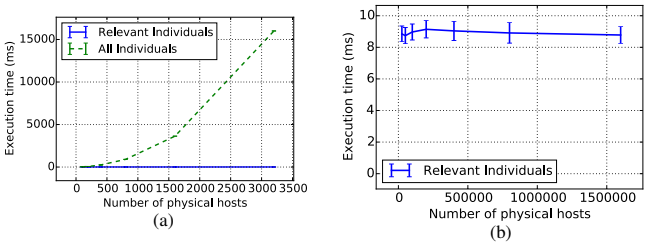

Fig. 6: Impact of physical network size

A. Impact of physical network size

To evaluate the impact of the physical network size, we use a fixed number of 5 virtual nodes and 5 (affinity or anti-affinity) constraints in the requested SFC, and vary the number of physical hosts as shown in Table I. For the case where we include an individual in the ontology for each physical location in the topology, it can be seen from Figure 6(a) that the execution time grows exponentially with increasing physical network size as was expected. Even for a network of only 3200 physical hosts, the semantic request validation already takes about $15s$. When we include only an individual for each physical location that is used in a constraint, there is no significant impact on the evaluation time, since the number of constraints is fixed in this case as demonstrated in Figure 6(b). These results demonstrate that it is indeed beneficial to only consider the relevant physical locations when semantically validating an SFC request.

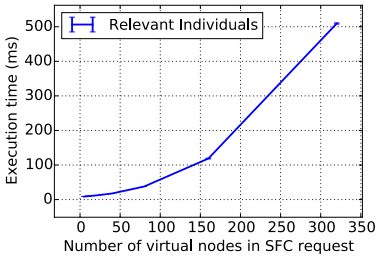

Fig. 7: Impact of virtual network size

B. Impact of virtual network size

To evaluate the impact of the size of the requested virtual topology in the SFC, a fixed number of 5 constraints and a network size of 25000 hosts is used, while the number of virtual nodes in the SFC request is varied as shown in Table I. Figure V-A shows an exponential increase of the execution time when the number of requested VNFs in the SFC request increases. For 320 VNFs in a single request, which is quite high, the execution time is $0.5s$, which could be considered as an acceptable calculation delay for SFC requests of that size.

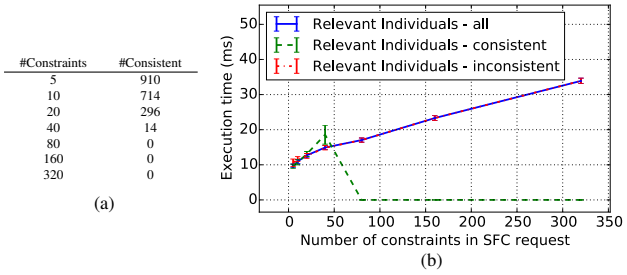

Fig. 8: Impact of number of constraints on number of consistent SFC requests (a) and execution time (b)

C. Impact of number of constraints

To evaluate the impact of the number of constraints in the SFC, a fixed virtual network size of 5 VNFs and a physical network size of 25000 hosts is used, and the number of constraints in the SFC request is varied as shown in Table I. Figure 8(b) shows how the total execution time is affected by increasing the number of constraints in the request. Two additional lines are plotted differentiating between the execution times for consistent and inconsistent SFCs. As can be seen from Table 8(a), when the number of constraints increases, the number of consistent SFCs is reduced significantly since the probability of conflicting constraints is increased. When the number of constraints exceeds 40, all generated SFC requests are inconsistent. Figure 8(b) shows a linear increase of the execution time when the number of constraints increases.

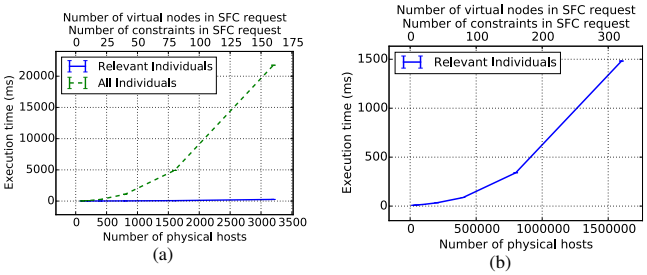

Fig. 9: Combined impact of increasing physical network size, requested virtual network size and number of constraints

D. Combined impact of relevant parameters

Finally we evaluated the combined impact of all parameters as stated in Table I, both for the case when all physical locations are considered and for the case when only relevant physical locations are taken into account. For both approaches, an exponential increase can be observed in Figure 9 when

the physical network size, requested virtual network size and the number of constraints increase. When considering all physical locations as individuals, Figure 9(a) shows that for a physical network of 3200 hosts, a virtual network of 160 VNFs and 160 constraints, the execution time is as high as $21.76s$. When reducing the ontology size by only including relevant locations, the execution time can be reduced to $0.34s$. Figure 9(b) shows the results for larger network topologies and when only relevant locations are added to the ontology. Also here, an exponential increase in execution times can be observed. However, when considering the size of the SFC request in both number of virtual nodes (320) and affinity constraints (320), a total execution time of $1.5s$ can be considered acceptable. Bearing in mind that increasing the size of the physical topology has a very limited impact on the total execution time since only the relevant individuals are added.

VI. Conclusion

In this paper we propose a means for Service Providers (SPs) to attach location constraints to the mapping of Service Function Chains (SFCs) onto the physical substrate. The SP's main interests to do this are for efficiency, resilience, legislative, privacy and economic reasons. We propose and define a set of affinity and anti-affinity constraints which allow the SP to define whether certain Virtual Network Functions (VNFs) or instances of certain VNF types are allowed to reside in the same host or the same part of the network. Furthermore, specific locations or sets of locations can be defined where certain VNFs or VNF types may or may not be placed. To allow the Virtual Network Function Infrastructure Provider (VNFInP) to check the validity of these requests, a semantic request checking framework is proposed. This allows the VNFInP to model both its physical substrate and the SFC request as an ontology of which the consistency is checked using a semantic reasoner. We propose an optimization where only the relevant physical locations are modeled as individuals of the ontology. This ensures that the execution times are not subject to increasing topology sizes and that for reasonable SFC request and constraint sizes, the validations of the SFC request can take place in less than a second.

Acknowledgment

Niels Bouten and Maxim Claeys are funded by a Ph.D. grant of the Agency for Innovation by Science and Technology (IWT). This work was partly 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. [Online]. Available: http://portal.etsi.org/NFV/NFV_White_Paper2.pdf

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

[4] 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.

[5] 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.

[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. 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.

[8] 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.

[9] 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.

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

[11] 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.

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

[13] 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.

[14] 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/

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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

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

[21] 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