A Semantic virtualized network functions description and discovery model

Nour El Houda Nouar, Sami Yangui, Noura Faci, Khalil Drira, Saïd Tazi

A Semantic Virtualized Network Functions Description and Discovery Model*

Nour el houda Nouar[1,2][0000−0003−3169−8539], Sami Yangui[2,3][0000−0001−9756−642X], Noura Faci[4,5][0000−0001−7428−6302], Khalil Drira[2][0000−0002−4770−1563], and Saïd Tazi[1,2]

[1] University Toulouse 1 Capitole, France
[2] LAAS-CNRS
{LastName}@laas.fr
[3] University of Toulouse, INSA, France
[4] Université Claude Bernard Lyon 1, France
[5] LIRIS-CNRS
noura.faci@univ-lyon1.fr

Abstract. Network Function Virtualization (NFV) has increasingly gained importance to address some emerging networking challenges like agility and cost-effectiveness. NFV enables to run Virtualized Network Functions (VNF) on top of any generic, Commercial-Off-The-Shelf (COTS) hardware, anytime and anywhere in the network. Specific service providers offer VNFs to prospective network providers. Service providers publish VNFs in dedicated marketplaces where network providers search VNFs and instantiate them according to a pre-established service-level agreement. On top of being proprietary and specific to the service providers, the existing VNF description models include details on VNF deployment but fail to fit VNF functional and non-functional specifications. This description alters an efficient selection of the most relevant VNFs and prevents full automation of the VNFs provisioning. This paper introduces a novel domain-independent VIrtualized networK functIoN ontoloGy (VIKING for short) for VNF description and publication in federated repositories. It also proposes a semantic-based matchmaking algorithm to discover and select the most relevant VNFs that satisfy prospective VNF consumers' requests. As for validation, a prototype called Mastermyr Chest, including VIKING's instantiation along with the matchmaker in Content Delivery Networks (CDN) domain was implemented. This prototype illustrates a new way to contribute to the redesign of the CDN's traditional architecture by enabling value-added CDN service provisioning in an agile and dynamic manner. A set of experiments was run to (i) evaluate the matchmaker performances and (ii) demonstrate its accuracy and precision.

Keywords: Network Function Virtualization (NFV) · Ontology · Semantic matchmaking · Virtualized Network Function (VNF).

* Supported by the IDEX "Chaire d'attractivité" program of the Université Fédérale Toulouse Midi-Pyréenées under Grant 2014-345.

1 Introduction

Network providers increasingly rely on Network Function Virtualization (NFV) to provide necessary network features (e.g., VPN gateways, CGI NATS, firewalls). NFV is a European Telecommunications Standards Institute (ETSI) initiative[1] to virtualize network services that traditionally run on proprietary and dedicated network hardware (e.g., DHCP servers, routers) [18] [52]. The networking hardware is decoupled and replaced with software running on Commercial-Off-The-Shelf (COTS) servers and intended to implement network appliances and middleboxes via the so-called Virtualized Network Functions (VNFs) [62] [42]. NFV significantly reduces CAPital EXpenditures (CAPEX), such as equipment purchases, and OPerational EXpenditures (OPEX), such as energy consumption cost, for network service providers. Furthermore, it enables agile, flexible, and cost-effective provisioning capabilities.

The ETSI NFV architectural framework provides an open environment where VNFs can be interoperable. This framework includes one fundamental building block, namely, NFV MANagement and Orchestration (MANO). MANO manages the NFV Infrastructure (NFVI) resources and the VNFs life-cycle, as well as performs services orchestration to satisfy requests of Operations Support System (OSS), such as network service delivery, and Business Support System (BSS) such as reduce energy consumption cost. Essentially, to deploy VNFs using NFVI, such as OpenStack-Tacker [2] or OPNFV [3], the MANO requires a deployment template called VNF Descriptor (VNFD). VNFD describes VNF's deployment requirements and operations, and is used for VNF instantiation and life-cycle management, and orchestration, respectively.

1.1 Motivations and research issues

Valuable standardization initiatives (e.g., ETSI NFV [24], IETF SFC [31], and OASIS TOSCA [60]) focus on providing an intent framework and enabling VNF provisioning capabilities such as description, publication, and discovery mechanisms for VNFs. Similarly, several research papers (e.g., [44], [13], [30]) proposed description models for VNFs. However, the existing discovery approaches that rely on VNFD are still in their early ages, and much work has yet to be done for optimal VNF provisioning. First, the existing discovery approaches remain specific to the owner providers. Each provider defines specific procedures and practicecs to parse VNF descriptors and select relevant information to them. Second, consumers still need to manually select the required VNF rather than having an automated discovery mechanism. Finally, these VNF descriptions and publication models are not comprehensive. Indeed, they do include details on VNF deployment but fail to cover their associated functional and non-functional specifications. Functional characteristics of VNFs refer to the business functionality

[1] `https://www.etsi.org/technologies/nfv`
[2] `https://wiki.openstack.org/wiki/Tacker`
[3] `https://www.opnfv.org/`

that a given VNF supports/implements (e.g. video compression, data mixer). By definition, non-functional characteristics refer to what the VNF needs/requires for proper functioning. In other words, these characteristics aim to describe the optimal use state and/or the requirements of the VNF in terms of security, reliability, performance, maintainability and so on.

All these limitations are due to several reasons. In NFV providers and technologies' broad landscape, a lack of a shared understanding of VNF descriptions can undoubtedly be observed due to technologies' and providers' heterogeneity. Besides, implicit knowledge leads to possible different interpretations. Consequently, consumers are obliged to parse a priori known sources to look for VNF candidates. This is time-consuming and often results in a a minimal number of VNF candidates, not all relevant concerning consumers' initial needs.

1.2 Objectives and contributions

This paper introduces a novel approach for VNFs description, publication, and discovery, to address the aforementioned limitations. The proposed approach draws on the service-oriented computing principles. The main contributions are twofold: 1) design of a domain-independent VIrtualized networK functIoN ontoloGy (VIKING for short) that enables a comprehensive and generic description of the VNF capabilities from functional and non-functional perspectives, and 2) development of a semantic-based matchmaker that relies on VIKING to ensure the best matching between requested VNFs and published ones.

As for validation, we refine VIKING for the Content Delivery Networks (CDN) domain through an illustrative use case where VNF description and discovery are realized. The implemented prototype, called Mastermyr chest, fully automates and simplifies the VNFs discovery and instantiation procedures. Furthermore, it enables cooperation and federation between heterogeneous and proprietary providers in the NFV landscape. The performed experiments highlight that the proposed VNFs discovery algorithm is more accurate and precise than the existing semantic matchmaking algorithms in service computing. Moreover, they also show that our algorithm can discover and select the most relevant VNFs with reasonable delays and overhead. Our initiative thus constitutes an important step for paving the way to NFV use in the novel and next-generation networks such as Content Delivery Networks (CDN), Internet of Things (IoT), and the fifth-generation (5G) of mobile telco networks, and consequently, fills a considerable gap in this emerging and promising area.

1.3 Paper structure

The rest of the paper is organized as follows. Section 2 reviews the existing work in the relevant literature. Section 3 presents the design considerations and identifies the challenges that need to be tackled in this work. Section 4 introduces VIKING for VNF description and its related matchmaker for semantic-based discovery. Section 5 describes an illustrative use case in CDN. Section 6 presents

the associated Mastermyr Chest prototype. Section 7 details the performed experiments and discusses the obtained results. Finally, Section 8 concludes the paper and presents future work.

2 Related work

This Section introduces background information and discusses the related work in the literature. The first subsection details the relevant efforts on semantics for network management in general and NFV in particular. We examine NFV-based approaches that use semantics in the second subsection. Finally, in the third subsection, we review the relevant work that focus on Intent-Based Networks (IBN) management and position our findings with regard to them.

2.1 Semantics in networking

A plethora of studies in service computing field investigated services' and users' queries description. Several concepts have been studied, however, the most important results were obtained when using semantics. Handling semantics in service discovery was primarily investigated from two main matching perspectives: syntactic and semantic. The first relies on graph theory, such as Resource Description Framework (RDF) [5] and DIANE Service Description [35]. In contrast, the second relies on ontologies, such as the W3C Web Ontology Language(OWL-S) [39] and Web Service Modeling Ontology (WSMO) [21]. Many research works compare the syntactic ones, exemplified by information retrieval metrics, *versus* the semantic matching ones, illustrated by logic inference (e.g., see [9] [53]). The latter turns out more efficient than the former in terms of precision and recall. This result is one of the reasons that led us to advocate for semantic matching for this work.

Generally speaking, in the networking domain, the use of semantics was widely used since the late eighties (e.g., [59], [54]). Artificial intelligence and machine translation were the first to develop and use semantic networks. More broadly, the use of semantics in networks is done through declarative graphic representation that aims to represent knowledge and supports automated plans for reasoning about learning. Some approaches are highly informal, but others are formally defined as systems of logic. In particular, the reason behind semantics is to build and evolve network ontologies (e.g., [66]), retrieve information in networks (e.g., [61] for peer-to-peer networks), and network slicing and segmentation (e.g., [38]).

When it comes to highly dynamic and/or virtualized environments such as ad-hoc networks and cloud computing (i.e., the main building blocks of NFV), OWL ontologies have been massively applied. For instance, OWL ontologies have been used for cloud environments to describe the heterogeneous multi-vendor cloud resources and users' SLA in the FP7 European mOSAIC project [48]. In dynamic and ad-hoc networks, we find that all of Network Description Language (NDL-OWL) [6], Network Mark-Up Language(NML), Infrastructure and

Network Description Language (INDL) [27], Network Innovation over Virtualized Infrastructures (NOVI) [65] and Federated Infrastructure Discovery and Description Language (FIDDLE) [69] use OWL ontologies.

2.2 Relevant work related to NFV-based networks life-cycle management

We classify these work into two categories: (i) within the standardization bodies and research projects, and (ii) within academia.

2.2.1 Standardization bodies and research projects Besides the previously discussed ETSI VNFD model, one of the most known and used approaches is Topology and Orchestration Specification for Cloud Applications TOSCA-based, namely TOSCA-NFV [60]. TOSCA is a data model standard managed by the OASIS industry group. This data model is used to describe services' operations and requirements [11]. It also explains how services can be deployed and managed at runtime through management plans (workflows). TOSCA-NFV is the concrete implementation of the model applied to NFV for VNFs provisioning and management. It proposes a model to describe topologies, dependencies, and relationships between virtual applications and simplify these services' complexities rather than define VNFs capabilities and requirements. TOSCA-NFV model assumes that the VNFs are already discovered. Its main scope is to deliver orchestration and interoperability of VNFs. The same observation is valid for the IETF Service Function Chaining[14] (SFC) initiative. SFC in NFV setting relies on VNFD for VNFs description and selection. The reader should note that these procedures only support the VNFs business (functional) operations. In fact, SFC enables VNFs composition by simply matching their related operations [41].

The EU-funded project T-NOVA [71] provides a VNF marketplace that: (1) helps VNF developers describe and store network functions, and (2) assists the consumers when browsing and selecting the network functions that match their needs. T-NOVA extends the ETSI NFV description model by applying business aspects from the TMForum SID model [3]. Additional fields enable business interaction among actors that communicate through the T-NOVA Marketplace (e.g., SLA specification, pricing), besides deployment details needed to deploy the network services. The VNF/NS discovery process is conducted through the brokerage module [71], which permits consumers to search for VNFs/NSs while specifying their specific requirement in terms of network SLA.

Cloud4NFV [56] is a virtualized platform for VNFs provisioning. It aims to deliver NF-as-a-service to end customers. Cloud4NFV is ETSI-compliant with significant contributions on the modeling and orchestration aspects. On one side, Cloud4NFV processes a front-end database that stores collections of VNFs along with a high-level description (e.g., ID, name, description, location). On the other side, it handles a back-end database that stores specific VNF information necessary for the VNF deployment and configuration. Cloud4NFV provides only

[14] `https://tools.ietf.org/html/rfc7665`

deployment and configuration information and lacks to support an automated discovery process.

2.2.2 Academic research work In the academic literature, Hoyos and Rothenberg propose an NFV Ontology called NOn and a Semantic nFV Services (SnS) [30]. NOn enables the description of NFV as a high-level framework with reusable element descriptors. As the concrete semantic application of NOn to the NFV domain, SnS can be used to create explicit service descriptors. It relies on agents from different fields to parse and evaluate NFV services capabilities. However, NOn only considers the resources' functional capabilities. Furthermore, the reader should note that this approach imposes strong constraints on existing providers and assumes that they could support these agents, which may or may not be accurate.

In [44], the authors propose an ontology for NFV that describes the whole network resources, including VNFs, properties, and relationships (dependencies). The resources description is achieved through reusable semantic concepts used to construct additional rules for reasoning over the network. For instance, this could be useful to automate network topology design and deployment. Although this work proposes a semantic-based description model for functional VNFs operations, it mainly focuses on network engineering and integration efforts. It does not cover the VNFs discovery given specific and precise user needs.

In [13], the authors identify and discuss a set of affinity and anti-affinity constraints useful for virtualized network management. The validation of these rules is semantic-based. The addressed limitations are mainly related to service function chain requests. For instance, they defined a VNFs placement strategy that considers the network provider constraints and the chain request. This work assumes that the VNFs are already discovered and deployed.

In [12], the authors introduce Onto-NFV, an OWL-based ontology. It offers a vocabulary with its relations and constraints to describe a VNF composition (called network service) policies and the hosting NFVI policies. The policies involve information related to resource usage, VNFs precedence, and location constraints (e.g., number of CPUs, amount of memory). The authors propose NSChecker, a semantic verification system integrated into the ETSI MANO that uses Onto-NFV. The ultimate goal of this work is to detect and diagnose policy conflicts in NFV environments. For the semantic description, Onto-NFV only focuses on functional properties with no reference to non-functional properties such as security and availability. For the VNFs publication and discovery, it relies entirely on the ETSI MANO procedures. Thus, it suffers from the same issues highlighted in Section 1.1.

In [33], the authors use Network Service Description (NSD) data and ontology to automate VNFs management and network services generation. Network services consist of VNFs bound to each other through virtual links to implement shared and more general functionalities. The proposed solution relies on semantic annotation of NSD information according to ETSI NFV. This descriptor contains functional, non-functional, and optional information blocks. The

functional block provides information related to the VNFs, as well as their connection and dependencies. The non-functional block provides the whole network service's general and profile data. Finally, the optional block provides policies and monitoring information. This work addresses one of the major limitations of TOSCA-NFV. It models the relationship between parameters that could not be defined by TOSCA. However, this work provides ontology only with neither investigated reasoning technology nor discovery algorithms/procedures for VNFs.

In [29], the authors advocate for microservices architecture as the preferred option for implementing VNFs. To exploit the microservices adoption's full potential in NFV, they highlight some challenges like microservice discovery. To foster VNF dynamic scaling, the authors claim that a real-time automated service discovery mechanism should be developed to enable the required dynamic service chains. Indeed, in such setting, service discovery is critical with regard to network dynamicity (e.g., relocation, autoscaling) and frequent on-the-fly events (e.g., failures, upgrades). More in-depth details on relevant discovery patterns in the microservice context are provided in [23]. Basically, the authors define two patterns, i.e., client-side and server-side discovery patterns. Both of them assume that microservices are already known and only, their instances should be discovered. In the former pattern, the service client is in charge of determining the network locations of available service instances and defining load balancing requests across them. Specifically, the client, first, queries a service registry referring to available instances and, then, asks a load balancer to keep the best instance. A major drawback of this pattern is that the client and the service registry are tightly coupled; each programming language used on the client-side requires a dedicated logic for service discovery. In the latter pattern, the client makes a request to a service (e.g., VNF) via a load balancer. The load balancer queries the service registry and routes each request to an available service instance. Discovery details are abstracted away from the client. Clients simply make requests to the load balancer. A major drawback of this pattern is that the load balancer should be provided by the deployment environment and, therefore, should be highly available (i.e., a single point of failure).

To our best of knowldege, there are few works on semantics in the context of microservices. In [51], the authors present a framework for aligning heterogenous ontologies in order to integrate data provided by different microservices. In line with microservices' design principles (e.g., loose coupling and independent maintenance), the design of heterogeneous ontologies to describe the same domain is *de facto*. An alignment contains a set of correspondences between entities and properties of such ontologies. Correspondences refer to semantic connections between concepts used to describe microservices data. The most important difference from the traditional ontology alignment is that equivalence statements can only be obtained at runtime. Indeed, entities are created during interactions between microservices and their consumers. Therefore, it is not possible to directly access a predefined comprehensive dataset. The proposed framework dynamically loads entities provided by registered microservices so, that, alignment statements can be inferred.

2.3 Relevant work related to intent-based networks life-cycle management

IBN is an emerging research field that incorporates several techniques related to service computing, machine learning, and network orchestration to automate administrative tasks across the network [15]. IBN could be relatively compared to NFV-based work for VNFs life-cycle support and management. Among IBN controllers, we can cite Open Network Operating System (ONOS) [2], Network Intent Composition (NIC) OpenDayLight [1], and NEtwork MOdeling (Nemo) [70].

In [28], the authors propose an intent-based virtual network management platform based on Software-Defined Network (SDN). This framework aims to automate the management and the configuration of virtual networks based on high-level specifications of intents. Three types of intents are considered (topology intent, endpoint intent, and chain intent). Specifically, this work processes a mapping between the intent and the vocabulary store containing the relevant virtual network information. To this end, it relies on the discovery protocol proposed in [19], representing an ontology-based knowledge that supports a semantic inference mechanism.

In [47], the authors propose a two-layer network service description model (business layer and orchestration layer) inspired from service-oriented principles. They considered Unified Service Description Language (Linked-USDL) for describing the features provided by the service in the business layer and ETSI-compliant NSD format to describe service deployment information in the orchestration layer. Concerning the network connectivity, they adopt an intent-based network modeling to request the build/destroy operations of the forwarding paths to the network controller.

In [17], the authors propose a semantic-based service composer system called CompRess. It provides a semantic user intent SPARQL expression to describe user intents to model and describe network topology. It takes as input the user intents and automatically generates the multiple services function chains comprised of VNFs. However, the chosen VNFs are selected based on automatic reinforcement learning and VNF type. The authors do not discuss any VNF discovery approach in this work.

In [34], the authors propose an intelligent Network Deployment - Intent Renderer Application system called iNDIRA. It offers a service description framework that enables users to express their intents in a natural language. To understand, interact, and create the required network services, they proposed an automatic conversion into RDF semantic. However, iNDRIA does not support nor provide any management procedures to deploy and manage the network functions.

2.4 Synthesis

Table 1 sums up the most relevant studied work concerning VNF decription, publication, and discovery. The literature study shows that several work (e.g., [41], [71]) tried to extend the VNFD proposed by ETSI with additional information using

different approaches. In addition to VIKING, only very few work (i.e., [71], [33]) succeeded in covering both the functional and non-functional properties of the VNFs in their proposed description models. The reader should note that the description of the non-functional properties in T-NOVA is limited. It only involves the business information (e.g., cost, SLA) necessary for interaction with other T-NOVA actors. Yet another observation related to the VNF description is the popularity of OWL as the most used semantic language to describe VNFs in the literature (i.e., [6] [27] [65] [69] [12]).

Table 1: Related work evaluation synthesis

Reference	**Description**		**Publication**	**Discovery**
	Functional properties	Non-Functional properties	Interoperability	Semantic matchmaking
ETSI VNFD	Yes	No	No	No
OASIS TOSCA NFV [60]	Yes	No	Yes	No
IETF SFC [41]	Yes	No	No	No
T-NOVA [71]	Yes	Partially	Yes	No
Cloud4NFV [56]	Yes	No	No	No
Hoyos et al. [30]	Yes	No	No	No
Oliver et al. [44]	Yes	No	No	No
Bouten et al. [13]	Yes	No	No	No
Bonfim et al. [12]	Yes	No	No	No
Kim et al. [33]	Yes	Yes	No	No
VIKING	Yes	Yes	Yes	Yes

When it comes to VNF publication, the conducted study highlights that most of the existing models require VNF publication in dedicated and proprietary repositories. T-NOVA and VIKING are the only approaches that do not impose any compatibility constraints on the provider side and enable NFV repositories federation. Since both rely on generic and unified semantic models, this eliminates dependencies related to technologies used when offering the VNFs to prospective consumers.

In addition, this study shows that all the existing work, except VIKING, either did not address the discovery process or propose simplistic procedures for the discovery phase. These procedures are often characterized by manual VNF selection or automated syntactic-based matchmaking between the offered VNFs and the required ones. In other cases, the studied work entirely rely on ETSI MANO for the discovery and the deployment of the VNFs. Thus, they all suffer from the same issues highlighted in Section 1.1. Generally speaking, the studied discovery approaches require solid domain knowledge, are time-consuming, and are not always efficient. These discovery procedures considerably decrease the agility and cost-effectiveness that one may expect from a virtualized network ecosystem.

When it comes to the case of VNFs implemented as microservices, the studied papers put emphasis on VNF instances discovery rather than on microservices discovery. To our best of knowledge, there is still no such ontology for microservices discovery.

Concerning the IBN, the conducted literature study shows that this emerging field's objectives are promising. However, the reader should note that, by definition, IBN incorporates SDN-like capabilities to manage the network control plane (dynamic routes and flow configuration) as well as machine learning and artificial intelligence to support the network functions life-cycle management. It may add a certain complexity and overhead to the network. To the best of our knowledge, there are still no studies today that evaluate and compare business and operational costs of integrating IBN to network providers' ecosystem. Moreover, the value added of IBN is relative and remains dependent from several parameters such as the network application domain, data quantity and relevance to cite a few. In fact, IBN efficiency remains mainly based on the quality of the machine learning and the accuracy of predictions. The use IBN is particularly suitable for networks where data analytics and machine learning permit to shed light on recurrent patterns and trends (e.g., closed and proprietary networks). This leads to take appropriate actions. IBN approach could be less efficient and precise in ad-hoc environments like CDN, cloud and edge systems. Finally, the reader shoud note that IBN is still at its early stages, while NFV is becoming more broadly adopted nowadays by network providers (e.g., CISCO Systems, Netflix, Ubicity, VMware, Nokia, Intel Corporation, Huawei Technologies, IBM, Brocade, Vnomic, NetCracker).

3 Design considerations and challenges

NFV is at the crossroad of networking and service-oriented computing research fields for many reasons. VNF falls into the definition of IT services at large [42]. NFV aims at provisioning the network funtions through the VNF concept. VNFs could be provided in the same way as any other kind of services such as telco or Web services. In fact, Service-Oriented Architecture (SOA) principles (e.g., service abstraction, discoverability, and composability) [63] [45] could ensure the viability of an ecosystem of network services that are dynamically and flexibly provisioned, thereby coping with changeable network provider (i.e., the service consumer in this case) needs and dynamic Quality of Service (QoS) requirements along with context conditions.

Similarly, the VNF life-cycle phases are directly inspired by the service provisioning life-cycle detailed in [72]. Although the life-cycle phases are the same, the reader should underline that the way these phases are implemented remains specific to web services' operating procedures. Therefore, these implemented phases fail to support the particularities and practical differences with the VNFs. Fig.1 and Fig.2 depict the respective structures along with the fundamental contrast between them.

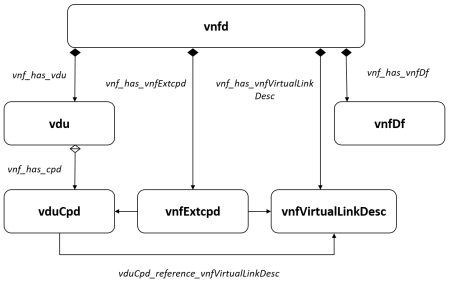

Fig. 1: VNF descriptor structure

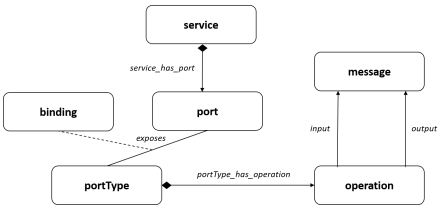

Fig. 2: WSDL descriptor structure

Fig. 1 is extracted from [43]. It shows a synthetic overview of the structure and format of a valid VNF descriptor. The VNFD (*vnfd*) is composed of one or many virtual deployment units (VDU) for hosting VNFs. A VNF could consist of several and distinct VNF Components (VNFC). Each VDU can support specific deployment resources and operation behavior and, consequently, hosts one or more VNFC. Specifically, a VDU describes mainly the Virtual Compute (VC), Virtual Storage (VS), and Virtual Memory (VM) resources. These resources' data are necessary for deploying a VNFC. VNFC can be linked via either connection points to local VDUs (*vduCpd*) or external connection point to VDUs that belong to other VNFs (*vduExtCpd*). The virtual links in the VNFD (*vnfVirtualLinkDesc*) indicate how the VDUs are connected and via which connection points (*vduCpd*). The deployment flavor (*vnfDf*) describes a specific template/image of a VNF with capacity and performance requirements. For instance, we consider a VNFD describing a VNF that implements a virtual and configurable IoT gateway. This VNF should be instantiated and deployed in a given network in between sensing devices and Web applications. The role of this VNF is to convert and format the data messages since the IoT devices and the IoT applications support different communication protocols. This VNF consists of several VNFCs where each implements a translation method from specific and proprietary communication protocol (e.g., COAP, MQTT) to HTTP. Several VDU and deployment flavors are needed to adapt the VNF to support various configuration and deployment scenarios, such as with/without data persistence or variation in the data transmission delay.

Fig. 2 is inspired from [67]. It schematizes the key elements of a valid WSDL descriptor and the relationships between them. A Web service exposes a Unified Resource Identifier (URI) (*service*) and a listening port (*port*) bound to one or several operating ports (*portType*). Each specified operating port corresponds to a business or management operation (*operation*) implemented/supported by the Web service. The service consumers invoke operations through remote calls (*message*). Depending on the service implementation, remote input message(s) that convey required input parameter(s) are sent to the appropriate operation through the corresponding operating port. Similarly, the service execution result(s) are sent back through output messages following the same route. Let us

consider a currency converter Web service hosted on a remote Web server. This Web service can be reachable to prospective consumers through a public address (i.e., URI and listening port). One sends a conversion request to the Web service using this address and specifying the relevant operation (e.g., conversion), as well as, required input parameters (e.g., base currency, target currency, amount). After the concrete service execution is performed on the Web server-side, the results are conveyed through an output message sent to the service consumer .

The discrepancy between the Web services and VNFs description models and operating procedures shows clearly that it is not appropriate to recall existing service description/discovery approaches and simply adapt or extend them to address appropriate and proper approaches for NFV. On a more general note, Table 2 sums up the fundamental differences between VNFs and Web services operations for every single phase of the life-cycle.

Table 2: Dissimilarities between Web services and VNFs life-cycle implementation

life-cycle phase	**Web Services**	**VNFs**
Design/ Development	Deployable that consists of artifact and source code. Web service deployable is simple code that needs to be hosted and executed within Web servers	Deployable that consists of standalone artifact capable of running in a serverless fashion
Description	Web Service Description Language (WSDL)	VNF Descriptor (VNFD)
Publication	Universal Description Discovery and Integration (UDDI)	Proprietary VNF repositories
Discovery	Manual (by user) or automatic (using matchmakers)	Manual (by user) or automatic (using matchmakers)
Instantiation	No instantiation is required. Web services are invoked as remote resources through valid Unified Resource Identifier (URI)	A copy of VNFs are downloaded, installed, and configured in a target domain within a network topology
Execution	Remote Procedure Call (RPC)	Local calls from the network domain

On one side, Web services are designed to be hosted and executed by service containers (e.g., Apache Tomcat, Apache Axis2). Consequently, minimal resources are packaged within the artifact. Specifically, the Web services are deployed only once over the hosting service containers and can be simultaneously invoked by several end-users. To this end, end-users rely on the information in the WSDL. The latter is stored in a centralized and unique UDDI. The discovery of the most relevant associated WSDL could be either syntactic or semantic. In all cases, Web services are never downloaded and instantiated in the local domain.

On the other side, VNFs are first designed and developed before being published in appropriate repositories for prospective consumers. VNF artifacts are standalone and need to properly incorporate all the necessary resources to execute the VNFs (serverless distribution). Before their deployment, VNFs are instantiated from proprietary marketplaces into the target network. After, they are configured to be integrated as part of a specific topology. Once deployed, VNFs are executed and, when necessary, are subject to management considerations at runtime (e.g., scale-up/down, migrate).

The VNF consumers rely on the information provided in the descriptors of the published VNFs to discover and select the most suitable ones that match the best to their needs and QoS requirements. The VNF description is a critical step where several discovery aspects need to be considered such as the VNF's business functionality, as well as, the perfect matching between the VNF's technical requirements for its deployment and the target (hosting) environment capabilities. Novel description and discovery mechanisms should take into consideration these specificities and differences with regard to Web services. For instance, unlike Web services [20], VNFs are not remotely invoked but are downloaded and executed locally, part of a given network topology. In addition to inputs/outputs parameters, a VNF description should contain more elaborated technical details such as supported technologies and VNF settings. Moreover, the interaction is not limited to basic operations like the case with Web services [67]. It should also include additional sophisticated operation management dedicated to each VNF. The existing service standards, studies, and frameworks do not address deployment-related issues. Moreover, they are process-driven, while VNFs are data-driven [67]. This interaction makes existing WS-related solutions for description and discovery, including the ones that are based on semantics, inadequate for operation in the NFV setting.

Yet another challenge is related to the substantial heterogeneity of the VNFs. Indeed, VNFs implement diverse and various network functions at either IP-level (e.g., firewall, NAT) or application-level (e.g., video mixer, virtual IoT gateway) [62] [42]. Furthermore, the functionalities supported by the VNFs could belong to very different domains (e.g., Telco, IoT, cloud, big data, multimedia). This heterogeneity makes the design of a standard description model challenging.

Finally, the last challenge is related to the potential environment's nature, where discovered VNFs should be instantiated and deployed. In fact, the end hosting nodes range from powerful computing servers to virtual machines and smartphones [18] [62]. Since nodes have different capabilities (e.g., CPU, RAM, graphics resolution, bandwidth), this implies that additional checking of the correct matching between the non-functional requirements of the discovered VNFs and the potential hosting nodes' characteristics needs to be integrated into the discovery procedure.

4 A semantic approach for VNF description and discovery

This Section first details VIKING ontology for semantically describing VNFs' capabilities from functional and non-functional perspectives. Then, it presents our VIKING-based matchmaking algorithm to discover the most relevant VNFs given specific network needs.

4.1 VNF description model

VIKING is an OWL-based (Ontology Web Language) ontology that allows describing VNFs. To design VIKING, we first determine what domain VIKING will cover (namely, network function virtualization), for what VIKING will be used (namely, VNF description, publication, and discovery), and for what types of queries VIKING should provide answers (namely, similarity and correlation). We then tackle the abstraction exercise by identifying the main common concepts shared by various application domains like CDNs, IoT, telco, and 5G networks. *Concepts* are organized as a class hierarchy where abstract concepts will be refined with more concrete ones specific to each domain application. They are also described with properties and connected to other concepts with semantic relations. We tried not to reinvent the wheel, so we further reuse existing ontologies mainly related to VNF deployment (e.g., [30]) and billing (e.g., [10]). Since concept refinement and instantiation are domain-dependent, they will be discussed in the illustrative use case presented in Section 5.

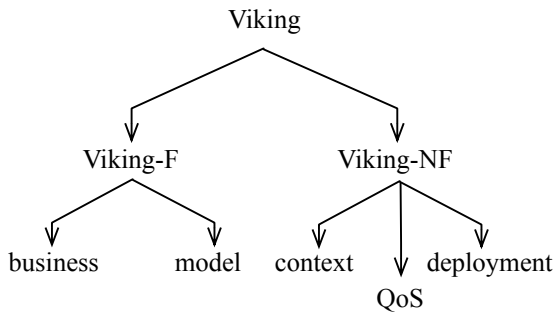

Fig. 3: A high-level view of VIKING design

To assist VNF providers when creating comprehensive and consistent VNF descriptors, VIKING relies on OWL's reasoning principles. Fig. 3 depicts VIKING's high-level skeleton that consists of two interrelated ontologies, namely VIKING-F and VIKING-NF, related to VNF's functional and non-functional properties, respectively.

On the one hand, VIKING-F refers to the formal specification of what precisely the VNF can do. It revolves around two dimensions known as **Business** and **Model**. **Business** denotes the VNF's type, inputs (i.e., details about the content upon which the VNF will take effect along with other necessary information), and outputs (i.e., details about the changes that will take place in the content). **Model** indicates the set of operations that ensure these inputs' conversion into outputs along with the related techniques and/or standards.

On the other hand, VIKING-NF refers to the formal specification of what precisely the VNF needs/requires for proper functioning. It revolves around three dimensions known as **Context**, **QoS**, and **Deployment**. **Context** refers to the necessary runtime information (e.g., operating system, specific libraries, and/or system packages), as well as, device types (e.g., smartphones, TVs, desktops) upon which the VNF's outputs can be readable. **QoS** specifies common quality features offered by the VNF (e.g., response time, operation cost) and can be refined with specific-domain ones (e.g., surrogate servers locations for CDN, the bandwidth for 5G applications). Finally, **Deployment** involves VNF's artifact and configuration parameters that are needed for VNF's execution.

Fig. 4 shows a more detailed view of VIKING dimensions. Each dimension encompasses abstract conceptual areas that are instantiated using concrete concepts, producing a dedicated VIKING-F and VIKING-NF ontologies. These concepts, as well as, the relations between them are discussed in-depth in the rest of this Section.

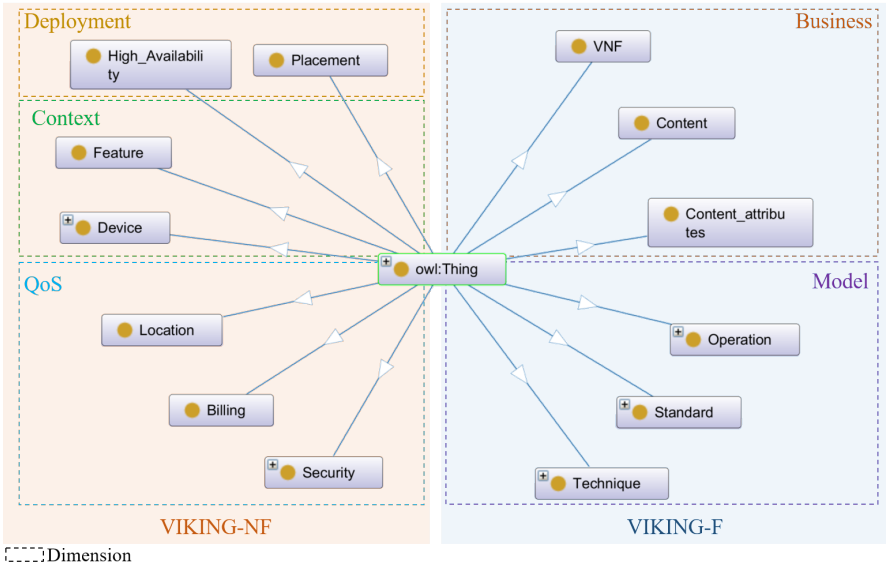

Fig. 4: VIKING's core concepts

4.1.1 VIKING-F ontology

As mentioned earlier, VNF's functional properties are specialized into **Business** and **Model** dimensions, described as follows.

Business. This dimension relies on existing classification standards (e.g., ISO/IEC[4], ETSI NFV[5]) and leading service providers. Obviously, VNF design is always related to a target application domain. The VNF business description consists of three main concepts, namely, *VNF*, *Content*, and *Content-Attribute*, along with their semantic relations. *VNF* describes all necessary details on VNFs for advertisement and query-building purposes. Basically, *VNF* will be refined into concrete virtualized network functions for a given application domain. Since these functions share common concepts and semantic relations but also have their own technical specificities, they should be considered concepts rather than concept instances. *Content* refers to different domain-related artifact types manipulated by the VNFs. *Content-Attribute* indicates the type of content(s) supported by the VNF. More specifically, this concept represents the content's technical specification (e.g., required/supplied *Resolution* and *Quality*). It is worth noticing that *VNF*, *Content*, and *Content-Attribute* are semantically connected with relations, namely, ***delivers*** between *VNF* and *Content*, and ***requires/supplies*** between *VNF* and *Content-Attribute*. The first relation states that any VNF provides some content, while the second relation captures the input/output attributes upon which the VNF will act for specific content. Besides, for consistency purposes, cardinality restrictions (e.g., *at least one*) and axioms (e.g., *disjoint*) are specified, so that concept instances are related to the right instance(s) and belong to the right concepts. To ensure a consistent instantiation of concepts, Semantic Web Rule Language (SWRL) rules (including axioms) help enforce restrictions on attribute values and semantic relations, as well. Hereafter, we only exemplify SWRL rules referring to concepts, while those referring to instances will be discussed in Section 5.3. For example, Equation 1 formally reflects the following statement: "*Any VNF (?x) that requires content-attribute (?y) should deliver specific content (?z)*".

$$\begin{aligned} & \mathit{VNF}(?x) \wedge \\ & \boldsymbol{requires}(?x, \mathit{content\_attribute}(?x, ?y)) \\ & \quad \rightarrow \ \boldsymbol{delivers}(?x, \mathit{content}(?x, ?z)) \end{aligned} \tag{1}$$

Model. Technical aspects are relevant when making content exchangeable and adaptive in heterogeneous networks and devices (e.g., be able to read video in one digital encoding format different from the original video format). We thus rely on these aspects to identify three main concepts related to **Model**, namely, *Operation*, *Standard*, and *Technique* linked to *VNF* through ***implements***, ***supports***, and ***applies*** relations, respectively. Specifically, *Operation* refers to how a VNF changes on some content(s) described in **Business**. *Standard* contains different standard(s) in the target application domain to foster content exchanges.

[4] `https://www.iso.org/standard/68291.html`
[5] `https://www.etsi.org/technologies-clusters/technologies/nfv`

Technique encompasses methods and procedures that a specialized VNF applies to make the necessary changes to the content.
Furthermore, as in **Business**, restrictions and axioms such as "Any VNF can apply some techniques" might be defined. Also, in some cases, mapping **Business** onto **Model**, or vice versa, is required (e.g., matching VNF requests with VNF advertisement). To this end, SWRL rules are defined to infer new semantic relations between instances during concept instantiation. For instance, Equation 2 formally reflects the following statement: "*Any VNF (?x) that applies technique (?y) should implement a specific operation (?u)*".

$$\begin{aligned} & VNF(?x) \wedge \\ & \boldsymbol{applies}(?x, technique(?x, ?y)) \\ \rightarrow\ & \boldsymbol{implements}(?x, operation(?z, ?u)) \end{aligned} \tag{2}$$

Table 3 sums up the defined relations between the *VNF* concept and the rest of the VIKING-F concepts.

Table 3: Relations in VIKING-F with the concept *VNF*

Dimension	Relation	(Target) Concept
Business	***delivers***	*Content*
	requires/supplies	*Content_attribute*
Model	***implements***	*Operation*
	supports	*Standard*
	applies	*Technique*

4.1.2 VIKING-NF ontology

As mentioned earlier, VNF's non-functional properties are specialized into **QoS**, **Context**, and **Deployment** description parts, described as follows.

QoS. This dimension consists of three concepts: *Location*, *Billing*, and *Security* linked to *VNF* through ***locates***, ***costs***, and ***ensures*** relations, respectively. *Location* refers to VNF's placement (e.g., network domain). *Billing* contains pricing models similar to those defined in cloud environments (e.g., time-based, volume-based, flat rate) [40]. Last but not least, *Security* is related to *VNF* regardless of security mechanisms provided by the hosting platform. Indeed, the VNF should not depend on the hosting platform that can be itself a source of threats (e.g., malicious orchestrator or administrator) and thus ensure its own security compliance to ETSI NFV SEC recommendation [4, 36]. Many existing security ontologies have been proposed in the literature, each for a specific purpose like eliciting security requirements [58], certifying security claims [64],

and determining cyber-attack goals [22], to cite just a few. In this work, we deem to encompass some certification of the VNF's security capabilities into VNF discovery so that the VNF prospective consumers trust the VNF. To this end, we define *Security* with three other concepts, namely, *Security Goal*, *Security Requirement*, and *Security Property*, as depicted in Fig. 5. The first specifies what the VNF should prevent, while the second describes what should happen in some specific situation. A *Security Goal* can also be associated with *CIA* (stands for *Confidentiality*, *Integrity*, and *Availability*) properties. Each *CIA* property refers to protection mechanisms (e.g., cryptography, signature, and redundancy) for the VNF's capabilities along with raw/processed data against intrusions. It is worth noticing that *Security Requirement* as a constraint contributes to the satisfaction of a *Security Goal*. To meet *Security Requirement*, the VNF puts in place different *Defense* types like *Authorization*, *Authentication*, and *Trust-based*. *Authorization* refers to control access to the VNF and its data along with capabilities, respectively, by an authorized entity in an authorized manner (e.g., Role- and Identity-based mechanisms). *Authentication* refers to verification mechanisms (e.g., public key, certification, and password) for checking a source's identity, including traffic provenance. *Trust-based* refers to evaluation mechanisms (e.g., direct and collaborative trust-based) to establish trust relationships between VNFs. Finally, we refine *Security Property* into *Auditability* and *Accountability*. *Auditability* refers to VNF examination techniques (e.g., knowledge- and behavior-based), while *Accountability* refers to internal tracking mechanisms (e.g., logging) to monitor the VNF's activities. Note that this ontological model for capturing the VNF's **QoS** aspects, including security, can be easily enriched with more sophisticated ones based on the application domain. For instance, one might consider extending the **QoS** dimension with additional attributes like *performance* and *adaptability* to cite a few.

Context. This dimension encompasses two main concepts, namely *Device* and

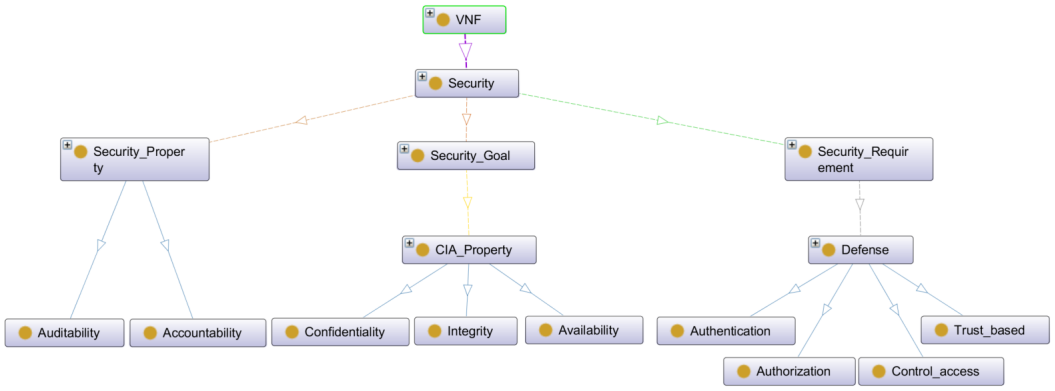

Fig. 5: Security concept

Feature. *Device* refers to additional details related to surrounding/target appliances (e.g., hosting machine), and *Feature* refers to options provided by the VNF (e.g., resize multimedia content in CDN, and switch communication protocol in IoT). To illustrate SWRL rules, Equation 3 formally reflects the following statement: "*Any VNF (?x) that implements operation (?y) and covers some device (?z) should supply a specific resolution (?u)*".

$$\begin{aligned} & VNF(?x) \wedge \\ & \textbf{\textit{implements}}(?x, operation(?x, ?y)) \wedge \\ & \textbf{\textit{covers}}(?x, device(?x, ?z)) \\ \rightarrow\ & \textbf{\textit{supplies}}(?x, resolution(?z, ?u)) \end{aligned} \tag{3}$$

Deployment. This dimension integrates the already existing ETSI VNFD. It enricheses it with additional/complementary details on the resources to be allocated for VNF's hosting and execution (e.g., number of required CPUs, amount of RAM), as well as its high-availability. Specifically, *Placement* involves a URI of a remote enriched ETSI VNFD. Undeniably, *Placement* is a mandatory property during VNF discovery. Last but not least, *High-Availability* refers to attributes like what type of redundancy, how much redundancy, and resource requirements for redundancy as per ETSI recommendations [26]. Since ensuring high-availability improves the VNF's security, specifically availability (e.g., [16] and [37]), we deem to link *High-Availability* and *Availability* (CIA property) with **increases** relation.

Table 4 sums up the defined relations between the *VNF* concept and the rest of the VIKING-NF concepts.

Table 4: Relations in VIKING-NF with the concept *VNF*

Dimension	**Relation**	**(Target) Concept**
QoS	***locates***	*Location*
	costs	*Billing*
	ensures	*Security*
Context	***covers***	*Device*
	offers	*Feature*
Deployment	***refers_to***	*Placement*
	warrants	*High-Availability*

Finally, it is well known that SWRL rules related to non-functional properties are domain-specific. Consequently, they will be defined in Section 5.3.

4.2 VNF discovery model

We propose a novel discovery model based on VIKING. VNF discovery process consists of two main steps: user request building and semantic matchmaking. First,

this process starts with assisting the user (i.e., a network provider) build his/her VNF requests in terms of what VNF capabilities are required. Afterward, it calls for a semantic matchmaking algorithm to seek candidate VNFs offered by the providers in the appropriate repositories. Finally, the discovery process provides the user with the most relevant VNFs based on their preferences. The algorithm and methodology that implement each one of these steps are detailed as follows.

4.2.1 User request building

We define functional/non-functional requirement ($\mathcal{REQ}^F/\mathcal{REQ}^{NF}$) as a set of concepts in VIKING-F/VIKING-NF requested by the user ($\mathcal{U}_i$). Formally, Equation 4 represents the syntax used for specifying $\mathcal{REQ}^F$.

$$\mathcal{REQ}_i^F = VNF(?x)\ [\wedge\ Concept_j(?x, y)]_{j=1..n} \tag{4}$$

where

- $Concept_j \in$ VIKING-F such as *Operation* and *Technique*.
- $?x$ corresponds to the *VNF* instance(s) to be retrieved. Note $\mathcal{U}_i$ can refine *VNF* into concrete concepts related to a specific domain (see Section 5).

Below, Equation 5 reflects the following $\mathcal{REQ}_i^F$: "*Any VNF (?x) that should implement some Operation (y) and require some Content_Attribute(z)*".

$$\mathcal{REQ}_i^F = VNF(?x) \wedge Operation(?x, y) \wedge Content\_Attribute(?x, z) \tag{5}$$

To specify $\mathcal{REQ}_i^{NF}$, we proceed as with $\mathcal{REQ}_i^F$ where VIKING-F is replaced with VIKING-NF. In accordance with the Web semantics principles, users can also define preferences among functional and/or non-functional requirements to select the most appropriate discovered services. To this end, we deem first to split $\mathcal{REQ}_i^F/\mathcal{REQ}_i^{NF}$ into $\{\mathcal{REQ}_{i,j}^F\}/\{\mathcal{REQ}_{i,k}^{NF}\}$ where $\mathcal{REQ}_{i,j}^F/\mathcal{REQ}_{i,k}^{NF}$ refers to $Concept_j(?x, y)/Concept_k(?x, w)$, respectively, as per Equation 4. Then, $\mathcal{U}_i$ defines her preference values for all $\mathcal{REQ}_{i,j}^F/\mathcal{REQ}_{i,k}^{NF}$. For the sake of simplicity, $\mathcal{REQ}_{i,j}^F/\mathcal{REQ}_{i,k}^{NF}$ will be classified into three preference clusters, namely, mandatory (M- $\mathcal{CL}$), high-requested (H- $\mathcal{CL}$), and optional (O- $\mathcal{CL}$). For readability purposes, Table 5 contains the notation used to formalize user requirements and preferences. After specifying all $\mathcal{REQ}_{i,j}^F$ and $\mathcal{REQ}_{i,k}^{NF}$ and labeling each requirement with a preference cluster $[\![\mathcal{REQ}_{i,j}^F, \mathcal{CL}_{i,j}]\!]$ and $[\![\mathcal{REQ}_{i,k}^{NF}, \mathcal{CL}_{i,k}]\!]$, $\mathcal{U}_i$ will define all preference values, namely, $\mathsf{Pref}(\mathcal{REQ}_i^F)$, $\mathsf{Pref}(\mathcal{REQ}_i^{NF})$, and $\mathsf{Pref}(\mathcal{CL})$. Note that all mandatory $\mathcal{REQ}_{i,j}^F$ and $\mathcal{REQ}_{i,k}^{NF}$ will serve to discard irrelevant VNFs. Note that $\mathsf{Pref}(\mathcal{REQ}_i^F) + \mathsf{Pref}(\mathcal{REQ}_i^{NF}) + \mathsf{Pref}(\text{H-}\mathcal{CL}_i) + \mathsf{Pref}(\text{O-}\mathcal{CL}_i) = 1$.
To sum up, the user request ($\mathcal{UR}_i$) is a 2-tuple defined as follows:

$$\begin{aligned}\mathcal{UR}_i =< \{[\![\mathcal{REQ}_{i,j}^F, \mathcal{CL}_{i,j}]\!]\}_{j=1,n}, \{[\![\mathcal{REQ}_{i,k}^{NF}, \mathcal{CL}_{i,k}]\!]\}_{k=1,m},\\ \mathsf{Pref}(\mathcal{REQ}_i^F), \mathsf{Pref}(\mathcal{REQ}_k^{NF}), \mathsf{Pref}(\text{H-}\mathcal{CL}_i), \mathsf{Pref}(\text{O-}\mathcal{CL}_i) >\end{aligned} \tag{6}$$

Table 5: Notation

Symbol	Description
$\mathcal{U}_i$	User i
$\mathcal{RFQ}_i^F$	$\mathcal{U}_i$'s functional requirement
$\mathcal{RFQ}_{i,j}^F$	$\mathcal{RFQ}_{i,j}^F \in \mathcal{RFQ}_i^F$
$\mathcal{RFQ}_i^{NF}$	$\mathcal{U}_i$'s non-functional requirement
$\mathcal{RFQ}_{i,k}^{NF}$	$\mathcal{RFQ}_{i,k}^{NF} \in \mathcal{RFQ}_i^{NF}$
Pref($\mathcal{RFQ}_i^F$)	User preference associated with $\mathcal{RFQ}_i^F$
Pref($\mathcal{RFQ}_i^{NF}$)	User preference associated with $\mathcal{RFQ}_i^{NF}$
M-$\mathcal{CL}$	Mandatory-preference cluster
H-$\mathcal{CL}$	High requested-preference cluster
O-$\mathcal{CL}$	Optional-preference cluster
Pref($\mathcal{CL}$)	User preference value associated with the cluster $\mathcal{CL}$
$[\![\mathcal{RFQ}_{i,j}^F, \mathcal{CL}_{i,j}]\!]$	$\mathcal{RFQ}_{i,j}^F$ is labeled with the preference cluster $\mathcal{CL}_{i,j}$
$[\![\mathcal{RFQ}_{i,k}^{NF}, \mathcal{CL}_{i,k}]\!]$	$\mathcal{RFQ}_{i,k}^{NF}$ is labeled with the preference cluster $\mathcal{CL}_{i,k}$

4.2.2 Semantic matchmaking

Algorithm 1 reflects the matchmaking logic used to return the relevant set of candidate VNFs (Cand). It relies on VIKING when matching VNFs provided in a given repository (Rep) with user requests.

Algorithm 1 consists of two types of matching, namely, `matchAll` (Line 2) and `matchSome` (Line 7). On one hand, since the set of all mandatory requirements (i.e., $\{[\![\mathcal{RFQ}_{i,j}^F, \text{M-}\mathcal{CL}]\!]\}$ and $\{[\![\mathcal{RFQ}_{i,k}^{NF}, \text{M-}\mathcal{CL}]\!]\}$) **should** be fulfilled, `matchAll` checks if the VNF exactly matches this set (i.e., true or false). Indeed, any VNF should be either kept or discarded in/from Cand depending on the result provided by `matchAll`. On the other hand, `matchSome` is applied to the rest of the user request. For each VNF in Cand, `matchSome` returns a set of matched capabilities (M-Cap) that could be empty if there is no matching at all. Finally, the Cand list will be ranked based on VNF scores.

5 Illustrative use case

The illustrative use case is implemented in Content Delivery Networks (CDN) setting. This Section briefly introduces CDN, as well as, CDN operating along with the use of NFV. The considered use case scenario description follows this introduction. Finally, the Section ends with discussing VIKING's refinement and instantiation in this specific use case.

Algorithm 1: VNF matchmaking

```
VNF-MATCHMAKING(UR_i, Rep)
foreach VNF_i ∈ Rep do
    if matchAll(VNF_i, {[[RESQ^F_{i,j}, M-CL]]}, {[[RESQ^{NF}_{i,k}, M-CL]]}) then
        append(VNF_i, Cand);
    end
end
foreach VNF_i ∈ Cand do
    score(matchSome(VNF_i, {[[RESQ^F_{i,j}, H-CL]]}, {[[RESQ^{NF}_{i,k}, H-CL]]},
      {[[RESQ^F_{i,j}, O-CL]]}, {[[RESQ^{NF}_{i,k}, O-CL]]}), M-Cap)
end
rank(Cand )
```

5.1 Content delivery networks in brief

CDN refers to a group of geographically distributed servers interacting with each other to enable fast content delivery to end-users over the Internet [50] [68]. Akamai[6], Swarmify[7], and Netflix Open Connect[8] are among the examples of CDN providers.

In addition to the primary video services, CDN providers provision value-added content. Additional services such as media management (e.g., transcoding, ad insertion, and content protection), dynamic site acceleration, and front-end optimization are injected into the raw content before delivery to end-users [8]. Enriching raw content with value-added services requires providing of the so-called middleboxes in between the media server, which hosts the content, and the end-user [55]. Middleboxes implement network functions that perform the required transformations on the raw content depending on the needs (e.g., end-users requests and preferences) and context (e.g., location and monitor capabilities). The CDN carries raw content through these middleboxes to get the needed enrichment (e.g., inject location-based ads, apply user-specified filters) before serving it to the end-users.

5.2 NFV in CDN setting

CDN middleboxes are provisioned as physical building blocks at fixed network locations and dedicated hardware [14]. The shortcomings of this traditional mode of middlebox provisioning are widely known. It is subject to a lack of automation, dynamicity, and flexibility when deploying and managing the services. Actually, for planned events (e.g., worldwide sports events such as the Olympic games or the soccer world cup), CDN can anticipate the most common prospective user requests and, consequently, predict and provide in advance the required

[6] https://www.akamai.com/
[7] https://swarmify.com/
[8] https://openconnect.netflix.com/en/

middleboxes that are needed when processing these requests. However, in the case of unplanned events, when, for instance, some videos go viral, CDNs might get short in time and can not provide the appropriate middleboxes for a specific content/location.

CDN providers leverage NFV to re-architect their traditional system architecture to provide value-added services as VNFs in agile and cost-effective ways. According to the business model introduced in [32], CDNs could interact with third-party VNF providers to get the required middleboxes. Each VNF provider handles a set of repositories where VNFs are published and stored by owners through proprietary specifications and description models. Although most of the existing VNF descriptors include the VNFD information about the VNF instantiation and deployment in the target CDN, they barely describe the VNF's business functionality (e.g., video mixing, text translation). Furthermore, they fail in describing the end-user preferences and devices' capabilities (e.g., available bandwidth, supported format, terminal type, and screen size). This actually adds more complexity to the VNF selection process and affects the relevance of the considered middleboxes with regard to the real CDN needs. Specifically, exact VNFs offered by other providers could be described with different terms and characteristics. Worse still, VNFs could be characterized with identical terms but having totally different capabilities. For instance, the "media mixer" description might refer to a middlebox with text mixing capability or sound/video mixing capability.

5.3 VIKING in CDN

As mentioned in Section 4, the upper ontologies, namely, VIKING-F and VIKING-NF, both revolve around dimensions that encompass core abstract concepts for NFV, regardless of any application domain. To produce a dedicated domain ontology, all abstract concepts in VIKING-F and VIKING-NF should be refined into concrete concepts. In this work, we target CDN as the illustrative application domain. Hereafter, we first discuss VIKING's refinement to obtain the CDN ontology, named VIKING-CDN and then, describe how to populate VIKING-CDN with instances. In the following text, abstract concepts are in italic while concrete concepts and instances are in script with upper- and lower- case first letter, respectively.

Table 6 depicts an excerpt of VIKING's refinement that results in VIKING-CDN. For **Business** and **Model**, we proceed as follows. We first refine *VNF* into *VNF4CDN* that refers to a representative set of VNF capabilities like *VConverter* and *VMixer*. For instance, *VConverter* will be specialized into `VTranscoder`. As per Section 4, VNFs implement operations acting upon content, support standards, and apply techniques. `VTranscoder` can operate on `Audio` and/or `Video Format`. For **Context**, we refine *Feature* into *Feature4CDN* while *Device* into *Device4CDN* then *SmartDevice* specialized into `Smartphone`, for instance. Last but not least, for **QoS**, *Location* was refined into *Location4CDN* specialized into `Region` and `Datacenter`.

Table 6: Excerpt of VIKING's refinement

Dimension	VIKING	VIKING-CDN
Business	*VNF*	*VNF4CDN*, *VConverter*, `VTranscoder`
	Content	*Content4CDN*, `Audio`, `Video`
	Content-Attribute	*Content-Attribute4CDN*, `Format`, `Resolution`
Model	*Operation*	*Operation4CDN*, *Conversion*, `O-Transcoding`
Context	*Feature*	`Feature4CDN`
	Device	*Device4CDN*, *SmartDevice*, `Smartphone`
QoS	*Location*	*Location4CDN*, Region, Datacenter

Table 7 shows an excerpt of VIKING-CDN's population. There are 3 instance types. The first refers to CDN technologies (e.g., `ffmpeg` and `rotate`), while the second and third refer to CDN applications (e.g., `news broadcast` and `MOOC`) and CDN administration (e.g., `western-Europe` and `Frankfurt`), respectively.

Table 7: Excerpt of VIKING-CDN's population

Dimension	Concept	Instances
Business	`VTranscoder`	`ffmpeg`, `vlc`
	`Audio`	`newsbroadcast`
	`Video`	`mooc`
	`Format`	`mp3`, `mp4`
Model	`O-Transcoding`	$\texttt{transcoding}_1$
Context	`Feature4CDN`	`rotate`, `resize`
	`Smartphone`	$\texttt{smartphone}_1$
QoS	`Region`	`western-Europe`, `north-America`
	`Datacenter`	`Frankfurt`, `San-Jose`

As stated in Section 4.1.1, SWRL rules are defined to infer new semantic relations between instances during VIKING-CDN's population. For instance, Equation 7 states that "*Any Converter (?x) that delivers a video in some video format (?y) implements transmuxing operation*".

$$\begin{aligned} &\texttt{VConverter}(?x) \ \wedge \ \boldsymbol{delivers}(?x, video) \ \wedge \ \texttt{Format}(\texttt{Video}, ?y) \\ &\quad \wedge \ \boldsymbol{supplies}(?x, ?y) \ \rightarrow \ \boldsymbol{implements}(?x, \texttt{O-Transmuxing}) \end{aligned} \tag{7}$$

The reader should note that VIKING-CDN's reasoner encompasses OWL reasoner, VIKING's axioms (e.g., *disjoint*), and its own domain-specific SWRL rules (e.g., Equation 7), as well. These SWRL rules are defined as part of VIKING-CDN to ensure consistent instantiation of concepts. For instance, Equation 8 formally reflects the following statement: "*Any Transcoder (?x) that implements Transsizing operation and covers an iPhone 10 device should supply a 2048p resolution*".

$$\begin{aligned} \texttt{VConverter}(?x) \ \wedge \ \boldsymbol{implements}(?x, \texttt{O-Transsizing}) \wedge \\ \boldsymbol{covers}(?x, \texttt{iphone10}) \ \rightarrow \ \texttt{Resolution}(\texttt{2048p}) \ \wedge \ \boldsymbol{supplies}(?x, \texttt{2048p}) \end{aligned} \tag{8}$$

To sum-up, any VNF that implements some CDN (injected) service (e.g., appliance, middlebox) should be described according to VIKING-CDN. This way of doing enables unifying the VNF's description, publication, and discovery procedures. Indeed, the description procedures are standard and homogenized regardless of any VNF provider. The discovery methods are simplified and could even be automated. Furthermore, providers can envisage cooperation and federation between them.

6 Proof-of-concept

This Section first presents the developed Proof-of-Concept (PoC). This is followed by the description of the integration of this PoC to the ETSI NFV MANO framework.

6.1 PoC architecture

The developed PoC is called the Mastermyr Chest. Its name refers to the tool chest found in Mastermyr[9] on the Gotland island, Sweden, in 1936. This chest box contained more than two hundred objects used by Viking carpenters. Similarly, our Mastermyr Chest prototype has several instruments useful for VNFs description, publication, discovery, and so on. Fig. 6 depicts the Mastermyr Chest tools, as well as the main interactions between them. The reader should note that the Mastermyr Chest was designed and implemented in a modular fashion to be easily extended with additional tools in the future. VIKING-CDN was implemented with Protégé 2000 ontology editor[10], while Mastermyr Chest tools were developed with Java. The associated source code is available on a GitHub repository[11].

The *description tool*[12] assists VNF developers (possibly, VNF owners) to semantically describe the VNFs that are relevant to the CDN context (action 1). In accordance with the model introduced in Section 4.1, some information are mandatory, and others are not. VNF descriptors can be enriched with QoS details

[9] https://en.wikipedia.org/wiki/M\%C3\%A4stermyr_chest
[10] https://protege.stanford.edu/
[11] https://github.com/NourelhoudaNouar/VNF-Description-Discovery
[12] A demo is available at: https://drive.google.com/drive/folders/1ocJgxdP_oEVdPmQMFQlvNft7jsx7IhKn?usp=sharing

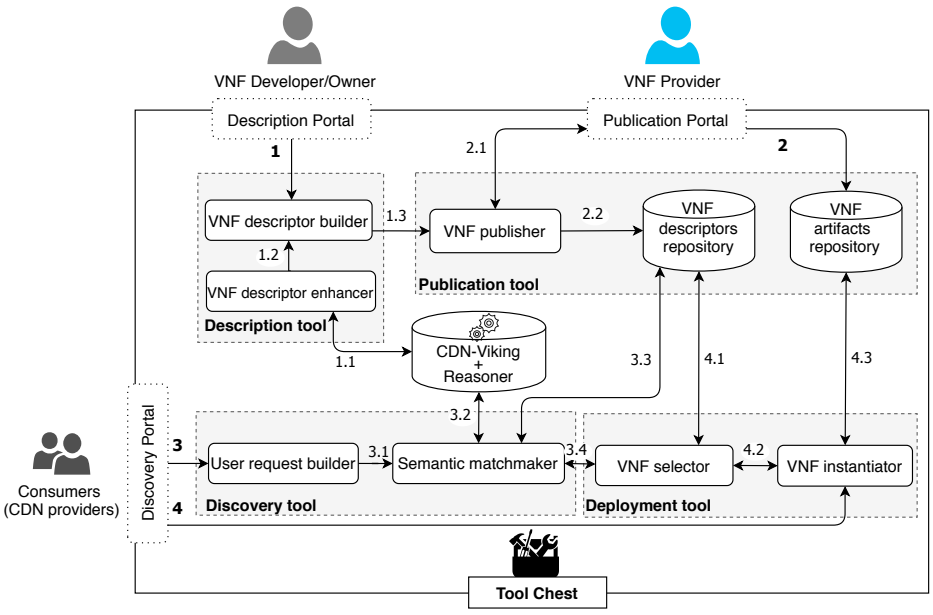

Fig. 6: The Mastermyr chest tool architecture

using the *VNF descriptor enhancer* (action 1.1). For instance, VNF developers can specify *Location* details about the VNFs that implement the CDN's surrogate servers. This would help CDN placing the popular multimedia content in the closest servers with regard to the end users location to reduce the delivery time. Afterwards, the *VNF descriptor builder* generates the VIKING-CDN-compliant descriptors of the VNFs (action 1.2) and forwards them to the *VNF publisher* (action 1.3). The VNF descriptors are implemented as OWL files. Snapshots of the *description tool* are shown in Fig. 7.

The *publication tool* enables publishing the VNF artefacts (deployables) in the *VNF artefacts repository* to make them available to CDN providers (action 2). The *VNF publisher* requests the VNF artefacts' Unified Resource Identifier (URI) (action 2.1). After that, it annotates the VNF descriptor file with this URI and saves it in the *VNF descriptors repository* (action 2.2). For the current prototype, we did consider concrete VNF artefacts that implement one of the following middleboxes:

- **A multimedia mixer** that enables mixing several multimedia contents and returns a resulting content (e.g., adding voice to a video, adding ads banner to an image/video),
- **A multimedia compressor** that enables compressing the size and quality of multimedia content (e.g., degrading a high-definition video quality to save storage space or to decrease delivery time),
- **A multimedia transcoder** that converts original multimedia content to other formats using appropriate codecs (e.g., converting MP4 video to AVI).

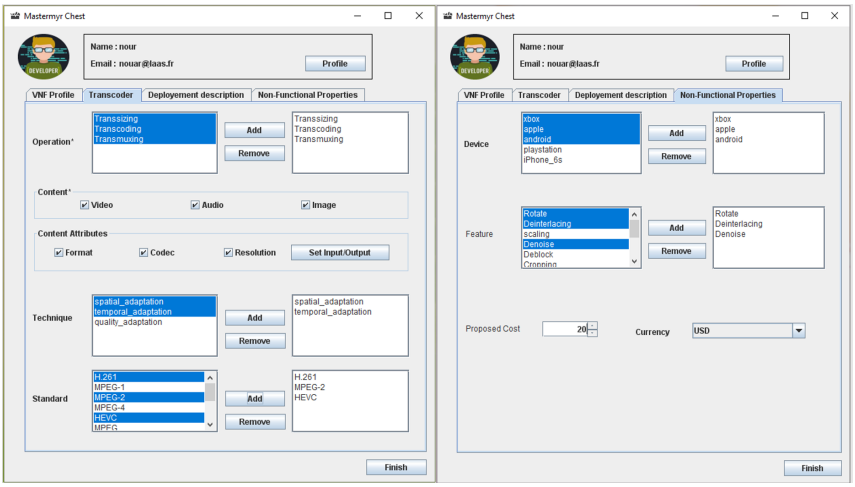

Fig. 7: Snapshots of the *description tool* interfaces

The FFmpeg[13] open-source solution was used to implement these three middleboxes as VNFs. FFmpeg involves a suite of codecs, libraries and programs to handle video, audio, and other multimedia files and streams. Several and various FFmpeg instances with different configurations and packaging are implemented, according to the characteristics and capabilities mentioned in the VNF descriptors. In turn, VNF providers store their instances into the *VNF artefacts repository* as Ubuntu-based virtual machine appliances.

The *discovery tool*[14] allows the VNF consumers (i.e., CDN providers in this specific case) to build their requests to calculate the matchmaking between required and offered VNFs (action 3). First, the *user request builder* assists VNF consumers to define a formal and VIKING-CDN-compliant request based on their functional and non-functional needs and preferences. The request is then forwarded as a required VNF descriptor to the *semantic matchmaker* (action 3.1). This *matchmaker* uses Algorithm 1 (Section 4.2.2) where `matchAll` and `matchSome` rely on VIKING-CDN's reasoner (Section 5.3) that infers relevant relationships between concepts and instances (action 3.2). Then, the *semantic matchmaker* calculates the matching scores of the requested VNF with regard to the offered VNFs descriptors published in the *VNF descriptors repository* (action 3.3). Finally, the *semantic matchmaker* transmits the obtained ranked list to the *VNF selector* (action 3.4) (e.g., see the snapshot in Fig. 17). The *semantic matchmaker* relies on OWL API[12] and Jena[13] plug-ins to parse OWL files and perform the OWL reasoning.

[13] `https://www.ffmpeg.org/`
[14] A demo is available at: `https://drive.google.com/drive/folders/1ocJgxdP_oEVdPmQMFQlvNft7jsx7IhKn?usp=sharing`.
[12] `http://owlapi.sourceforge.net/`
[13] `https://jena.apache.org/documentation/ontology/`

The *deployment tool* enables providing a published VNF in a target network topology (action 4). First, the *VNF selector* downloads and parses its VNF descriptor. Obviously, following a discovery procedure, it selects and processes the VNFs descriptor with the highest matching score (action 4.1). Then, it forwards its URI to the *VNF instantiator* (action 4.2). The latter is responsible for downloading the VNFs, deploying them in the target CDN network, configuring, and integrating them into the existing topology (action 4.3).

6.2 Software integration to ETSI NFV MANO

For dissemination and normalization purposes, Fig. 8 depicts the integration plan for the Mastermyr Chest tool built around VIKING into ETSI NFV standards. As mentioned in Section 1, the ETSI MANO framework supports VNF provisioning according to the procedures defined by ETSI and described in [24]. MANO mainly consists of three key components namely, NFV Orchestrator (NFVO), VNF Manager (VNFM), and Virtualized Infrastructure Manager (VIM). Broadly speaking, NFVO is responsible of instantiating, deploying and executing VNFs according to the strategies established by the OSS/BSS. To each provisioned VNF, MANO associates a dedicated VNFM that manages the VNF lifecycle at runtime (e.g. starting, scaling, migrating, and terminating). VNFM closely interacts with VIM that maintains the necessary compute and storage appliances from the NFV Infrastructure (NFVI) for the proper functioning of the VNFs.

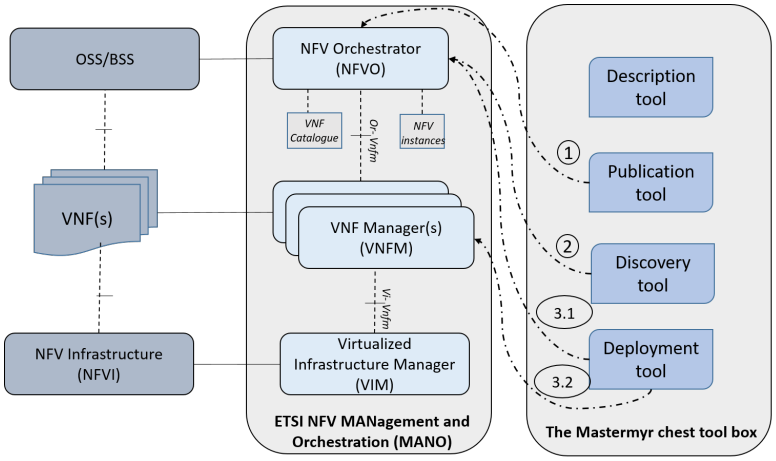

Fig. 8: The Mastermyr Chest integration to ETSI NFV MANO

The reader should note that all the tools of the Mastermyr Chest tool box were developed with respect to ETSI NFV specification and procedures. Specifically, the VIKING descriptor format is compliant with the VNFD information model specification described in [43]. The data types and communication protocols

supported by Mastermyr chest tool box are in line with the ETSI specification described in [7]. Indeed, all the tools are RESTful and compliant with the ETSI policy management generic interface.

To integrate and connect the Mastermyr Chest tools to the MANO components, we propose the following configuration. First, the *description tool* is standalone. It will assist VNF developers specify VIKING-compliant VNF descriptors.

When it comes to the VNFs publication, the *publication tool* provides the NFVO with the VIKING-compliant descriptors and their related VNF artifacts (action 1). As is presently the case for regular VNFD storage in the MANO, the NFVO stores the VIKING descriptors in appropriate VNF catalogue (e.g., document-based database) and the associated atefacts in the NFV repository (e.g., VM image).

As for the VNFs discovery, the ETSI standard procedure is kept. Specifically, the *discovery tool* first extracts a valid VNFD from a VIKING-compliant request and forwards it to the NFVO. The latter processes the request according to the standard VNFD discovery procedure supported by the MANO and sends back, to the *discovery tool*, a list of VNF candidates (action2). This list is refined thanks to our semantic matchmaker implemented within the *discovery tool* (eee Section 6.1).

The most relevant VNF with regard to the request is forwarded to the *deployment tool*. The latter first interacts with the NFVO to trigger the required VNF instantiation and the initialization of its corresponding VNFM (action 3.1). Then, the *deployment tool* communicates with the VNFM to acknowledge the creation of the VNF and start it (action 3.2).

7 Benchmark experiments

This Section discusses the performed experiments to evaluate and validate our findings. First, it describes the considered test collection and the comparative study metrics. Then, it presents the obtained measurements in terms of performance and robustness.

7.1 Test collection

To conduct experiments on VNF semantic discovery, we first proceed with the test collection creation. This collection includes three items:

1. A comprehensive set of valid VIKING-compliant VNFDs ($\mathcal{D}$) that covers conversion, mixing, and/or compression functions in the CDN domain,
2. A set of test queries ($\mathcal{Q}$) that challenges our semantic matchmaker in terms of false positive/negative outcomes,
3. A set of relevant VNFs per query ($\mathcal{V}_{\mathcal{Q}}$) that denotes all true positive outcomes.

These three items are detailed in the rest of this Section.

7.1.1 Illustrative VNFDs for the CDN use case

To achieve a comprehensive coverage for $\mathcal{D}$ (i.e., all possible valid VNFDs), we proceed as follows. First, we define the association rules listed in Table 8. These rules map VIKING's tree structure with semantic parsing onto VIKING's hypergraph structure (see Fig. 9) with syntactic parsing.

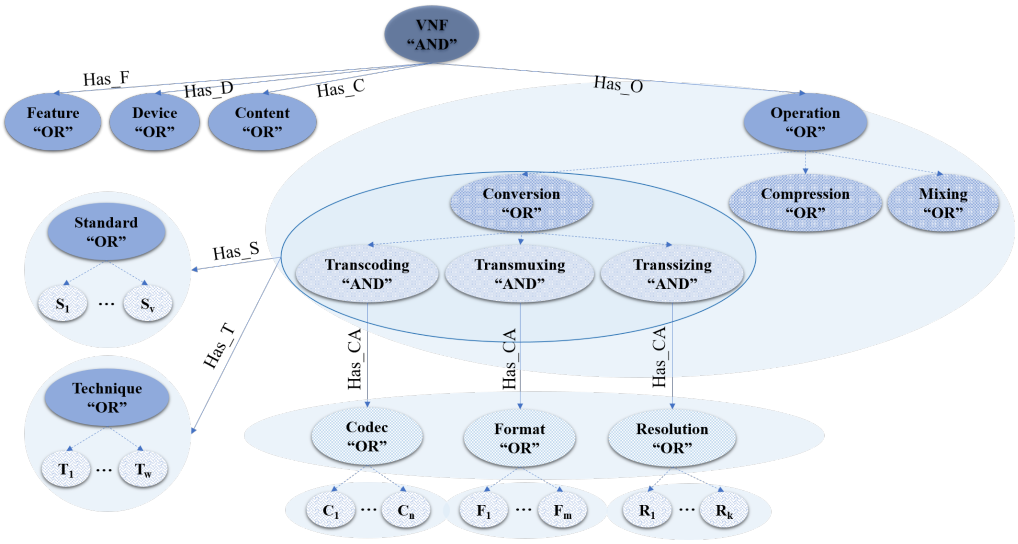

Fig. 9: VIKING's partial syntactic representation

Formally, this hypergraph $\mathcal{G}$ is defined as a 3-tuple $< \mathsf{T},\ \mathsf{NT},\ \mathsf{H} >$ where

- T denotes the multiset of terminal nodes that correspond to VIKING's instances where each set ($\mathsf{T}(\mathsf{c})$) refers to specific concept c.
- NT denotes the set of non-terminal nodes corresponding to VIKING's abstract and concrete concepts for the CDN domain (e.g., Operation and Transmuxing). We refine non-terminal into OrNode and AndNode to represent inheritance (INH) and object properties (OP) among concepts (c_j) respectively. $\{\mathsf{R}_i\}_{i=1,3}$ reported in Table 8 indicate when and how to create OrNode and AndNode.
- H represents the multiset of labeled hyperedges that refers to a set of OrNode and AndNode. Formally, H is defined as a 2-tuple $< \mathsf{NT}, \mathsf{L}, 2^{\mathsf{NT}} >$ where L refers to a set of labels like OP and/or OR (Fig. 9). $\{\mathsf{R}_i\}_{i=4,6}$ reported in Table 8 indicate when and how to create HyperEdge.

To generate $\mathcal{D}$, we adapt the well-known Depth First Search (DFS) so that possible valid VNFDs are built incrementally during visiting nodes. Algorithm 2 reflects this adaptation. This algorithm associates each visited node with some partial VNFD template where field names refer to all the parent nodes' terms.

In Lines 4-10, the algorithm splits OrNode's children into a set of nodes, each corresponding to some combination of children. In Lines 11-16, it concatenates AndNode's children partial VNFD templates. As a result, $\mathcal{D}$ contains 695 VIKING-complaint VNFDs for our CDN use case.

Table 8: Association rules

Rule	Condition	Action
R_1	$\mathsf{c} \in \mathsf{NT}$ & $\mathsf{T(c)} \neq \emptyset$	$\mathsf{createOrNode}_1(x, \mathsf{T}(x))$
R_2	$\lvert\{\mathsf{INH}(x, y_i)\}\rvert \geq 2$	$\mathsf{createOrNode}_2(x,\{y_i\})$
R_3	$\lvert\{\mathsf{OP}[x, y_i]\}\rvert \geq 2$	$\mathsf{createAndNode}(x, \{\mathsf{OP}, y_i\})$
R_4	$\mathsf{OP}[x, \mathsf{OrNode}_1(y_i, \{y_{i,j}\})]$	$\mathsf{createHyperEdge}(x, \{y_{i,j}\})$
R_5	$\mathsf{OrNode}(x, \mathsf{OrNode}_2(y_i, \mathsf{OP}, \{y_{i,j}\}))$	$\mathsf{createHyperEdge}(x, \{y_{i,j}\})$
R_6	$\mathsf{AndNode}(x, \{\mathsf{OP}, \mathsf{OrNode}_2(y)\})$[6]	$\mathsf{createHyperEdge}(x, \{\mathsf{OP}, \mathsf{OrNode}_2(y)\})$

7.1.2 Sample queries

To challenge our semantic matchmaker presented in Section 4.2.2, we build $\mathcal{Q}$ by using two types of query ambiguity introduced by Song et al. [57]. These authors classify Web queries into broad but clear and ambiguous. The former refers to queries that cover diverse "subtopics but a narrow topic," and the latter relates to queries with more than one meaning. For experimentation purposes, we refine broad-but-clear and ambiguous as follows. In the first, user requirements are specified with implicit (or hidden) terms that can be inferred by VIKING-CDN's reasoner, only. In the second, user requirements include the same naming for different instances, but only the user can remove ambiguity. Table 9 depicts 10 sample queries, and each described with $\mathcal{RE}\mathcal{Q}_i^F$ and/or $\mathcal{RE}\mathcal{Q}_i^{NF}$. For clarity purposes, we omit preferences from Table 9. On top of this, the queries are classified into either broad but clear or ambiguous depending on the source of ambiguity (Table 10).

We, thus, expect that our semantic matchmaker with broad-but-clear/ambiguous user queries as inputs and little knowledge about the user preferences will provide VNFs candidates that would correspond to probably inconsistent interpretations (i.e., false positive/negative outcomes).

7.1.3 VNFD relevance

Prior to proceeding with the set of relevant VNFDs per query ($\mathcal{V}_\mathcal{Q}$), we build some VNF template (t_j) per $\mathcal{Q}_j$ referring to expected VNF properties required

[6] For AndNode, same as OrNode_2.

Algorithm 2: $\mathcal{D}$'s generation

```
Adapted-DFS(G, queue, current_t, template, D)

// G is the hypergraph
// queue is initialized to G.root
// current_t refers to the current node in G
// template refers to a certain VNFD template
// D is the set of all VNFD produced

if queue ≠ ∅ then
    current_t = queue.pop()
    template.add(current_t)
    if leafNode(current_t) then
        D.add(template)
    end
    else if OrNode(current_t) then
        foreach subset_t ∈ current_t do
            queue.push(subset_t)
            Adapted-DFS(G, queue, current_t, template, D)
        end
    end
    else if AndNode(current_t) then
        foreach child_t ∈ current_t do
            queue.push(child_t)
            Adapted-DFS(G, queue, current_t, template, D)
        end
    end
end
```

Table 9: Sample queries ($\mathcal{Q}$)

Query	$\mathcal{RE}\mathcal{Q}_i^F$ and/or $\mathcal{RE}\mathcal{Q}_i^{NF}$
$\mathcal{Q}_1$	$VConverter(?x) \wedge C\_A(aac) \wedge C\_A(mp3) \wedge Device(Zune)$
$\mathcal{Q}_2$	$VConverter(?x) \wedge C\_A(aac) \wedge C\_A(wav) \wedge Device(iRiver)$
$\mathcal{Q}_3$	$VConverter(?x) \wedge C\_A(avc) \wedge C\_A(wmv) \wedge Device(ppc)$
$\mathcal{Q}_4$	$VConverter(?x) \wedge C\_A(mp4) \wedge C\_A(wmv) \wedge Device(pmp)$
$\mathcal{Q}_5$	$VConverter(?x) \wedge C\_A(mp3) \wedge Device(iPhone\_6s)$
$\mathcal{Q}_6$	$VConverter(?x) \wedge C\_A(mp4) \wedge Device(Galaxy_s3)$
$\mathcal{Q}_7$	$VConverter(?x) \wedge C\_A(mkv) \wedge C\_A(mp3)$
$\mathcal{Q}_8$	$VConverter(?x) \wedge C\_A(avi) \wedge C\_A(gif)$
$\mathcal{Q}_9$	$VMixer(?x) \wedge Content(video) \wedge Content(image)$
$\mathcal{Q}_{10}$	$VMixer(?x) \wedge Content(image) \wedge Content(audio)$

$C\_A : Content\_Attribute$

Table 10: Query classification

Query type	Query	Source
ambiguous	$\mathcal{Q}_1$	*Content_attribute*
	$\mathcal{Q}_2$	
	$\mathcal{Q}_3$	
	$\mathcal{Q}_4$	
broad-but-clear	$\mathcal{Q}_5$	*Device*
	$\mathcal{Q}_6$	
	$\mathcal{Q}_7$	*Technique*
	$\mathcal{Q}_8$	
	$\mathcal{Q}_8$	*Operation*
	$\mathcal{Q}_{10}$	

to satisfy this query (Table 11). To obtain $\mathcal{V}_{\mathcal{Q}}$, we automatically annotate $\mathcal{D}$ with binary relevance values. A VNF$_i$'s relevance ($\mathcal{R}$) to a certain query ($\mathcal{Q}_j$) refers to what extent this VNF's VNFD_i would be compliant with t_j (i.e., user satisfaction degree). Formally, Equation 9 computes $\mathcal{R}$ as follows.

Table 11: Expected VNF properties per query

Query	VNF template
$\mathcal{Q}_1$	$Operation(transcoding) \wedge Content(audio) \wedge C\_A(codec)$
$\mathcal{Q}_2$	$Operation(transmuxing) \wedge Content(audio) \wedge C\_A(format)$
$\mathcal{Q}_3$	$Operation(transcoding) \wedge Content(video) \wedge C\_A(codec)$
$\mathcal{Q}_4$	$Operation(transmuxing) \wedge Content(video) \wedge C\_A(format)$
$\mathcal{Q}_5$	$Operation(transsizing) \wedge Content(audio) \wedge C\_A(resolution)$
$\mathcal{Q}_6$	$Operation(transsizing) \wedge Content(video) \wedge C\_A(resolution)$
$\mathcal{Q}_7$	$Operation(transcoding) \wedge Content(video) \wedge Content(audio) \wedge Technique(audio\_separation)$
$\mathcal{Q}_8$	$Operation(transcoding) \wedge Content(video) \wedge Content(image) \wedge Technique(image\_sequence)$
$\mathcal{Q}_9$	$Operation(video\_image\_mixing) \wedge Content(video) \wedge Content(image)$
$\mathcal{Q}_{10}$	$Operation(image\_audio\_mixing) \wedge Content(image) \wedge Content(audio)$

$C\_A : Content\_Attribute$

$$\mathcal{R}^{\mathcal{Q}_j}(VNF_i) = \begin{cases} 1 & \text{if } |\{t_j.(\mathsf{c}_k|\mathsf{inst}_p) \in \mathsf{VNFD}_i\}| \geq \sigma_{\mathsf{c}_k},\ \forall c_k \in t_j \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where

- $t_j.(\mathsf{c}_k|\mathsf{inst}_p)$ refers to p^{th} instance of the concept c_k in the template t_j.
- σ_{c_k} denotes the minimum number of c_k's instances that should be included in any satisfactory VNF$_i$.

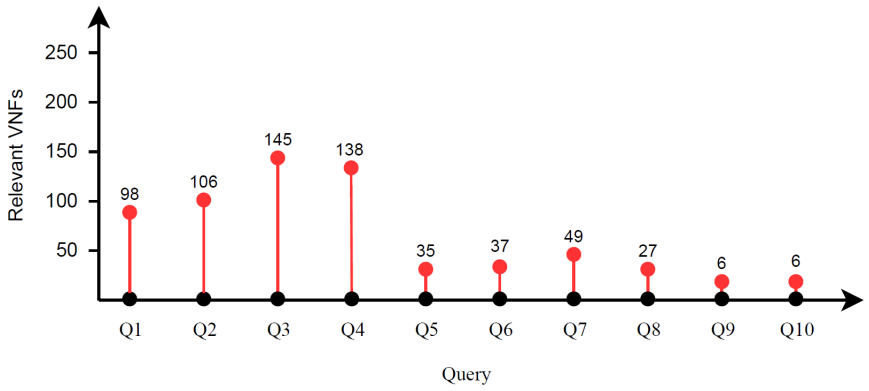

Fig. 10: Probability distribution in $\mathcal{D}$ over $\mathcal{Q}$

This annotation was performed on every VNFD in $\mathcal{D}$ for all test queries ($\{\mathcal{Q}_j\}_{j=1,10}$). Fig. 10 shows how VNFs annotated with 1 (see Equation 9) are distributed over $\mathcal{Q}$. We can observe a non-uniform distribution over $\mathcal{D}$. For instance, the set of generated VNFs satisfying queries related to conversion ($\{\mathcal{Q}_j\}_{j=1,8}$) is more represented than the other sets. This distribution is due to a significant number of possible conversion-related capabilities compared to other operations like mixing.

7.2 Performance analysis

To assess the proposed approach's performance, we use 2 metrics namely, completeness and efficiency. The former describes how well our VIKING-based matchmaker (Mastermyr chest) identifies the relevant VNFs compared with the total number of such VNFs that exist in the test collection. The latter describes how well Mastermyr chest identifies only those relevant VNFs, by comparing the number of target VNFs identified with the total number of VNFs retrieved.

7.2.1 Performance metrics

First, we classify the retrieved VNFs provided by each matchmaker into three sets, namely True Positive (TP), False Positive (FP), and False Negative (FN) where

- TP contains the **retrieved** VNFs that are relevant as per Section 7.1.3,
- FP contains the **retrieved** VNFs that are not relevant, and
- FN contains the **relevant** VNFs that are not retrieved (i.e., discarded by the matchmaker).

Once the sets mentioned above are established, we use two well-known performance measurements in the semantic Web and Machine Learning communi-

ties (e.g., [49]), namely, recall ($\mathcal{R}$) and precision ($\mathcal{P}$) that implement completeness and efficiency metrics (e.g., [46]), respectively, and are defined as follows:

- $\mathcal{R}$ refers to the ratio between the number of true positive VNFs and the number of **relevant** VNFs, including true positive VNFs and false negative VNFs, as well (Equation 10).

$$\mathcal{R} = \frac{\mathsf{TP}}{\mathsf{TP} + \mathsf{FN}} \tag{10}$$

- $\mathcal{P}$ refers to the ratio between the number of true positive VNFs and the total number of **retrieved** VNFs, including true positive and false positive VNFs (Equation 11).

$$\mathcal{P} = \frac{\mathsf{TP}}{\mathsf{TP} + \mathsf{FP}} \tag{11}$$

It happens that $\mathcal{R}$ and $\mathcal{P}$ can be inversely related. On the one hand, lower $\mathcal{R}$ increases the risk to miss relevant VNFs and, therefore, would penalize VNF providers. On the other hand, lower $\mathcal{P}$ denotes a significant number of irrelevant VNFs and, thus, would mislead the end-users. To estimate to what extent VNF providers/end-users should trust our matchmaker, we rely on $\mathcal{F}$-measure that reflects the right balance between $\mathcal{R}$ and $\mathcal{P}$. Formally, $\mathcal{F}$-measure can be defined as follows:

$$\mathcal{F}\text{-measure} = 2 \times \frac{\mathcal{R} \times \mathcal{P}}{\mathcal{R} + \mathcal{P}} \tag{12}$$

Note that Equation 12 considers $\mathcal{R}$ and $\mathcal{P}$ as equally important.

On top of the aforementioned metrics, we deem appropriate to measure the overhead in terms of response time ($\mathcal{RT}$) defined as the amount of time necessary to get a response from the matchmaker following a discovery request sent by an end-user.

7.2.2 Measurement and discussion

We run experiments that challenge the *discovery tool* of the Mastermyr chest with the test collection including $\mathcal{Q}$. To demonstrate our matchmaker's completeness and efficiency, we first consider another sample of queries ($\mathcal{Q}'$) obtained from $\mathcal{Q}$'s (partial or full) disambiguation by adding new VNF property per $\mathcal{Q}_i$, as depicted in Table 12. After, we measured *discovery tool* performance in terms of precision, recall, $\mathcal{F}$-measure, and response time. The reader should note that we consider all matching outcomes obtained by our semantic matchmaker, given some $\mathcal{Q}_i$. The obtained results are discussed in the rest of this Section.

Fig. 11 depicts $\mathcal{R}$ rates for all $\mathcal{Q}_i$ and their corresponding $\mathcal{Q}'_i$. We can observe that our matchmaker with queries $\mathcal{Q}'_5$, $\mathcal{Q}'_6$, $\mathcal{Q}'_9$, $\mathcal{Q}'_{10}$ provides better $\mathcal{R}$ rates than with $\mathcal{Q}_5$, $\mathcal{Q}_6$, $\mathcal{Q}_9$, $\mathcal{Q}_{10}$, up to 33%. For instance, for $\mathcal{Q}'_9$, $Operation(video\_image\_mixing)$ helps retrieving VNFs that implement mixing capabilities without specific $Content$. As for queries $\mathcal{Q}'_1$ to $\mathcal{Q}'_4$ and $\mathcal{Q}'_7$ to $\mathcal{Q}'_8$, we can observe constant $\mathcal{R}$ rates compared to their corresponding $\mathcal{Q}_i$. Indeed, VIKING-CDN's reasoner through SWRL rules help the matchmaker identify relevant VNFs described with minimal required

Table 12: Sample queries ($\mathcal{Q}'$)

Query	$\mathcal{RE}\mathcal{Q}_i^F$ and/or $\mathcal{RE}\mathcal{Q}_i^{NF}$
$\mathcal{Q}'_1$	$VConverter(?x) \wedge C\_A(aac) \wedge C\_A(mp3) \wedge Device(Zune) \wedge C\_A(audio\_codec)$
$\mathcal{Q}'_2$	$VConverter(?x) \wedge C\_A(aac) \wedge C\_A(wav) \wedge Device(iRiver) \wedge C\_A(audio\_format)$
$\mathcal{Q}'_3$	$VConverter(?x) \wedge C\_A(avc) \wedge C\_A(wmv) \wedge Device(ppc) \wedge C\_A(video\_codec)$
$\mathcal{Q}'_4$	$VConverter(?x) \wedge C\_A(mp4) \wedge C\_A(wmv) \wedge Device(pmp) \wedge C\_A(video\_format)$
$\mathcal{Q}'_5$	$VConverter(?x) \wedge C\_A(mp3) \wedge Device(iPhone\_6s) \wedge Content(audio)$
$\mathcal{Q}'_6$	$VConverter(?x) \wedge C\_A(mp4) \wedge Device(Galaxy_s3) \wedge Content(video)$
$\mathcal{Q}'_7$	$VConverter(?x) \wedge C\_A(mkv) \wedge C\_A(mp3) \wedge Content(audio)$
$\mathcal{Q}'_8$	$VConverter(?x) \wedge C\_A(avi) \wedge C\_A(gif) \wedge Content(image)$
$\mathcal{Q}'_9$	$VMixer(?x) \wedge Content(video) \wedge Content(image) \wedge Operation(video\_image\_mixing)$
$\mathcal{Q}'_{10}$	$VMixer(?x) \wedge Content(image) \wedge Content(audio) \wedge Operation(image\_audio\_mixing)$

$C\_A : Content\_Attribute$

VNF properties. For instance, for $\mathcal{Q}'_8$, $Content(image)$ does not help retrieving VNFs that support $Technique(image\_sequence)$ due to the SWRL rule (Equation 1) instantiated with gif and $image$.

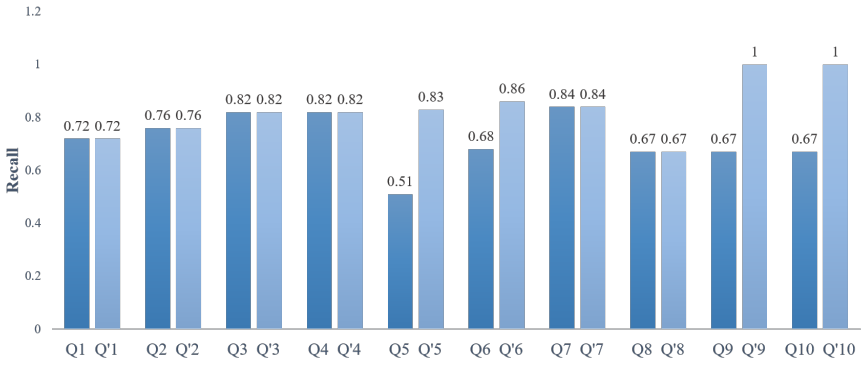

Fig. 11: Recall rates

Fig. 12 depicts $\mathcal{P}$ rates for all $\mathcal{Q}_i$ and their corresponding $\mathcal{Q}'_i$ with respect to the relevant VNFs for $\mathcal{Q}_i$ as reported in Fig. 10. Overall, we can observe that our matchmaker achieves better $\mathcal{P}$ rate with queries $\mathcal{Q}'_1$ to $\mathcal{Q}'_6$ than with $\mathcal{Q}_1$ to $\mathcal{Q}_6$, respectively, with an approximate increase up to 15%. For instance, for $\mathcal{Q}_1$, the required VNF property $C\_A(audio\_codec)$ helps overcoming the ambiguity raised by $C\_A(aac)$ as an audio format. As for $\mathcal{Q}_7$ to $\mathcal{Q}_{10}$, consid-

ering additional VNF properties do not help discarding irrelevant VNFs. For instance, $Content(audio)$ in $\mathcal{Q}'_7$ does not permit to exclude the VNFs with $Operation(transcoding)$ on $Content(video)$ **and/or** $Content(audio)$ while relevant VNFs should implement $Operation(transcoding)$ on $Content(video)$ **to** $Content(audio)$.

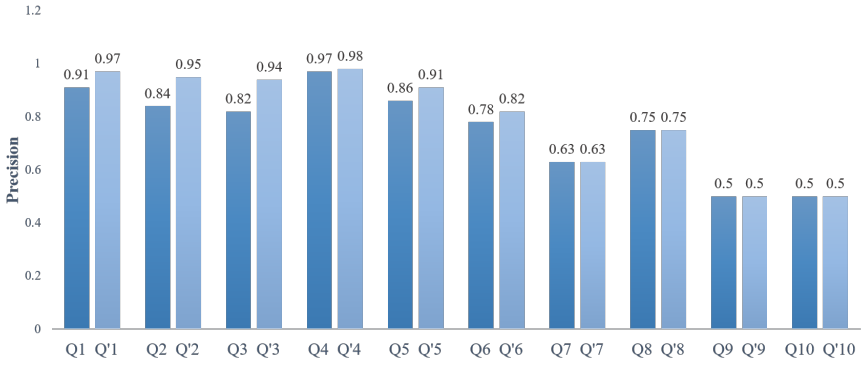

Fig. 12: Precision rates

Since improving $\mathcal{R}$ typically reduces $\mathcal{P}$ and *vice-versa*, a decision threshold (δ) should be defined to determine a good trade-off between $\mathcal{R}$ and $\mathcal{P}$. Fig. 13 illustrates the way the discovered VNFs for each query (i.e., $\mathcal{Q} \cup \mathcal{Q}'$) are partitioned. We, then, compute $\mathcal{R}$ and $\mathcal{P}$ at each δ_j per query along with the recall and precision averages for $\mathcal{Q}$ and $\mathcal{Q}'$, respectively, at δ_j. Fig. 14 depicts the trade-off as the ratio between $\mathcal{R}$ and $\mathcal{P}$. The end-user fixes either $\mathcal{R}_j$ and $\mathcal{P}_j$, or both and obtains the corresponding δ_j. For instance, whether the end-user seeks for the most relevant VNFs, only (i.e., $\mathcal{P} = 1$), δ_1 should not exceed 10% of discovered VNFs. We can observe that when $\mathcal{R}$ is improved, $\mathcal{P}$ remains satisfactory (i.e., ≈ 0.75 and 0.8 for $\mathcal{Q}$ and $\mathcal{Q}'$, respectively).

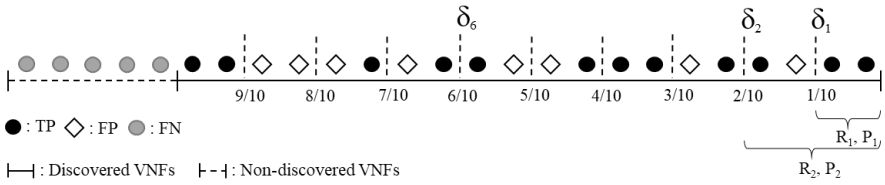

Fig. 13: Decision threshold

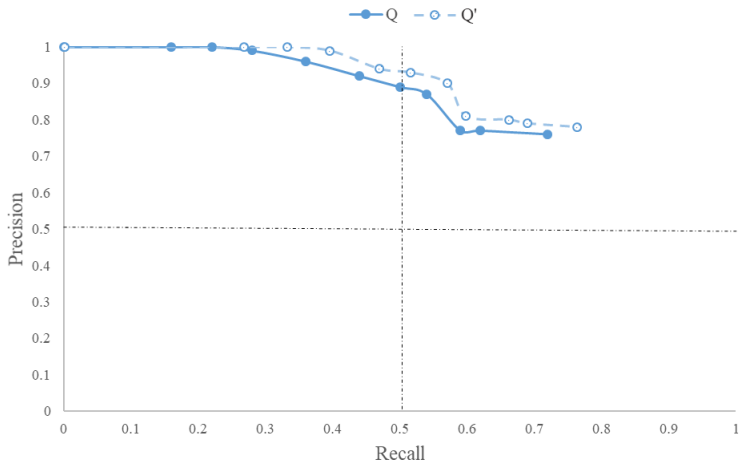

Fig. 14: $\mathcal{R}/\mathcal{P}$ ratio

Fig. 15 shows F-measures for $\mathcal{Q}$ and $\mathcal{Q}'$. Note that larger F-measures indicate better overall results. Globally, the obtained F-measures vary over [0.57, 0.89] and [0.67, 0.89] for $\mathcal{Q}$ and $\mathcal{Q}'$, respectively. This indicates satisfactory results for the matchmaker from both provider and end-user perspectives. However, some improvements are still needed in terms of additional SWRL rules related to missing relationships between *Content* and *Operation* as highlighted for $\mathcal{Q}_9$ and $\mathcal{Q}_{10}$, for instance.

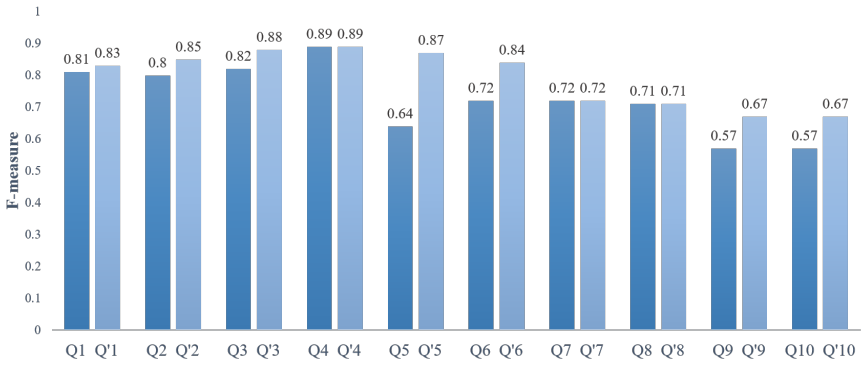

Fig. 15: F-measures

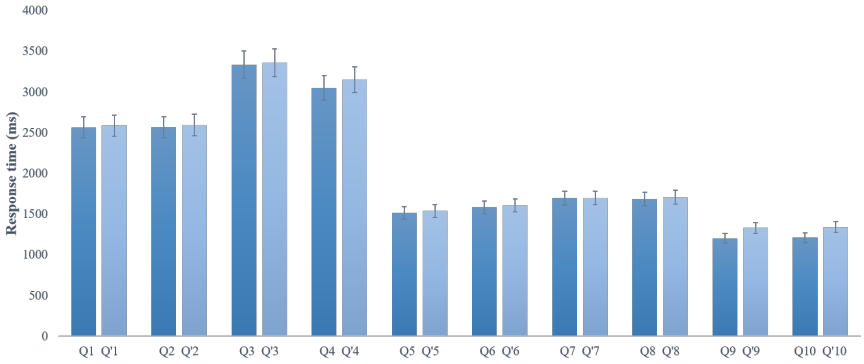

Fig. 16: Overhead

Fig. 16 depicts the overhead ($\mathcal{RT}$) for $\mathcal{Q}$ and $\mathcal{Q}'$. We notice that the overhead varies according to the query. For instance, converter queries (i.e., $\mathcal{Q}_1$ to $\mathcal{Q}_8$) takes longer time than mixer queries ($\mathcal{Q}_9$ and $\mathcal{Q}_{10}$). This can be explained by complexity underlying converter's semantics (i.e., more concepts and instances along with semantic relations) compared to mixer. This complexity, thus, induces additional time for matching. However, we observe minor increase in $\mathcal{RT}$ for $\mathcal{Q}_i$. Compared to having better $\mathcal{R}$ and $\mathcal{P}$ obtained for $\mathcal{Q}'_i$, the overhead remains acceptable (i.e., $\approx$ 60ms).

7.3 Robustness evaluation

We also evaluate our matchmaker robustness in terms of consistency. For a given query, we examine and validate the obtained VNFs and their calculated matching scores considering a specific predefined set of published VNFs. For instance, the matching results for the VTranscoder are shown in Fig. 17. We observe that VNFs ranked from 8 to 10 have the same final score. This raking would mean that these VNFs are either identical or equivalent in terms of satisfying the query and its preferences. Let us parse the following capabilities:

- $\{\mathsf{CAP}_{vt7588}{}^{\mathsf{F}}\}^H$=$\{$`O-Transmuxing`$\}$ & $\{\mathsf{CAP}_{vt7588}{}^{\mathsf{F}}\}^O$=$\{$`Std-MPEG_4`$\}$
- $\{\mathsf{CAP}_{vt9500}{}^{\mathsf{F}}\}^H$=$\{$`O-Transcoding`$\}$ & $\{\mathsf{CAP}_{vt9500}{}^{\mathsf{F}}\}^O$=$\{$`Std-MPEG_2`$\}$
- $\{\mathsf{CAP}_{vt4341}{}^{\mathsf{F}}\}^H$=$\{$`O-Transcoding`$\}$ & $\{\mathsf{CAP}_{vt4341}{}^{\mathsf{F}}\}^O$=$\{$`Std-MPEG_4`$\}$

We note that `O-Transmuxing` and `O-Transcoding` are both functional requirements and are associated with the same preferences. This description explains the equivalence between these VNFs. To evaluate robustness, we modify the preference values for `O-Transcoding` to O- $\mathcal{CL}$. Based on the updated list of scores shown in Fig. 18, we notice that the final score changes. The list is refined to better satisfy the H- $\mathcal{CL}$'s requirements. Contrary to the previous list, we see that *Tr_Virtual_Transcoder7588* is ranked before *Tr_Virtual_Transcoder9500* in the updated list.

N°	VNF Name	Final Score	FP matching	NFP matchi...
0	Tr_Virtual_Transcoder3874	78.04 %	83.0%	67.0%
1	Tr_Virtual_Transcoder5543	77.45 %	75.0%	83.0%
2	Tr_Virtual_Transcoder3177	72.25 %	75.0%	67.0%
3	Tr_Virtual_Transcoder8554	67.25 %	75.0%	50.0%
4	Tr_Virtual_Transcoder8823	67.06 %	83.0%	33.0%
5	Tr_Virtual_Transcoder5631	66.08 %	58.0%	83.0%
6	Tr_Virtual_Transcoder7411	66.08 %	50.0%	100.0%
7	Tr_Virtual_Transcoder7058	62.06 %	83.0%	17.0%
8	Tr_Virtual_Transcoder9500	61.47 %	75.0%	33.0%
9	Tr_Virtual_Transcoder7588	61.47 %	75.0%	33.0%
10	Tr_Virtual_Transcoder4341	61.47 %	75.0%	33.0%
11	Tr_Virtual_Transcoder4148	61.27 %	75.0%	33.0%
12	Tr_Virtual_Transcoder3261	61.08 %	67.0%	50.0%
13	Tr_Virtual_Transcoder7854	61.08 %	50.0%	83.0%
14	Tr_Virtual_Transcoder6413	60.88 %	67.0%	50.0%
15	Tr_Virtual_Transcoder4606	60.88 %	67.0%	50.0%
16	Tr_Virtual_Transcoder4446	60.69 %	50.0%	83.0%
17	Tr_Virtual_Transcoder6087	56.47 %	75.0%	17.0%
18	Tr_Virtual_Transcoder8916	56.47 %	75.0%	17.0%
19	Tr_Virtual_Transcoder7342	56.27 %	58.0%	50.0%
20	Tr_Virtual_Transcoder9249	56.08 %	67.0%	33.0%
21	Tr_Virtual_Transcoder3686	55.88 %	67.0%	33.0%
22	Tr_Virtual_Transcoder2878	55.88 %	58.0%	50.0%
23	Tr_Virtual_Transcoder5183	55.69 %	67.0%	33.0%
24	Tr_Virtual_Transcoder8087	55.49 %	58.0%	50.0%
25	Tr_Virtual_Transcoder9273	55.29 %	58.0%	50.0%
26	Tr_Virtual_Transcoder7450	55.29 %	58.0%	50.0%
27	Tr_Virtual_Transcoder6637	55.29 %	50.0%	67.0%
28	Tr_Virtual_Transcoder1756	55.29 %	42.0%	83.0%
29	Tr_Virtual_Transcoder6339	55.1 %	42.0%	83.0%

Fig. 17: The total ranked list of the discovered VNFs

8	Tr_Virtual_Transcoder7588	62.5 %	75.0%	33.0%
9	Tr_Virtual_Transcoder9500	62.27 %	75.0%	33.0%
10	Tr_Virtual_Transcoder4148	62.27 %	75.0%	33.0%
11	Tr_Virtual_Transcoder4341	62.27 %	75.0%	33.0%

Fig. 18: Excerpt of a list of relevant VNFs following a requirement change

8 Conclusion and perspectives

This paper introduced a novel semantic-based methodology to describe, publish and discover Virtualized Network Functions (VNFs). It proposes a domain-independent VIrtualized networK functIoN ontoloGy (called VIKING) that enables VNFs developers and owners to sufficiently describe the capabilities and the requirements of their VNFs prior to publication. Prospective VNFs consumers use the same model to automate the discovery process, improve its precision and rely on federated repositories system if needed. Yet another contribution is supporting the VNFs non-functional properties and user preferences during the discovery, in addition to the classical functional properties.

As for validation, an extensive and real-life use case was designed and implemented. The considered use case explores VIKING in the context of Content Delivery Networks (CDN). A validating chest tool called Mastermyr Chest is developed. It consists of several tools that enable describing, publishing, discovering, and instantiating VNFs as middleboxes for CDN providers. The performed experiments on the discovery tool of Mastermyr Chest show that our semantic-based approach is accurate, precise, and with moderate delays.

We plan to extend this work with describing and automatically building VNF chains using semantics in the near future. Current VNF chains are represented

through ETSI VNF Forwarding Graphs (VNF-FG) descriptors. These descriptors are manually designed by network administrators when using a straightforward model and sequential execution of VNFs. We do believe that enhancing VNF-FG descriptors with semantics could enable the automatic build of more complex and sophisticated VNF chains. Another perspective for this work is integrating our findings within the results of the newly Zero-touch network and Service Management (ZSM)[25] ETSI working group. The ZSM working group focuses on describing automation in network management and aims to deliver policy-driven automation, intent-based automation, as well as, intent-based service orchestration. We believe that this research work meets the objectives and the topics of interest of the ETSI ZSM working group.

References

1. Network intent composition. Online, `https://docs.opendaylight.org/en/stable-fluorine/user-guide/network-intent-composition-(nic)-user-guide.html`
2. Onos intent framework. Online, `https://wiki.onosproject.org/display/ONOS/Intent+Framework`
3. Tm forum information framework (sid) gb922. Tech. rep. (Nov 2013)
4. Nfv, network functions virtualisation. etsi gs nfv-sec 009 v1. 1.1 (2015-12) (2015)
5. Online (Feb 2020), `https://www.w3.org/RDF/`
6. Ndl owl (Jan 2020), `http://geni-orca.renci.org/owl`
7. Network functions virtualisation (nfv) release 3; protocols and data models; restful protocols specification for the policy management interface. etsi gs nfv-sol 012 v3.4.1. techreport (Oct 2020)
8. The streaming media magazine. Online (2020), `https://www.streamingmedia.com/Articles/Editorial/Featured-Articles/DSA-FEO-Transparent-Caching-CDN-Value-Add-Services-Demystified-85376.aspx`
9. Adala, A., Tabbane, N., Tabbane, S.: A framework for automatic web service discovery based on semantics and NLP techniques. Advances in Multimedia **2011**, 1–7 (2011). https://doi.org/10.1155/2011/238683
10. Afify, Y.M., Badr, N.L., Moawad, I.F., Tolba, M.F.: A comprehensive business domain ontology for cloud services. In: 2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS). pp. 134–143. IEEE (2017)
11. Binz, T., Breitenbücher, U., Kopp, O., Leymann, F.: Tosca: portable automated deployment and management of cloud applications. In: Advanced Web Services, pp. 527–549. Springer (2014)
12. Bonfim, M., Freitas, F., Fernandes, S.: A semantic-based policy analysis solution for the deployment of nfv services. IEEE Transactions on Network and Service Management **16**(3), 1005–1018 (2019)
13. Bouten, N., Claeys, M., Mijumbi, R., Famaey, J., Latré, S., Serrat, J.: Semantic validation of affinity constrained service function chain requests. In: 2016 IEEE NetSoft Conference and Workshops (NetSoft). pp. 202–210 (June 2016). https://doi.org/10.1109/NETSOFT.2016.7502414
14. Bouten, N., Famaey, J., Mijumbi, R., Naudts, B., Serrat, J., Latré, S., De Turck, F.: Towards nfv-based multimedia delivery. In: 2015 IFIP/IEEE International

Symposium on Integrated Network Management (IM). pp. 738–741 (May 2015). https://doi.org/10.1109/INM.2015.7140364
15. Campanella, A.: Intent based network operations. In: 2019 Optical Fiber Communications Conference and Exhibition (OFC). pp. 1–3. IEEE (2019)
16. Casazza, M., Fouilhoux, P., Bouet, M., Secci, S.: Securing virtual network function placement with high availability guarantees. CoRR (2017)
17. Chen, X., Yu, H., Xu, S., Du, X.: Compress: Composing overlay service resources for end-to-end network slices using semantic user intents. Transactions on Emerging Telecommunications Technologies **31**(1), e3728 (2020)
18. Chowdhury, N.M.K., Boutaba, R.: A survey of network virtualization. Computer Networks **54**(5), 862–876 (apr 2010). https://doi.org/10.1016/j.comnet.2009.10.017
19. Cohen, R., Barabash, K., Rochwerger, B., Schour, L., Crisan, D., Birke, R., Minkenberg, C., Gusat, M., Recio, R., Jain, V.: An intent-based approach for network virtualization. In: 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013). pp. 42–50. IEEE (2013)
20. Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., Weerawarana, S.: Unraveling the web services web: an introduction to soap, wsdl, and uddi. IEEE Internet computing **6**(2), 86–93 (2002)
21. Domingue, J., Roman, D., Stollberg, M.: Web service modeling ontology (wsmo)-an ontology for semantic web services (2005)
22. Doynikova, E., Kotenko, I.: Approach for determination of cyber-attack goals based on the ontology of security metrics. IOP Conference Series: Materials Science and Engineering **450**, 052006 (Nov 2018)
23. Dzone: (2018), `https://dzone.com/articles/design-patterns-for-microservices`
24. ETSI: Gs nfv-man 001. techreport DGS/NFV-MAN00, ETSI (Dec 2014)
25. ETSI, G.: Zsm-005,". Zero Touch Network and Service Man. agement (ZSM)
26. ETSI, I.: Etsi gs nfv-rel 001 v1. 1.1: Network functions virtualisation(nfv); resiliency requirements (2015)
27. Gruber, T.R.: A translation approach to portable ontology specifications. Knowledge Acquisition **5**(2), 199 – 220 (1993). https://doi.org/https://doi.org/10.1006/knac.1993.1008, `http://www.sciencedirect.com/science/article/pii/S1042814383710083`
28. Han, Y., Li, J., Hoang, D., Yoo, J.H., Hong, J.W.K.: An intent-based network virtualization platform for sdn. In: 2016 12th International Conference on Network and Service Management (CNSM). pp. 353–358. IEEE (2016)
29. Hawilo, H., Jammal, M., Shami, A.: Exploring microservices as the architecture of choice for network function virtualization platforms. IEEE Network **33**, 202–210 (2019)
30. Hoyos, L.C., Rothenberg, C.E.: Non: Network function virtualization ontology towards semantic service implementation. In: 2016 8th IEEE Latin-American Conference on Communications (LATINCOM). pp. 1–6 (Nov 2016). https://doi.org/10.1109/LATINCOM.2016.7811570
31. IETF: Service function chaining (sfc) operation, administration and maintenance (oam) framework. draft-ietf-sfc-oam-framework-05. techreport, IETF (Sep 2018), `https://datatracker.ietf.org/doc/draft-ietf-sfc-oam-framework/`
32. Jahromi, N.T., Yangui, S., Larabi, A., Smith, D., Salahuddin, M.A., Glitho, R.H., Brunner, R., Elbiaze, H.: NFV and sdn-based cost-efficient and agile value-added video services provisioning in content delivery networks. In: 14th IEEE Annual Consumer Communications & Networking Conference, CCNC 2017, Las Vegas, NV, USA, January 8-11, 2017. pp. 671–677 (2017)

33. Kim, S.I., Kim, H.S.: Semantic ontology-based nfv service modeling. In: 2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN). pp. 674–678. IEEE (2018)
34. Kiran, M., Pouyoul, E., Mercian, A., Tierney, B., Guok, C., Monga, I.: Enabling intent to configure scientific networks for high performance demands. Future Generation Computer Systems **79**, 205–214 (2018)
35. Küster, U., König-Ries, B., Stern, M., Klein, M.: Diane: an integrated approach to automated service discovery, matchmaking and composition. In: Proceedings of the 16th international conference on World Wide Web. pp. 1033–1042 (2007)
36. Lal, S., Taleb, T., Dutta, A.: Nfv: Security threats and best practices. IEEE Communications Magazine **55**(8), 211–217 (2017)
37. Lin, P., Wu, C., Shih, P.: Optimal placement of network security monitoring functions in nfv-enabled data centers. In: International Symposium on Cloud and Service Computing (SC2). pp. 9–16. IEEE Computer Society, Los Alamitos, CA, USA (nov 2017)
38. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3431–3440 (June 2015). https://doi.org/10.1109/CVPR.2015.7298965
39. Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., et al.: Owl-s: Semantic markup for web services. W3C member submission **22**(4) (2004)
40. Mazrekaj, A., Shabani, I., Sejdiu, B.: Pricing schemes in cloud computing: An overview (2016)
41. Mechtri, M., Ghribi, C., Soualah, O., Zeghlache, D.: Nfv orchestration framework addressing sfc challenges. IEEE Communications Magazine **55**(6), 16–23 (2017)
42. Mijumbi, R., Serrat, J., Gorricho, J.L., Bouten, N., Turck, F.D., Boutaba, R.: Network function virtualization: State-of-the-art and research challenges. IEEE Communications Surveys & Tutorials **18**(1), 236–262 (2016). https://doi.org/10.1109/comst.2015.2477041
43. Nguyenphu, T.: Sol001:vnf descriptor (vnfd) overview. techreport, ETSI (May 2018)
44. Oliver, I., Panda, S., Wang, K., Kalliola, A.: Modelling nfv concepts with ontologies. In: 2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN). pp. 1–7 (Feb 2018). https://doi.org/10.1109/ICIN.2018.8401590
45. Ordanini, A., Pasini, P.: Service co-production and value co-creation: The case for a service-oriented architecture (soa). European Management Journal **26**(5), 289–297 (2008)
46. Pace, N.M., Zakaras, L.: Moving beyond eyes-on review: The promise of computer-categorized review. In: Where the money goes: Understanding litigant expenditures for producing electronic discovery, chap. 5, pp. 59–69. Rand (2012)
47. Paganelli, F., Paradiso, F., Gherardelli, M., Galletti, G.: Network service description model for vnf orchestration leveraging intent-based sdn interfaces. In: 2017 IEEE Conference on Network Softwarization (NetSoft). pp. 1–5. IEEE (2017)
48. Petcu, D., Martino, B., Venticinque, S., Rak, M., Máhr, T., Lopez, G., Brito, F., Cossu, R., Stopar, M., Šperka, S., Stankovski, V.: Experiences in building a mOSAIC of clouds. Journal of Cloud Computing: Advances, Systems and Applications **2**(1), 12 (2013). https://doi.org/10.1186/2192-113x-2-12

49. Powers, D.M.W.: Evaluation: From precision, recall and f-measure to roc., informedness, markedness and correlation. Journal of Machine Learning Technologies **2**(1), 37–63 (2011)
50. Sahoo, J., Salahuddin, M.A., Glitho, R.H., Elbiaze, H., Ajib, W.: A survey on replica server placement algorithms for content delivery networks. IEEE Communications Surveys & Tutorials **19**, 1002–1026 (2016)
51. Salvadori, I., Oliveira, B., Huf, A., Inacio, E., Siqueira, F.: An ontology alignment framework for data-driven microservices. In: Proceedings of the 19th International Conference on Information Integration and Web-Based Applications & Services (2017)
52. Schaffrath, G., Mathy, L., Werle, C., Papadimitriou, P., Feldmann, A., Bless, R., Greenhalgh, A., Wundsam, A., Kind, M., Maennel, O.: Network virtualization architecture. In: Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures - VISA '09. ACM Press (2009)
53. Seog-Chan Oh, Kil, H., Dongwon Lee, Kumara, S.R.T.: Algorithms for web services discovery and composition based on syntactic and semantic service descriptions. In: The 8th IEEE International Conference on E-Commerce Technology and The 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06). pp. 66–66 (2006). https://doi.org/10.1109/CEC-EEE.2006.12
54. Shapiro, S.C., Rapaport, W.J.: Sneps considered as a fully intensional propositional semantic network. In: The knowledge frontier, pp. 262–315. Springer (1987)
55. Sherry, J., Hasan, S., Scott, C., Krishnamurthy, A., Ratnasamy, S., Sekar, V.: Making middleboxes someone else's problem: network processing as a cloud service. ACM SIGCOMM Computer Communication Review **42**(4), 13–24 (2012)
56. Soares, J., Dias, M., Carapinha, J., Parreira, B., Sargento, S.: Cloud4nfv: A platform for virtual network functions. In: 2014 IEEE 3Rd international conference on cloud networking (cloudnet). pp. 288–293. IEEE (2014)
57. Song, R., Luo, Z., Wen, J., Yu, Y., Hon, H.W.: Identifying ambiguous queries in web search. In: Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007. pp. 1169–1170 (2007)
58. Souag, A., Salinesi, C., Mazo, R., Comyn-Wattiau, I.: A Security Ontology for Security Requirements Elicitation, pp. 157–177. Springer International Publishing (2015)
59. Sowa, J.F.: Semantic networks (1987)
60. Specification, C.: Tosca simple profile for networkfunctions virtualization (nfv) version 1.0. techreport, OASIS (May 2017), `http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/csd04/tosca-nfv-v1.0-csd04.pdf`
61. Tang, C., Xu, Z., Dwarkadas, S.: Peer-to-peer information retrieval using self-organizing semantic overlay networks. In: Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications - SIGCOMM '03. ACM Press (2003). https://doi.org/10.1145/863955.863976
62. Tao, X., Han, Y., Xu, X., Zhang, P., Leung, V.C.M.: Recent advances and future challenges for mobile network virtualization. Science China Information Sciences **60**(4), 040301 (Mar 2017)
63. Thomas, E.: Soa principles of service design. Boston: Prentice Hall **37**, 71–75 (2007)
64. Trabelsi, S., Gomez, L., Roudier, Y.: Context-Aware Security Policy for the Service Discovery, vol. 1, pp. 477–482 (2007)
65. van der Ham, J., Stéger, J., Laki, S., Kryftis, Y., Maglaris, V., de Laat, C.: The novi information models. Future Generation Computer Systems **42**, 64 –

73 (2015). https://doi.org/https://doi.org/10.1016/j.future.2013.12.017, http://www.sciencedirect.com/science/article/pii/S0167739X13002811
66. Voigt, S., Howard, C., Philp, D., Penny, C.: Representing and reasoning about logical network topologies. In: Croitoru, M., Marquis, P., Rudolph, S., Stapleton, G. (eds.) Graph Structures for Knowledge Representation and Reasoning. pp. 73–83. Springer International Publishing, Cham (2018)
67. Walsh, A.E.: Uddi, Soap, and WSDL: the web services specification reference book. Prentice Hall Professional Technical Reference (2002)
68. Wang, M., Jayaraman, P.P., Ranjan, R., Mitra, K., Zhang, M., Li, E., Khan, S., Pathan, M., Georgeakopoulos, D.: An Overview of Cloud Based Content Delivery Networks: Research Dimensions and State-of-the-Art, pp. 131–158. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
69. Willner, A., Loughnane, R., Magedanz, T.: Fiddle: Federated infrastructure discovery and description language. In: 2015 IEEE International Conference on Cloud Engineering. pp. 465–471 (2015). https://doi.org/10.1109/IC2E.2015.77
70. Xia, Y., Jiang, S., Zhou, T., Hares, S., Zhang, Y.: Nemo (network modeling) language. Working Draft, IETF Secretariat, Internet-Draft draft-xiasdnrg-nemo-language-03 (2015)
71. Xilouris, G., Trouva, E., Lobillo, F., Soares, J.M., Carapinha, J., McGrath, M.J., Gardikis, G., Paglierani, P., Pallis, E., Zuccaro, L., et al.: T-nova: A marketplace for virtualized network functions. In: 2014 European Conference on Networks and Communications (EuCNC). pp. 1–5. IEEE (2014)
72. Yangui, S., Glitho, R.H., Wette, C.: Approaches to end-user applications portability in the cloud: A survey. IEEE Communications Magazine **54**(7), 138–145 (2016)