Future Generation Computer Systems 142 (2023) 101–116

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Intent-based network slicing for SDN vertical services with assurance: Context, design and preliminary experiments

B. Martini [a,b], M. Gharbaoui [c,*], P. Castoldi [c]

[a] CNIT, Italy
[b] Universitas Mercatorum, Rome, Italy
[c] Scuola Superiore Sant'Anna, Pisa, Italy

ARTICLE INFO

Article history:
Received 20 May 2022
Received in revised form 16 December 2022
Accepted 22 December 2022
Available online 30 December 2022

Keywords:
SDN
NFV
Intent-based networking
Network slicing
MANagement and Orchestration
ETSI NFV MANO

ABSTRACT

Network slicing is announced to be one of the key features for 5G infrastructures enabling network operators to provide network services with the flexibility and dynamicity necessary for the vertical services, while relying on Network Function Virtualization (NFV) and Software-defined Networking (SDN). On the other hand, vertical industries are attracted by flexibility and customization offered by operators through network slicing, especially if slices come with in-built SDN capabilities to programmatically connect their application components and if they are relieved of dealing with detailed technicalities of the underlying (virtual) infrastructure.

In this paper, we present an Intent-based deployment of a NFV orchestration stack that allows for the setup of Qos-aware and SDN-enabled network slices toward effective service chaining in the vertical domain. The main aim of the work is to simplify and automate the deployment of tenant-managed SDN-enabled network slices through a declarative approach while abstracting the underlying implementation details and unburdening verticals to deal with technology-specific low-level networking directives. In our approach, the intent-based framework we propose is based on an ETSI NFV MANO platform and is assessed through a set of experimental results demonstrating its feasibility and effectiveness.

1. Introduction

The digital transformation and novel vertical applications and services are leading to a rapid growth in the volume and requirements of traffic forwarded by underlying networks and data centers [1]. Enabled by Network Function Virtualization (NFV) and Software-Defined Networking (SDN), network slicing is the telco operators' best answer on how to build and manage a customized network, that meets such emerging requirements from a wide range of vertical industry applications, e.g., industrial automation, telemedicine, self-driving cars [2,3].

Through network slicing, vertical industries (herein after verticals) can benefit from virtualized infrastructure assets to flexibly run their application platforms with resource capacity (e.g., amount of virtual CPU and disk capabilities, virtual link bandwidth) and network functions (e.g., virtual EPC, security functions) tailored to their business or service operation needs (e.g., user demands to serve, preventing specific security threats). On top of that, verticals are attracted by SDN programmability into the deployed slices to adaptively establish and manage their service graphs based on dynamic context information (e.g., user location, service availability, service response time) and toward maximizing user satisfaction and service continuity [4]. However, verticals are not inclined to handle virtual infrastructure technicalities and to specify low-level network configuration parameters, such as virtual machine parameters, network configurations, topology and protocols (e.g., VLAN tags, connection points), needed to realize their business goals and address their programmability requisites through network slices established in the underlying virtual infrastructure [5].

An effective approach to address this challenge is to leverage Intent-Based Networking (IBN), a work in progress presented in the Internet Engineering Task Force (IETF) Internet-Draft [6]. IBN was conceived as a novel approach in network management to enable a loosely-coupled interworking between network and application layers. Applications (i.e., customer applications or service management applications) are able to express desired operational goals using more congenial terms for them, i.e., descriptive high-level specifications known as intents [7]. On the other side, network layers (i.e., network operators) are left the flexibility of meeting those goals based on their own optimization targets (e.g., consolidating the usage of resources).

Originally targeted to increase automation in network management and simplify the process of network configurations [8],

* Corresponding author.
E-mail address: molka.gharbaoui@santannapisa.it (M. Gharbaoui).

https://doi.org/10.1016/j.future.2022.12.033
0167-739X/

recently IBN enlarged the scope to cover different levels of intents (i.e., service intent, operational intent) to address multi-layer business relationship among multiple involved stakeholders in the emerging NFV ecosystems (i.e., application providers, network operators, service providers, application developers) [9]. As for service-level intent specifications, the SDN capabilities at the slice control plane level are pretty attractive for verticals to dynamically steer the traffic among Virtual Network Functions (VNFs) and flexibly realize their service operational and business goals (e.g., context-aware service workflows) [10].

However, addressing those goals poses a number of challenges in particular in terms of (i) defining rich semantics to express (and possibly change) the intent of verticals for a tenant-managed SDN-enabled network set-up into a slice, (ii) translating intents into a set of networking actions, and (iii) assuring the established intent performance during the service lifecycle.

From a thorough analysis of the state-of-the-art, we can state that, on the one hand, there are several works dealing with the Intent-based networking concept in general, and some works with intents in SDN/NFV environments. On the other hand, just a few works discuss the adoption of intents in network slicing contexts. However, in all cited works, the provisioning of SDN programmability has not been considered. Moreover, in most of the works, the intent concept is only discussed while there are no prototyping efforts of intent-based frameworks. In addition, no concrete implementations of the assurance phase including the verification of the slice intents after their execution are tackled. Regarding the adoption of SDN in tenant-managed domains, there are several works dealing with the management of tenant-side SDN connectivity for the provisioning of services in the application domain, given the need for service and application providers to have elasticity of the network that connects the service nodes. Those works however do not consider the network slicing context.

In this work, we propose an IBN-based framework for NFV network slicing to allow verticals to specify customized high-level specifications and their connectivity needs through an intent while getting benefits from SDN programmability within their service domains; and network operators to effectively establish network slices with SDN capabilities that are also assured, in line with the expressed intent during the slice lifecycle. More specifically, the proposed intent-based SDN network slicing addresses the following challenges: (i) the expression of high-level SDN network requirements for vertical network slices through a declarative model using intents; (ii) the automated deployment of tenant-managed SDN-enabled network slices using an ETSI NFV MANO platform; (iii) the provision of a reliable network service supporting the specified slice in a virtualized telco cloud environment, and (iv) the continuous check of the current status of the slices to guarantee their correct behavior with respect to the objectives, during the whole lifecycle of the intent. Our framework is fully inline with the IBN functional framework defined at the Internet Research Task Force (IRTF) Network Management Research Group (NMRG) [6] and all its building blocks are implemented and assessed.

This paper extends the preliminary work presented in [11] with (*i*) a thorough overview on Intent-based networking and network slices orchestration and management in NFV environments, (*ii*), an expansion of the description of the proposed Intent-based framework and the vertical service scenarios that might be attracted by it, (*iii*) and a more comprehensive evaluation of the intent layer considering two different virtualized testbeds and a richer set of metrics.

The remainder of the paper is organized as follows. Section 2 gives a background on the intent-based networking, orchestration of network services and management of network slices in NFV environments. Section 3 elaborates on service scenarios that are relevant for QoS-aware and SDN-based network slices in the vertical domain. Section 4 introduces the intent model adopted for the expression of SDN network slices. Section 5 presents the IBN approach in NFV and the intent-layer design for NFV environments. Section 6 describes the intent layer implementation details of the proposed framework and presents a preliminary evaluation of it. Section 7 provides a review of recent research studies on IBN and network slicing. Section 8 provides the challenges and future directions, to motivate more innovative research in this field. Finally, Section 9 draws the conclusions.

2. Background

2.1. Intent-based networking

The IBN concept represents a new approach to network management conceived by the IETF as an advance to network management where users at an Application Layer (e.g., vertical's Operations Support System/Business Support System (OSS/BSS)) or a service layer (e.g., network operator OSS/BSS) can express their business, service or operational goals through high-level prescriptive directives (i.e., intents) thus unburdening applications to deal with technology-specific low-level networking directives needed to achieve those goals. Examples of intents are: "*set a connection as a high availability network service*", "*always maintain high quality of service and high bandwidth for gold level users*". On the other hand, the network operators are left the flexibility of addressing the expressed goals based on their own optimization decisions in the light of a loosely-coupled interworking with applications [6].

The IBN approach is possible through the mediation of an Intent Orchestration Layer (or *Intent Layer*) that allows to (i) automate the implementation of network configurations required to realize the goals expressed by applications, and (ii) regulate the lifecycle of the established configurations in line with expressed goals. Overall it manages and regulates the lifecycle of intent demands from applications through *fulfillment* and *assurance* operations in a closed loop workflow. More specifically:

- *fulfillment* operations deal with processing intents from their origination by a user to their realization in the network, including translation and any required orchestration of coordinated configuration operations;
- *assurance* operations deal with ensuring that the network actually complies with the desired intent once it has been fulfilled also based on real-time collection, aggregation and assessment of monitoring data.

These two kinds of operations are realized through the interworking of the following main functional blocks: (i) the *Translation* which parses the application's intent and translates it into a set of networking actions, mainly configurations; (ii) the *Management and Decision* which is responsible of identifying the needed actions to ensure that the intent is achieved including derivation of the algorithm to be used, learning on how to optimize outcomes over time, and coordination of configurations and deployment actions, e.g., rendering of high-level abstractions into lower-layer parameters, (iii) the *Analyses/Verification* which continuously verifies the status of the intent, and if necessary triggers corrective actions leveraging on the *Management and Decision* block, and finally (iv) the *Intent Repository* which is a database that interacts with the intent Management and Translation modules to provide mapping between the "intent" and its "configuration".

IBN allows application layers to interact with the Intent Layer avoiding to learn the technical-specific language of the underlying system. On the other hand, it also allows network providers

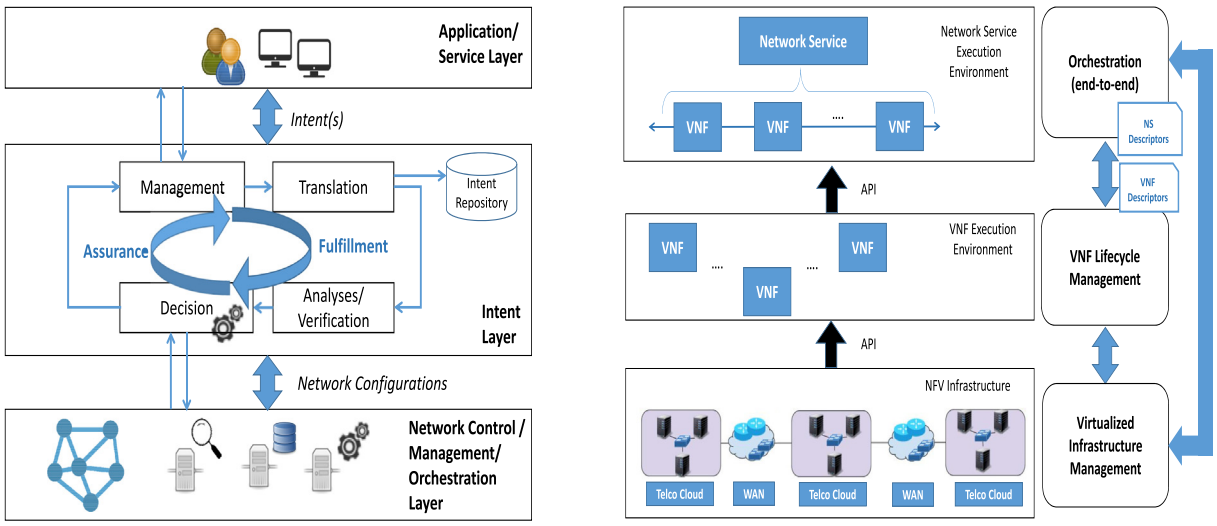

Fig. 1. (a) Intent-based networking architecture and (b) NFV MANagement and Orchestration (NFV MANO) framework.

to (i) improve the network agility and availability, (ii) manage networks holistically at a higher level of abstraction, and (iii) continuously verify that tenant goals are met. Hence, IBN not only contributes to increase network automation and flexibility to upper layers but also to improve the robustness of the network thanks to closed-loop operations and continuous learning so as to improve automation and maintenance and reduce costs [8].

2.2. Network slice orchestration

Network slices are flexible partitions caved out from a shared virtualized infrastructure while leveraging NFV and SDN to (i) deploy network functions (e.g., router, firewall, intrusion detection systems) as software components in virtual machines (i.e., VNFs) and (ii) dynamically and programmatically interconnect them via software network controllers [12] under the ETSI NFV MANagement and Orchestration (NFV MANO) architectural framework [13]. As shown in Fig. 1(b), the framework is based on decoupling the levels of (i) NFV Infrastructure (NFVI), i.e., the pool of physical/virtual computing, storage and network capabilities operated on top of distributed telco clouds; (ii) VNFs, i.e., the pool of individual network functions running in virtual machines deployed in the NFVI; (iii) end-to-end Network Services (NSs), i.e., combination of connected VNFs in a self-consistent network service. Correspondingly, each level has the respective control and management functionalities: (i) Virtual Infrastructure Manager (VIM) for compute and storage resources in telco clouds, and WAN Infrastructure Manager (WIM) for networks to connect multiple resources across different clouds in NFVI; (ii) VNF Manager (VNFM) responsible for VNF lifecycle management under the control of the orchestrator, i.e., instantiation, configuration, scaling, termination operations at VNFs: (iii) NFV Orchestrator (NFVO) that dynamically composes and orchestrates VNFs and virtual links to automatically establish end-to-end (E2E) NSs underpinning the network slices. The NFVO also takes care that VNFs have adequate compute, storage and network resources throughout the entire lifecycle and, if needed, trigger proper scaling operations at VNFs to assure service integrity according to specified requirements.

In order to perform E2E NS and VNF orchestration, the NFVO does not just need the VNFs software images but also data models (i.e., descriptors) specifying the deployment and the operational behavior of VNFs and NSs. For this purpose, the ETSI NFV MANO specifies, among others, the VNF Descriptor (VNFD) and the Network Service Descriptor (NSD) [14,15]. The NSD is the top-level deployment template that defines the pool of VNFs composing the E2E NS, their connection points and how they are connected to form the network service. The VNFD is a deployment template that describes the attributes, requirements, and capabilities of a single VNF and related relevant components, i.e., external connection points, internal components and connectivity, performance and resource requirements and monitoring parameters.

As service delivery (data) models, NSD and VNFD involve a certain degree of abstraction to cope with heterogeneity of networking devices and systems. However, they differ from intents in the prescriptive nature they still have while specifying, e.g., service technical parameters, corrective actions and life-cycles. Instead, intents are declarative in nature (i.e., specify operational or business goals of the user or network operator) and do not specify how those goals should be achieved. In intent-based systems, the algorithm or the rule to apply are not pre-defined but they can be automatically derived from the intent or even learned, e.g., through machine learning techniques.

3. SDN network slices in vertical service scenarios

Network slices are not merely established for the provisioning of virtual connectivity and computing capabilities but also, and especially, to run vertical services to the extent of the assigned (virtual) resource chunks. Hence, network slices can be also seen under the perspective of a virtualized service infrastructure assets offered by network operators to verticals [16]. Regarding slice service capabilities, verticals are highly attracted by leveraging SDN within slices to programmatically establish connectivity, e.g., service chains, among software service and application components (e.g., content caches, edge servers) in a flexible and scalable manner. In such a case, the service chains are established on top of a virtual network of nodes within the slice and, hence, within the tenant domain [17]. This scope of operation and control of slices from tenants is referred to as tenant-managed slice, with tenants having the full control of the resources and functions allocated into the slice [18], e.g., controlling the data plane forwarding rules into SDN switches composing the virtual tenant network established into the slice.

As shown in Fig. 2, three different service scenarios can be considered where verticals are interested in SDN-based network slices featuring different structure and deployment set-ups.

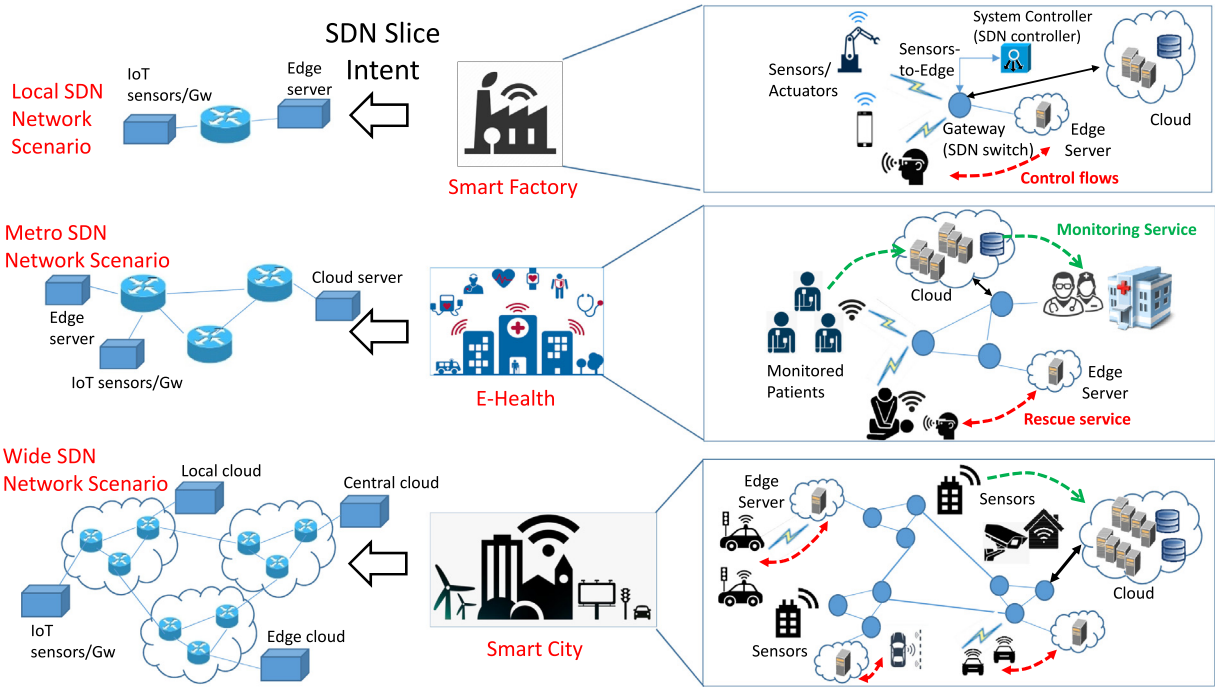

Fig. 2. Service scenarios for SDN network slices and intents.

3.1. Smart factory (local scenario)

The Smart Factory scenario refers to the fourth stage of industrialization where emerging technologies such as Internet of Things (IoT) and edge computing represent the main enabling factors. In such a scenario, multiple nodes (i.e., sensors, actuators) are deployed that communicate with a server located beyond the gateway and local network (i.e., edge server) to send data related to the state and process information of the smart machines in the factory and then receive the sequence of operations to be enforced on the products [19]. In this context, edge computing provides a very elastic solution since it brings the storage space and computing ability close to IoT nodes, thus reducing the response time [20]. Such communication between IoT nodes and the edge server is characterized by a low throughput. In addition, reliability is a strict requirement due to criticality of factory systems and automation. In this scenario, an SDN-based network slice can suitably support the smart factory services with a simple SDN topology composed of one (redundant) switch (coupled with a controller) that connects (i) Virtual Functions (VFs) representing the IoT sensors and/or gateway and the edge server. An external cloud could be also used (e.g., enterprise cloud) to offer centralized services such as data analytics, off-line process optimizations, informative services.

3.2. E-Health (metro scenario)

The E-Health scenario refers to the delivery of smart-healthcare services that provide faster and more efficient solutions to patients thanks to IoT and cloud-based technologies [21]. Basically, an E-Health system is generally composed of (i) IoT wearable medical sensors, (ii) few services deployed at the edge, and hence closer to the patients, which can provide support for latency-sensitive operations (such as virtual/augmented reality (VR/AR) in rescue operations), and (iii) centralized back-end services placed in the cloud, which allow for processing the collected data and making it accessible for the medical staff to analyze it during normal care situations. This system allows for instance to set-up a service for the continuous monitoring of life functions of patients through wearable devices. Normally, those data are stored in the cloud for the medical staff from hospital to analyze them during visits. In case event failures are detected on the vital signs (e.g., heart attack, seizure disorder), a rescue service can be timely requested to act earlier on emergency cases. Indeed, an alarm is triggered to activate an edge server in close area of the patient to support paramedics services with low-latency and high-throughput communication as required by AR/VR equipments. In such a situation, the edge server can collect data for a rapid diagnosis locally or for a more in-depth analysis performed by a qualified medical staff from the hospital. Hence, reliability of data transfer is also a strict requirement. This scenario can be deployed through an SDN-based network slice composed of a set of switches covering a metro area and a number of VFs connected to them representing both the monitoring back-end server and the emergency services at the edge close to the patients [22].

3.3. Smart city (wide scenario)

The Smart City use case refers to a wide range scenario where thousands of devices for monitoring applications and service facilitation to the citizens are interconnected. In fact, future smart cities will be characterized by a dense deployment of sensor nodes to advance systems and services, including smart buildings, autonomous vehicles (e.g., V2V, V2X), intelligent transportation (e.g., traffic signal control systems, container management systems), just to name a few. Those sensors and nodes need to be easily interconnected with both edge facilities for time sensitive applications and either local or central clouds (i.e., storage, computation servers) to process and analyze the collected data at local and global scale, respectively [23]. The amount of such data differs from one scenario to the other. Smart buildings for example are characterized by sporadic and intermittent transmissions of very small packets, typically on the order of a few hundred bytes, while augmented reality applications in the vehicle-to-vehicle use case require higher throughput values. Other examples such as smart traffic lighting systems or autonomous driving

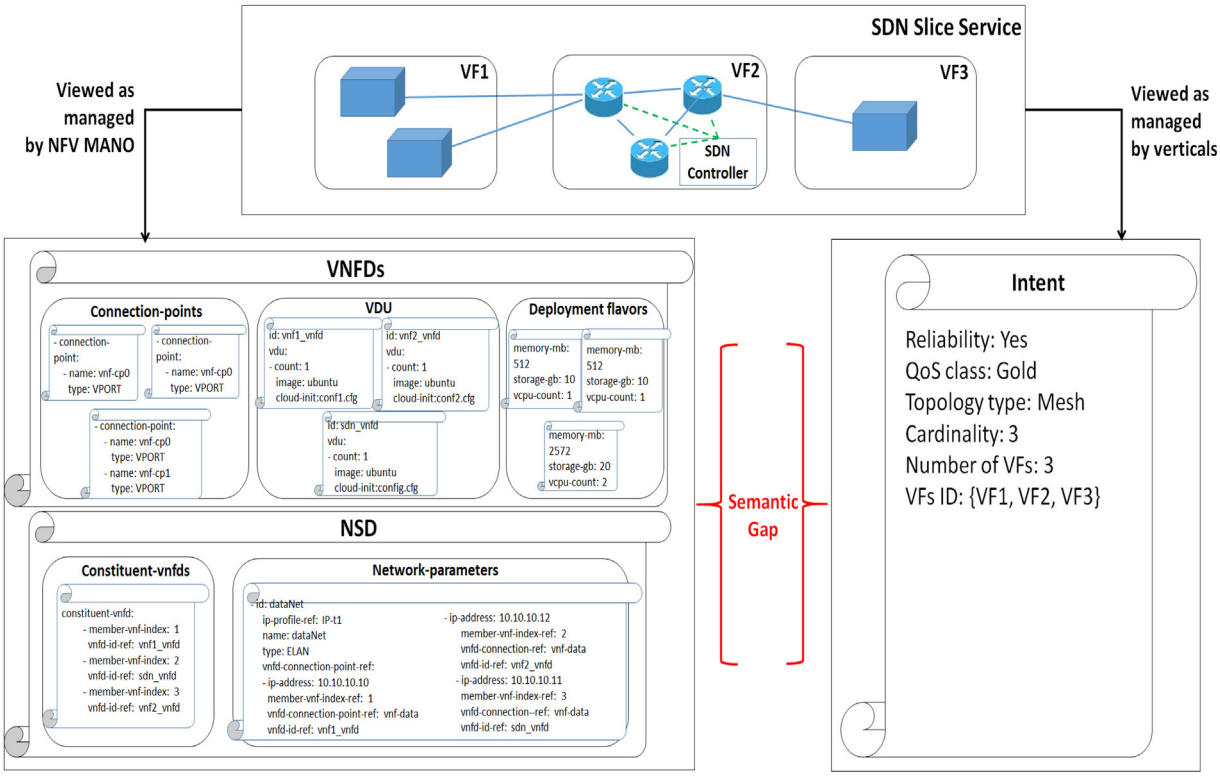

Fig. 3. Network slice descriptors.

might require high reliability features and ultra-low latency. In this scenario, an SDN-based network slice can be composed of a high number of switches, which facilitates the management of the smart infrastructures and allows for the customization of data flows of the smart city services that can pass through several VFs. The SDN switches cover a wide area and connect a relatively high number of VFs representing the sensor nodes and the several cloud/edge servers. In this scenario, the case of multiple yet interconnected slices can be considered due to the higher complexity and wider area extension that needs to be covered.

4. SDN network slice services as intents

For verticals it is not straightforward to rely on NS lifecycle management functionalities (and related descriptors and APIs) to specify their demands on slice capabilities due to the conceptual distance between parameters of slice deployables (i.e., slice service operational settings) and of ETSI NFV descriptors (i.e., NSD, VNFD) [24]. Indeed, the MANO framework assumes a tight coupling of the slice-related operations to NS APIs which require verticals to handle virtual infrastructure technicalities in NSDs and VNFDs to specify slice service demands, such as virtual machine parameters, network configurations, topology and protocols (e.g., VLAN tags, connection points).

The semantic gap between the verticals intentions on service settings and the slice and network configuration parameters handled by NFV MANO orchestrator is shown in Fig. 3. The case of SDN slice service is considered for Metro Scenario (i.e., E-health service) as shown on top. In the left-hand side of Fig. 3 we report some extracts from the VNFDs and NSD used to deploy the given network slice. Each block in the descriptors includes the details necessary to deploy the requested VFs as well as their interconnections within the slice. More specifically, in the VNFD of each VF, the *VDU* block reports the characteristics of the VM that will host the virtual function (e.g., cloud image, configuration file). The *Deployment flavors* block shows the computing capabilities of each VM (i.e., vCPU, RAM and storage), while the *Connection-points* block details the characteristics of the network interfaces for each VM. On the other hand, the NSD deals with the interconnections between the VFs by describing their order in the forwarding graph, the IP addresses assigned to their interfaces, etc. In the right-hand side of Fig. 3 we report the set of service-related operational parameters for the same network slice service as intended by verticals, i.e., intent. The semantic gap appears evident since verticals would prefer to mainly focus on (i) overall required capacity of virtual resources and the set of component network functions, (ii) connectivity slice requirements in terms of e.g., kind of network control and programmability, network topology in the vertical domain [25], and (iii) QoS terms such as level of reliability, bandwidth demand, latency constraints [26].

To fill this semantic gap, a prescriptive model of a slice service should be conceived and an IBN approach adopted in NFV environments to support slice intents. According to the scenarios shown in Section 3, in this paper, we consider the case of SDN-enabled slice service intents which enable verticals to flexibly establish vertical service connectivity in their slices. In such a context, the IBN approach for network slicing allows to (i) capture the demands (intents) of slice services from verticals including SDN capabilities; (ii) transform those demands into network slice requests in terms of fully-specified NSD and VNFD coupled with measurable performance indicators for SDN connectivity; and (iii) continuously verify the matching between the actual and the expected slice service state and, in case, adjust the virtual infrastructure and NS settings accordingly to reflect the vertical business goals and slice service needs [27].

5. Intent layer for SDN-enabled network slicing

In order to address IBN in NFV environments we present in this work an Intent Layer designed to run on top of NFV MANO

Table 1
Slice intents types.

Service scenario	Reliability	QoS class	Topology type	Cardinality	Number of VFs
Local scenario (Smart Factory)	Yes	Bronze	Single	Small [1]	[2, 4]
Metro scenario (E-Health)	Yes	Gold	Linear-Mesh	Medium [3, 10]	[2, 4]
Wide scenario (Smart City)	Yes	Silver (Smart building)/Gold (Autonomous driving)	Linear-Mesh	Large [25, 50]	[2, 6]

orchestrators and able to address vertical operational goals for network slices. In this section, we describe the Intent specification and API exposed by the Intent Layer 5.1 and the Intent Layer design and implementation 5.2.

5.1. Intent specification and API

The SDN slice service requirements for different scenarios reported in Section 3 can be reflected into an intent specification that captures relevant parameters for verticals to express their operational goals for SDN-enabled slices. In particular, the intent specification should allow verticals to express the need for deploying SDN capabilities to connect VFs (i.e., a network slice composed of an SDN controller, a set of OpenFlow (OF) switches connected in a specific topology and a number of VFs connected to the switches) as well as relevant requirements in terms of QoS and reliability. Both kinds of features differ from one service scenario to the other (different SDN topology, throughput demand, latency constraints, reliability features) as reported in Table 1.

For the purpose of intent specification, we rely on the Network Intent LanguagE (Nile) as an intent definition language [28] for verticals to express their operational goals to the Intent Layer using a language that resembles the natural one and is easy to understand. As shown in [29], Nile is used in this work as an intermediate intent representation that is congenial for both upper layers (e.g., Natural Language Processing (NLP) systems) and for the intent translation purposes toward lower networking layers. Indeed, Nile as a structured intent language works well as a target for the learning algorithms, e.g., to map named entities extracted from unstructured text in natural language through a chatbot. On the other hand, it allows a straightforward translation to different target descriptors or policies to carry out needed configurations in the underlying network. Hence, using Nile in this work opens the path to smoothly use the proposed Intent Layer with NLP-based solutions to understand vertical requirements expressed in natural language.

Consider as an example the following intent: *"Set-up an SDN-based slice where reliable traffic for Gold customers flows from the gateway to the server with at least 5mbps of bandwidth through an SDN mesh topology composed of 3 switches and 2 VFs and where VF_1 is connected to Switch 1 and VF_2 is connected to Switch 3"*. This intent can be expressed using the Nile intent shown in Listing. 1.

```
Define intent sdnSliceIntent:
   from endpoint('gateway')
   to endpoint('server')
   for group('Gold')
   for service('reliable')
   set topology('mesh', 'medium')
   set size('3')
   add node('s1'), node('s2'), node('s3')
   add middlebox('VF_1', 's1'),
   middlebox('VF_2', 's3'),
   with throughput('more or equal', '5mbps')
```

Listing 1: Nile intent example

The vertical first needs to specify the Id of the intent. The Id refers to a unique identifier used to associate the intent to one vertical and then to link it to the set of operations performed during its lifecycle (e.g., execute, check status, delete). The intent contains also the following parameters:

- *Endpoint*: refers to the Service Access Points (SAPs) required to connect to the slice;
- *Group*: it refers to the set of clients that can be grouped based on the established SLA in terms of QoS, and in particular, in our case regarding the amount of transmitted throughput between a given couple of VFs within the network slice. More specifically, a *Gold* group refers to a QoS level that guarantees 100% of the amount of throughput required, *Silver* to a guaranteed level of 75% of required throughput, while *Bronze* to a guaranteed level up to 50% of the required throughput;
- *Service*: if set to *"Reliable"*, it forces the Intent Engine to continuously check if the network slice is correctly deployed (e.g., VFs composing the slice are up and running, connectivity among the VFs is guaranteed), and trigger countermeasures if not;
- *Topology*: refers to the type and cardinality of the SDN-based topology. This topology is composed of a number of OF switches and allows for interconnecting the VFs to each others. The type of topologies allowed can be chosen among *single* (i.e., only one switch), *linear* (i.e., all switches are linked one after the other in a sequential chain) or *mesh* (i.e., each switch is interconnected with one another). The cardinality refers to the range of network nodes that a topology can contain (i.e, *small*, *medium* or *large*);
- *Size*: expresses the extent of the SDN-based topology in terms of number of OF switches composing it. This number is directly related to the cardinality of the topology. More specifically, a *Small* topology can contain only 1 switch, a *Medium* topology can contain between 5 and 10 switches while the *Large* one can contain between 25 and 50 switches;
- *Node*: indicates the identifier of each OF switch composing the SDN topology. Those ids are necessary for the Middlebox parameter to indicate which Middlebox is connected to which switch;
- *Middleboxes*: in this work we consider that a network slice is necessarily composed of one SDN-based VF and at least 2 VFs attached to it representing the virtual appliances that differ according to the considered use case. The parameter *Middleboxes* in the intent expresses the id of the VFs and their number, which is between 2 and 4 in the local and metro SDN scenarios and between 2 and 6 in the wide SDN scenario.
- *Throughput*: refers to the actual amount of data that a vertical wants to transmit between any couple of VFs composing its network slice. It is expressed in Mbps.

Regarding the syntax of the intent, we leverage on the Nile grammar presented in [28] to guarantee that the intent contains

the information necessary for its correct deployment and prompt verification. More specifically, we used the set of supported operations specified in [29] including the *endpoints* targeted by the intent, the *group* and *traffic* functions to abstract an aggregation of customers asking for the same service standards (e.g., reliability, QoS class) and the set of *middleboxes* identified by a specific id that must be connected to the SDN topology. However, to meet the specificities of the intent presented in this work and to include also topological specifications, we also extended the Nile language by introducing two novel functions, which are *topology* and *size* and by adding a novel attribute to the existing *middlebox* function referring to the id of the switch referred to as node to which the VF is connected. For this purpose, we extended the grammar of Nile, in EBNF notation [30], as shown in Listing 2.

```
<intent>      ::= 'define intent' intent_name ':'
                  <commands>
<middleboxes> ::= 'add' <middlebox> { (',' | ', ')
                  <middlebox> }
<middlebox>   ::= 'middlebox('middlebox_id' ','
                  'node id')'
<nodes>       ::= 'add' <node> { (',' | ', ')
                  <node> }
<node>        ::= 'node('node_id')'
<endpoint>    ::= 'endpoint('endpoint_id')'
<topology>    ::= 'topology('type'','cardinality')'
<type>        ::= single|linear|mesh
<cardinality> ::= small|medium|large
<size>        ::= 'size('topology size')'
<group>       ::= 'group('group_id')'
<service>     ::= 'service('service_id')'
<traffic>     ::= 'traffic('traffic_id')' |
                  'flow(' [<five_tuple>]+ ')'
...
```

Listing 2: Nile grammar extension

Those terms are enough for the Intent layer to allocate the network resources and deploy the suitable network functions for the requested SDN-based slice. They are first enforced during the configuration and execution phase of the intent, and then are continuously checked to be validated and, in case, corrected, thus constantly assuring the quality of experience expected by the verticals.

5.2. Intent layer building blocks and implementation

As shown in Fig. 4, the architecture of the intent-based framework is composed of three main elements: (i) the vertical, which sends intent requests to the Intent Layer to deploy SDN slices with specific requirements. The design of the framework also offers to the vertical the possibility to receive some feedback regarding the deployment of the slice, which we leave to future implementations; (ii) the Intent Layer, which represents the core of the framework; and (iii) NFV MANO, on top of which the Intent Layer is supposed to run. In this implementation, we considered Open Source MANO (OSM) [31] as a relevant representative of an NFV MANO orchestrator.

In the following, we describe the Intent layer main components giving some details on their implementation considering the case of SDN-based slice intents [32]:

- the *Intent Engine* component plays the role of a front-end with the vertical, in charge of receiving the intent and parsing it through a specific API to extract the requirements.
  In our prototype, the Intent Engine is composed of two elements: a Graphical User Interface (GUI), which allows verticals to express their goals in a user friendly manner and an Intent Processor, which extracts the values specified by the vertical and provides them to the Translation & Execution component. Upon the reception of an Intent request, the Intent Engine checks the eligibility of each intent by verifying the feasibility of its deployment according to the current status of the deployment infrastructure. More specifically, it makes sure that there are enough resources available to answer the request (i.e., an intent can be directly blocked if for example no enough computational and/or network resources are available to deploy the requested VNFs), thus guaranteeing that the deployment of the intent with its specific requirements (e.g., number of VNFs, amount of throughput) will not impact the existing intents in the system [25].
- the *Translation & Execution* component analyzes the intent and translates the vertical goals into a set of configuration constraints. Such parsing and translation effort strictly depends on the way the intent is expressed (e.g., natural language, template-based approach). The component is also in charge of verifying the effectiveness of the intent, and in case of conformity, triggers the automated deployment of the slice. This component is inline with the execution verification component described in [8].
  In our prototype, the Middlebox parameter is translated into N applicative VNFDs where each VNFD corresponds to the specified Middlebox id, while the throughput value is updated within the cloud-init file of each descriptor. For the sake of simplicity, we consider that all the VFs have the same type. The topology type and size are mainly used to associate the intent to a specific NSD. The Intent ID is used as a parameter for the creation of the NS.
- the *Slices Management Engine* component is responsible for interacting with the NFV MANO layer for the purpose of the automated deployment of the slices supporting the intent. Such interaction is performed through the REST API of NFV MANO which allows for either selecting the necessitated configuration files (e.g., blueprints) from a specific catalogue or uploading them after having created them in a custom way. Those files mainly aim at determining the deployment and operational behavior of each VF composing the slice. They are then sent to the NFV MANO layer which provides a set of functionalities for the orchestration of the VFs and E2E NSs underpinning the network slices. The Slices Management Engine component can be part of an extended set of functionalities at network provider's OSS/BSS for slice-aware service management operations [33].
  In our prototype, the *Slices Deployment Engine* sends a set of RESTful commands to the northbound interface of *OSM*. Once the descriptors are onboarded and validated, it receives an acknowledgment which allows it to effectively trigger the deployment of the network slice. To do so, OSM interacts with the VIM (i.e., *OpenStack* in our implementation) to start the creation of the VFs and their configuration.
- the *Monitoring & Validation* component continuously verifies the status of the slices through different possible approaches (e.g., polling of specific metrics, comparison to known baselines). If the validity of the slice is not verified, it interacts with the Intent Engine to perform countermeasures regarding the execution of the intent. This component is inline with the validity verification component described in [8].
  In our prototype, the *Monitoring & Validation* component continuously inspects the behavior of the network slice in terms of connectivity and QoS (e.g., effective transmitted throughput), and in case of non compliance of the network slice with the specified requirements (e.g., non respected throughput) receives an alarm from the deployed slice and triggers the Intent Engine to perform corrective actions. In

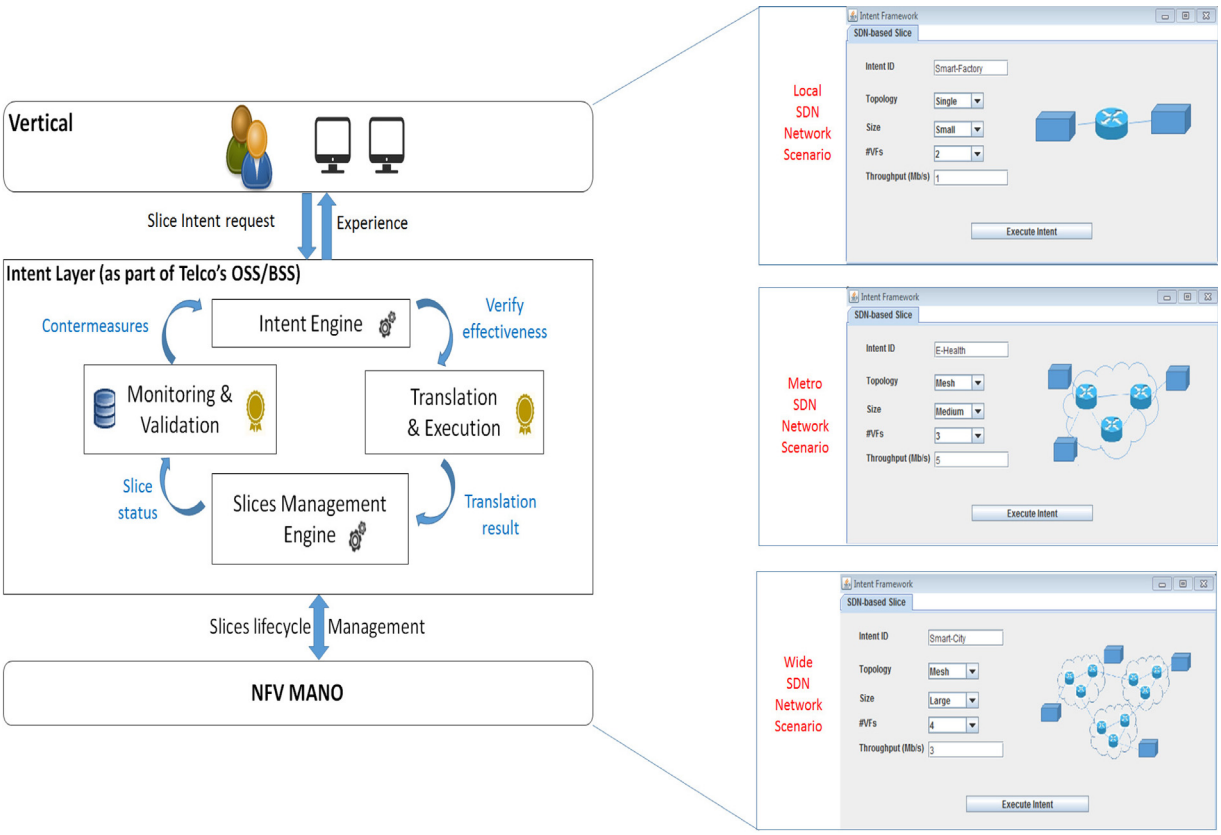

Fig. 4. Intent-based framework architecture and GUI.

such a way, automated intent refinements are possible in case of misconfiguration of the intent or sudden service degradation to restore the performance to the level required by the intent and assure its correct execution without any intervention from the vertical and/or the network operator.

A preliminary implementation of the GUI is shown in the right side of Fig. 4. The verticals only need to specify few operational objectives of the intent they want to deploy (e.g., throughput, topology, number of VFs) and then click on *"Execute Intent"*. The Intent framework parses the provided parameters, maps them to a set of descriptors, which are then sent to OSM to deploy the slice.

In the workflow shown in Fig. 5 we depict the interactions between the different components of our intent framework and we highlight the main operations performed during the deployment of a given network slice. More specifically, the Intent layer receives a request from a vertical through the API for the deployment of a network slice. Since the request is expressed through the simple NILE language, the Intent Engine with the help of the Translator filter the key words and extract the actual slice parameters. The intent is then mapped to a set of descriptors, namely (i) VNFDs, which define the behavior of each type of VF specified in the intent (e.g., VF image, storage, memory, CPU and networking resources), and (ii) one NSD, which mainly defines the logical graph of the network service (i.e., how the VFs are connected in the network provided in the slice). Once the descriptors are correctly associated to the vertical's intent, the Intent Engine interacts with OSM via its REST API to onboard the descriptors and deploy the NS. The OSM then processes the NS and if no errors are found, the slice instance is created. This finalizes the slice deployment process, which does not require any technical expertise or manual intervention from the vertical [6]. Once the network slice is deployed, the Monitoring & Validation component comes into play to periodically check its status and, in case of non conformity with the intent goals, perform remedial actions.

6. Experimental set-up and validation results

In this work, we focus on the design, the implementation and validation of the intent layer. Out of the 3 scenarios described in Section 3, we specifically focus on the metro scenario. The performance evaluation aims at determining the feasibility of our approach and assessing the operation of the whole network slice (and not the performance of the single VFs composing it) in terms of instantiation, connectivity and throughput, thereby validating the intent under different conditions. To this purpose, we leverage on open source tools capable of the deployment of the network slices. More specifically, as shown in the right side of Fig. 6, our testbed consists in the IBN layer we developed, the OSM platform [31], which represents the NFV MANO framework and Openstack [34], which plays the role of the VIM. As depicted on the left side of the figure, OSM MANO and Openstack are separately installed on 2 different bare metal servers hosted on the Virtual Wall (VW) testbed [35], while the Intent Layer's code and the GUI used as a northbound interface to access it run on a third server also hosted by the VW testbed. The characteristics of the testbed components are summarized in Table 2.

Our intent framework connects the OSM north-bound interface to the verticals/applications to facilitate the deployment of the network slices. OSM starts by onboarding the descriptors and then deploys the requested slice with the help of the Openstack VIM. As shown in the bottom of Fig. 6, any slice is composed of one SDN-based VF (which contains the SDN controller and the emulated network topology) and other N VFs that are connected to the SDN-based one (N corresponds to the number of virtual functions specified by the vertical in the intent's GUI).

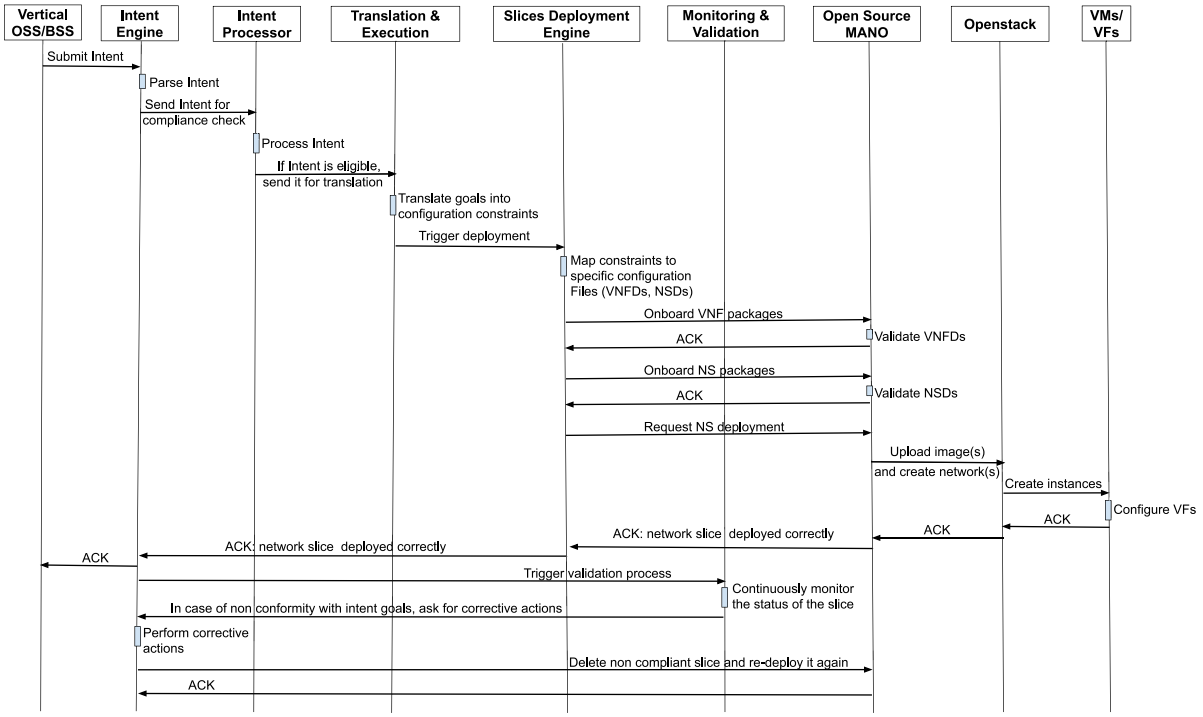

Fig. 5. Intent layer components workflow.

Table 2
Testbed components characteristics.

Component	Characteristics
IBN layer/GUI	Ubuntu 16.04 CPU: 3 core i7-3770 at 3.40 GHz RAM: 8 GB Disk storage: 250 GB
OSM	Ubuntu 18.04 LTS CPU: 4 core E3-1220v3 at 3.1 GHz RAM: 16 GB Disk storage: 500 GB
Openstack	Ubuntu 18.04 LTS CPU: 4 core E3-1220v3 at 3.1 GHz RAM: 16 GB Disk storage: 500 GB

In this section, we only considered one scenario to validate our framework (i.e., the E-health application in the metro scenario) and we fixed some values that we consider to be realistic. More specifically, we varied the number of switches from 5 to 50 and the number of VNFs from 2 to 6. Those numbers are in our opinion compatible with metro scenarios as reported in [36–38]. Regarding the throughput, we fix it between 1Mbps and 10Mbps. This is in our opinion a reasonable compromise mainly because, as reported in the Refs. [37,39,40], the throughput values in metro scenarios vary between 0.1 and 50 Mbps.

In Fig. 6 we consider a metro SDN network scenario composed of 3 VFs and a topology composed of 3 switches. The whole slice creation process is totally automated including the configuration of the VFs composing the slice and their inter-connectivity. Once the slice is deployed, an automated monitoring process is triggered to periodically check its compliance with the vertical goals (e.g., established connectivity, guaranteed throughput) and operate in case of issues.

6.1. Slice intent deployment time

We first consider the overall slice intent deployment time, measured as the time from when a vertical requests an intent to the time when data begins to flow between at least a couple of VFs. We take the example of the metro service scenario where a vertical requests an intent to create an SDN network composed of few switches to which two VFs are attached, and then to guarantee their inter-connectivity and the amount of throughput transmitted among them.

The intent is characterized by a network slice composed of 3 VFs: one SDN-based VF containing an OF *Mesh* topology of a *Medium* size and 2 others VFs connected to it. We fix the QoS level to gold which corresponds to a guaranteed throughput equal to 5 Mb/s. The 3 VFs have the following characteristics in terms of computational resources. The SDN based one requires 2vCPU, 2572MB of RAM and 20GB of storage, while each one of the other 2 VFs requires 1vCPU, 512 MB of RAM and 10 GB of storage. We repeat the deployment of the slice 10 times under the same experiment conditions, then we take the average of our measurements. Results show that traffic starts flowing across the network after around 282.7 s in the virtual wall testbed.

In order to check the impact of the bare metal servers characteristics on the performance, we repeated the same experiment using the same Openstack version but installed on a server from the CloudLab testbed [41] having the following characteristics: 8 64-bit ARMv8 cores at 2.4 GHz, 64 GB of RAM and 500 GB of disk storage. Results show that the overall slice intent deployment time is reduced to an average of 179.47 s which represents a decrease of about 65% with respect to the Virtual Wall testbed. This shows the impact of the physical infrastructure on the overall performance of the framework. This time can be further reduced by tens of seconds by using pre-configured VM images where the required software has been priorly downloaded and installed offline.

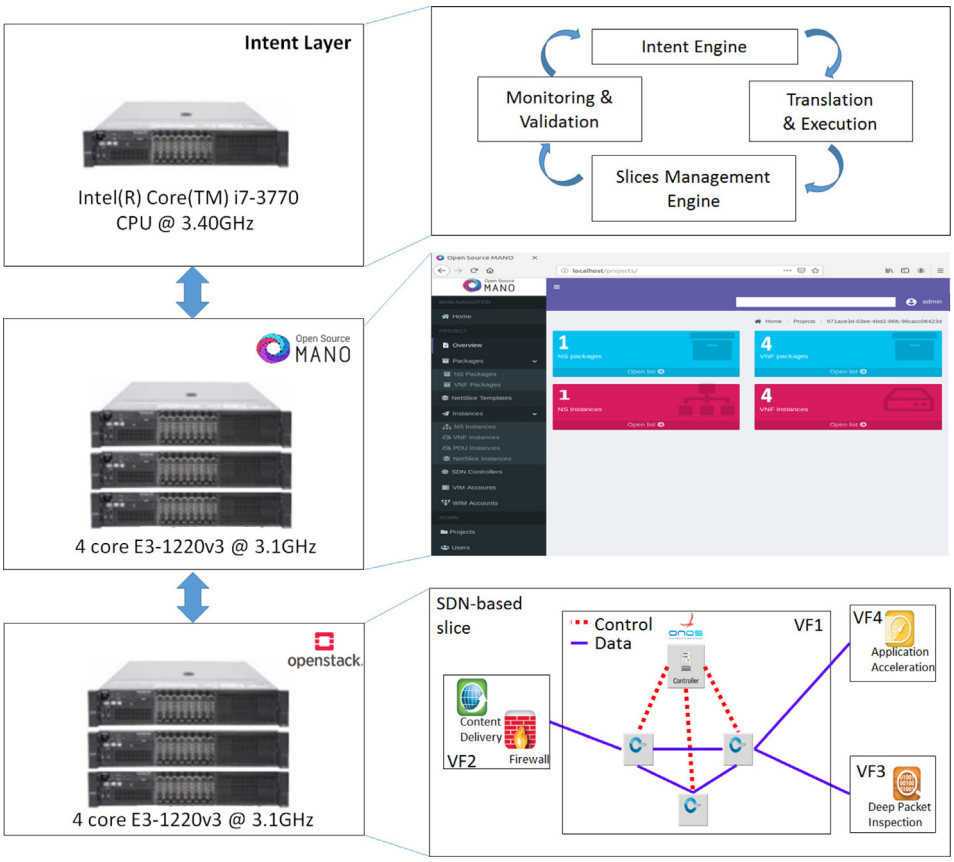

Fig. 6. Experimental Set-up and example of an SDN-based slice components for the metro scenario.

6.2. Components of the slice intent deployment time

In this paragraph, we detail the main components of the slice intent deployment time and assess the average time taken by each element of our intent framework to perform its task. In Fig. 7, we plot the average execution time values in the two testbeds: the Virtual Wall testbed and CloudLab. Results show that the main component of the overall execution time is due to the slice deployment time in Openstack (around 77% of the overall time) while the remaining time is mainly divided between the NS instantiation time in OSM MANO (around 22%) and the descriptors onboarding time (around 1%). The parsing time of the intent is negligible (less then 1 s). Moreover, we can notice that in the CloudLab testbed, the main impact of the servers characteristics regards the deployment time, while the instantiation and onboarding times undergo a minor decrease. The parsing time remains the same since the Intent layer is unchanged and runs on the same VM for both cases.

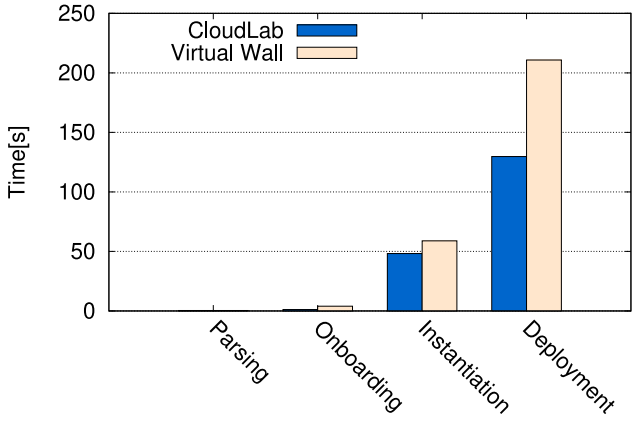

Fig. 7. Average execution time of the Intent framework elements: Virtual Wall vs. CloudLab.

6.3. Impact of the number of virtual functions

In this paragraph, we fix the number of OF switches in the SDN network to 10 and then we vary the number of virtual functions required in the intent. To do so, we have generated 5 different NSDs characterized by an increasing number of VFs, ranging from 2 to 6. For each case, 10 iterations have been performed and then the average value of the overall slice deployment time has been evaluated. Fig. 8 plots the average deployment time of the network service. The measurement of the deployment time starts at the creation, on the VIM, of the virtual machines on which the VNFs composing the NS will be deployed and ends after the fulfillment of their configuration. The configuration steps are the following: (i) the creation of a virtual machine using the VM image specified in the VNF descriptor, (ii) the performance of access-oriented configurations such as SSH keys, hostname, network interfaces, which mainly permit the access and the communication with the VM, (iii) the execution of the script defined in the cloud-init file of each VNF, which mainly consists in downloading the necessary software packages, their installation and the execution of the desired service. While the first two steps are almost identical for all the VNFs (except the number of network interfaces to be activated), the third step strictly depends on the type of VNF that will be deployed (e.g., SDN, Firewall, DPI, NAT). Fig. 8 shows that as the number of VFs increases, the overall deployment time linearly increases. This is mainly due to the

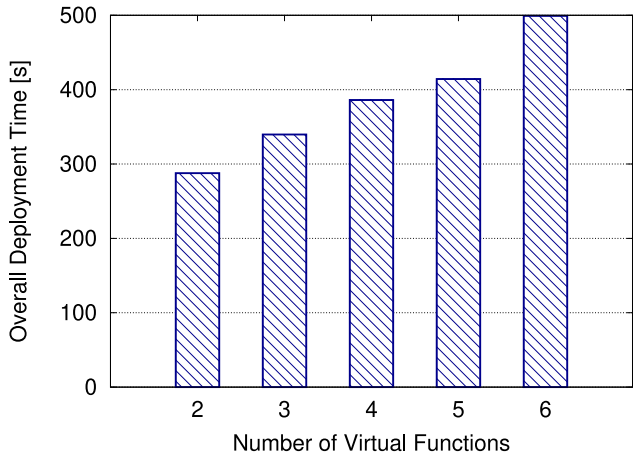

Fig. 8. Overall deployment time vs. number of virtual functions.

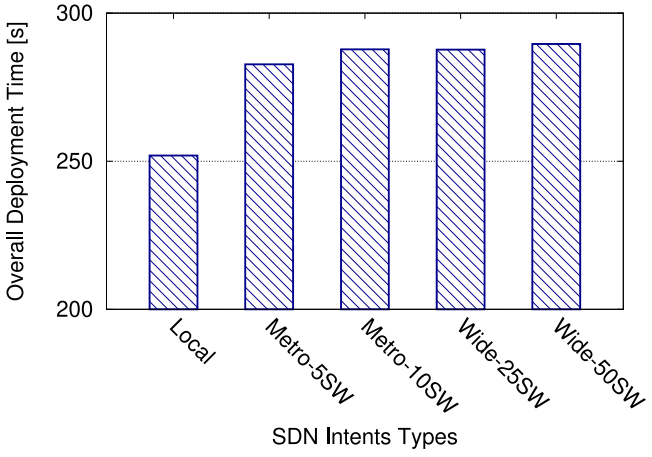

Fig. 9. Overall deployment time vs. number of switches in the SDN network.

characteristics of the OSM MANO NS process which handles the creation and the configuration of the VNFDs that constitute the NSD sequentially following the order of their appearance in the descriptor. Most of this time is dedicated to the configuration of the VM that contains the SDN controller and network (around 180 s) while each VF connected to the network increments the deployment time by around 53 s. This time is reduced to less than 10 s if we perform the experiments on the CloudLab testbed.

6.4. Impact of service scenario complexity

We then assess the impact of the SDN network complexity on the overall deployment time by varying the number of OF switches in the topology while keeping the number of VFs equal to 2. We repeat the experiment 10 times and we take the average of the overall deployment time. It is worth pointing out that in the Local service scenario, we considered an OvS switch [42] that we automatically configured to allow the connectivity between the VFs attached to it, while in the other cases we chose to use the Mininet topology emulator [43]. In fact, instead of focusing on the linear behavior of the deployment time due to the configuration of the OvS switches, in this paragraph we evaluate the impact of the topology complexity on the time necessary for the intent configuration. Fig. 9 shows that the most increase in the deployment time happens when we pass from 1 to 5 switches. This is explained by the fact that the adoption of Mininet requires the installation of further packages in the virtual machine hosting

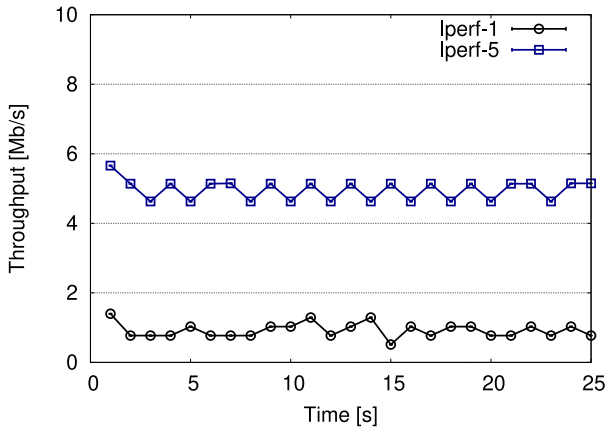

Fig. 10. Average throughput measured in the deployed slices.

the SDN-based topology, which is illustrated by this increase in the overall deployment time. Varying the number of switches in Mininet then does not have a significant impact on the results since the overall deployment time increases by around 7 s while expanding the SDN topology from 5 switches to 50 switches. It is worth pointing out that the SDN topology can be implemented either using a cluster of OF virtual switches (OvS) or the Mininet simulator. In this proof-of-concept we opted for Mininet. Indeed, the VF hosting it still needs to be deployed and configured in an automatic way like any other VFs.

6.5. Intent verification process

When expressing the intent, verticals can specify a throughput value that must be guaranteed between the VFs and the network topology. If the vertical does not specify anything, the value of the throughput is automatically assigned to 5 Mb/s, while we consider that the QoS level for all intents is fixed to "Gold". Once the slice is deployed, the connectivity among its different components is checked and the value of the throughput is measured. If the connectivity is not guaranteed (i.e., reliability issue) or the values of the throughput are not respected (i.e., the actual amount of throughput does not match the value specified in the intent objectives), an alarm is automatically sent to the intent layer to resolve the issue. This test is periodically performed to verify the status of the intent and carry out corrective actions if necessary.

In this paragraph, we send a request for the setup of 2 different slices with fixed throughput values: 1 Mb/s and 5 Mb/s. Both slices are composed of a simple SDN-based topology and two VFs connected to it. Once the slices are deployed, we automatically check the connectivity among the VFs and measure the throughput. To do so, we use Iperf, which is an open source and real-time network throughput measurement tool [44]. The first test is performed immediately after the deployment of the slices and the result is automatically sent to the intent layer, while the other verification tests are carried out periodically every 5 minutes. In average, this operation takes around 51 s to be completed. This time includes the configuration of the network interfaces, the connectivity establishment and the Iperf test which, in this example for consistency purposes, is performed over a 25 s time period. The overall verification time can be significantly reduced if we reduce the Iperf test time period (i.e., can be reduced up to 1 s).

In Fig. 10, we show the outcome of the Iperf tests for the 2 slices immediately after their deployment. The plot confirms the establishment of the connectivity among the VFs composing each slice. Moreover, we can clearly notice that for the whole duration

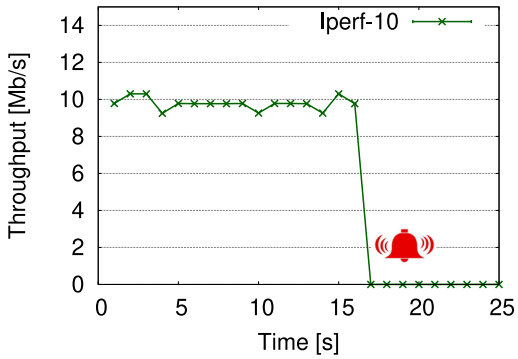

Fig. 11. Intent validation process.

of the test (i.e., 25 s), the throughput is almost equal to the value requested in the intent.

During the periodical validation tests, the connectivity could suffer from sudden degradations that are automatically reported to the Intent Layer thanks to the *Monitoring & Validation* component. Those degradations can happen during the validation test, as shown in Fig. 11 or before its execution. Once such deterioration in the performance is detected, an alarm is sent to the Intent engine, which undertakes a set of actions to fix the issue and restore the performance to the level required by the intent. As a preliminary implementation, our intent engine sends a *Terminate* command to NFV MANO to stop the execution of the slice, then it deploys the slice again considering the initial requirements of the intent. Such operation takes around 200 s in the CloudLab testbed and allows to restore the connectivity among the slice's components. In Fig. 12 we plot the time taken by each phase of the validation process to restore the performance. More specifically, we can clearly notice that most of the time (almost 90%) is devoted to the re-setup of the slice while the rest is taken by the deletion of the VMs and the communication messages between the OSM and Openstack.

7. Related works

7.1. Intent-based networking

IBN was conceived as a novel approach in network management to enable a loosely-coupled interworking between applications (i.e., tenant applications or service management applications) and network operators. Recently, IBN has attracted increasing attention from research works, which provided solutions addressing different aspects and challenges of IBN systems [45]. The concept of intent has also been introduced in the context of autonomic networks [46]. However, while both autonomic networks and Intent-based networking are driven by the need to simplifying network management through automation and under an end-to-end view, there is still a deep difference among them. In fact, the intent concept in autonomic networks is basically a kind of policy used to express instructions to operate the network, hence, not aware of the goals the network providers and service instances have to address like in intent-based networks.

In [47] authors present a high-level description of a slice intent for the creation and dynamic management of an application-aware network slice. Although only theoretical, the concept is presented within a smart city use case. In [48] authors introduce an IBN approach based on machine learning techniques to ensure network automation and self-assurance on top of SDN and NFV platforms. Some implementative aspects of the automated platform are described but not assessed. In [49], authors present an intent management system for VNF modeling and end-to-end orchestration. The system is tested using 2 different open source orchestrators showing its efficiency in creating network slices and providing a seamless service orchestration. However, after the deployment of the intents, the implementation lacks a monitoring and assurance module to check their correct setup and execution.

With respect to the workflow presented in Section 2.1, the IBN fulfillment operations have been tackled in [50–53][54], while the assurance operations have been addressed in [55–57]. More specifically, intent representation is considered in [50], where a service description framework, namely iNDIRA, is used to converse with users in a natural manner by adopting an ontology-based approach. The intent translation is tackled in [51] where authors use specific mechanisms to translate their intent into a set of configurations across SDN, IoT and Cloud domains while allowing to satisfy the QoS requirements. In [52], authors propose the NEtwork MOdeling (NEMO) language to be used for intent-based NBIs. NEMO provides a simple model definition that facilitates intents description and translation. The automation of resources management for SDN applications is addressed in [53] where an intent-driven approach is presented providing high-level APIs that allow to express the requirements of various applications simultaneously. A concrete implementation of the automation feature can be shown through the network programmability as stated in [54] where an end-to-end intent is deployed using the P4 programming language.

The assurance operations have been addressed in [55–57]. More specifically, [55] addressed the challenge of monitoring the status of deployed intents and adapting them to the changes in the underlying network by checking the current network status and then automatically rerouting services in order to dynamically align network flows to the desired intent. [56] developed an IBN framework which contains a machine learning module that continuously monitors the network resources statistics and decides whether upcoming requests can be admitted or not. [57] implements an intent-based control loop for video service assurance that triggers policy driven actions (e.g., functions placement, bitrate changes) to dynamically respond to service-specific requirements and improve the QoS.

All these works address IBN in different kinds of networks and on top of network control or management systems. In this paper, we address IBN in virtualized infrastructures whose resources are orchestrated through ETSI MANO framework. We present a comprehensive framework where all the building blocks of the Intent Orchestration Layer are implemented and assessed. Our work tackles all the aspects of IBN operations, going from the translation to the execution and the verification.

7.2. Intent-based networking in SDN/NFV

IBN in SDN/NFV infrastructures including 5G (and beyond) technologies, has been recently tackled in several works [58]. [59,60] present an intent-based provisioning framework with proper abstractions and intent engine operation to address the required set-up of 5G mobile backhauling services and cloud radio access network slices, respectively. The potentiality of IBNs for 6G wireless networks is elaborated in [27] where architectures, key techniques and emerging new products are comprehensively described as well as open issues and challenges. Finally, the VNF placement problem and the automated deployment of VNFs in cloud-based infrastructures through an intent-based approach is tackled in [61].

IBN in network slicing orchestration and service chaining is also considered in several works. [48] provides an IBN solution for network slice orchestration on top of NFV platforms, which allows to support high-level service requirements that are

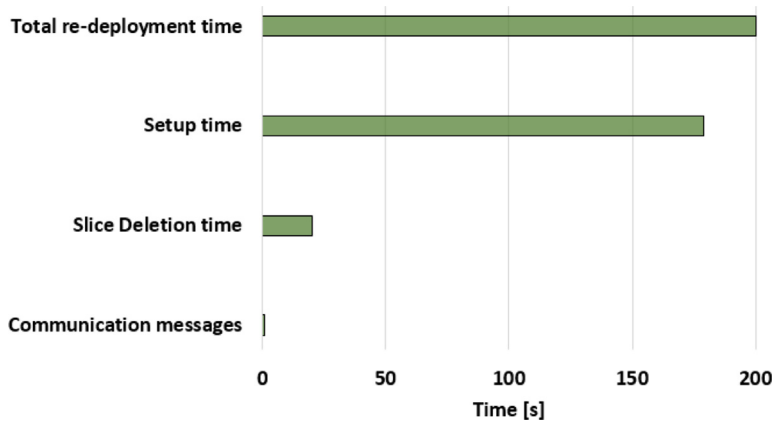

Fig. 12. Break-down of intent validation process phases.

then converted into low-level TOSCA configurations. Moreover, assurance operations are provided through monitoring and a machine learning model to guarantee a stable service provisioning. In [49], authors present an intent-based system for end-to-end network service orchestration on top of Mobile Central Office Re-architected as Data Center (CORD) and Open-Source Management and Orchestration (OSM) orchestrators. [62] proposes the usage of the intents to request transport network slices as part of an end-to-end slice for 5G vertical services. The work mainly focuses on the set of mechanisms required to translate the user's intent into a transport slice. [63] implements a northbound interface (NBI) supporting intents in plain text that are translated into pre-compiled network policies. The policies are then bound to the underlying Software-Defined Infrastructure management tool which interacts with a cloud platform and SDN controllers to enforce the low-level rules. The approach has been validated through a service function chaining and dynamic firewall programming use cases. [64] introduces an IBN-based solution, INSpIRE, which translates intents into a set of configurations to deploy specific service chains while meeting QoS and security requirements. INSpIRE relies on a score and clustering mechanism to determine the suitable set of VNFs required by the intent and allows for automatically performing traffic steering among them. It is based on the ETSI NFV MANO framework which has been extended to include novel elements that help in enforcing VNFs ordering preferences.

All these works address IBN in network slicing and orchestration on top of an NFV platform as our proposed system does. However, our work focuses on SDN-based network slices as a solution toward flexible delivery of vertical services.

7.3. European initiatives for intent-based networking

Several efforts have been carried out in the area of service management to provision network slices with vertical-specified features and to allow their set-up in automated way as much as possible. More specifically, the EU-funded 5G-TRANSFORMER project considers a set of simple slice service blueprints that allow for the deployment of vertical services without the need to specify the slice details [65,66]. The blueprint offers an abstraction of a VNFD and is translated, through mapping, to a NSD which allows for the slice deployment [67]. MATILDA is another EU-funded project where an intent-oriented approach has been considered [68]. A slice intent is specified as a service data model used to express the terms and constraints of the slice services, e.g., required network functions, maximum latency. Then, the overall intent information is mapped to the MATILDA metamodel, aiming to represent all the requirements that should be satisfied by a telco provider during the creation of a slice [47]. Although in both cases a vertical-oriented approach is used, they cannot be indicated as intent-based solutions. In fact, they formulate a slice service data model that is still strongly dependent on the NSD data models and they do not express neither business nor operational goals desired by the verticals, e.g., high-available slice service with redundant slice links. On the contrary, an intent-based approach has been considered in the 5G EVE project [69], where a flexible 5G end-to-end infrastructure is offered [70]. The platform extends the concept of blueprints already presented in the 5G TRANSFORMER project to allow vertical industries to deploy their services automatically. The intent-based tool translates text input introduced by users to the most suitable blueprints and parameters. The validation of the services execution is also performed through monitoring and KPI results are then summarized in a report. However, no real-time corrective actions are performed as we do in this work. Moreover, unlike in our proposed approach, those initiatives did not focus on slices with SDN capabilities and rather tackled the provisioning of vertical-oriented slices that do not require SDN features.

7.4. SDN in tenant managed domains

The importance of tenant-side network through SDN for flexible management of network services in the application (i.e., vertical) domain has been attested by number of research initiatives. [71] proposes a "Bring Your Own Controller" (BYOC) paradigm and presents a BYOC VISOR, an SDN virtualization platform that provides a customized, secure and scalable SDN services to cloud users. [72] proposes an integrated management and orchestration architecture for multi-tenant transport networks with tenant SDN controllers deployed as VNFs. The system is presented as a possible improvement of the ETSI NFV orchestration platform [15] in terms of fast connectivity establishment and decreased recovery time in case of failures. In both cases, the proposed solutions do not provide a fully tenant-side network management capabilities and still rely on the integration with management features of the underlying (virtual) infrastructure. In [17], authors propose a platform that provides tenants with latency-aware service function chaining management capabilities. Although truly tenant-side network control capabilities are provided along with abstraction and programmability features to some extent, the solution does not rely on OF nor offers full SDN-based network control capabilities to tenants. [73] presents an NFV platform that aims at automating the management and configuration of virtual networks based on high level tenant requirement specifications. The platform also plays the role of an SDN controller, which provides abstraction and control capabilities for independent, isolated and multi-tenant virtual networks.

With respect to our work, the main difference resides in the provisioning of the SDN features to tenants. In fact, while in [73] the platform relies on a unique SDN controller that handles network abstractions and control capabilities to all virtual networks, in our proposal we provide each tenant with a separate SDN controller that manages network configurations within the slice. Such approach guarantees more flexibility, efficiency and isolation among tenants. In [74], authors present a workload slicing scheme to handle data-intensive applications in a multi-edge cloud environment using an SDN controller designed to provide optimal traffic scheduling to VMs with minimum energy consumption. Moreover, in order to handle the huge amount of traffic, authors propose to extend the physical network into virtual SDN networks (vSDNs) managed by virtual controllers. The scheme has been also validated for a healthcare ecosystem in [75]. In [76] authors propose an architecture that integrates cloud and fog computing in the 5G environment using SDN and NFV technologies. A network service chaining model is presented where the focus is on high-level architectural issues related to virtualization and security. Although the papers discuss the integration of SDN and NFV in different architectures and scenarios, the presented work is not sufficiently focused on the network slicing concept as we do in this paper.

8. Future work and challenges

The intent-based framework that we propose in this paper presents a step forward in the adoption of intent-based network slicing for SDN vertical services, which is still not a well explored research area. However, the proposed intent layer is still under development and some of its aspects still need to be improved. Moreover, there are several challenges and open issues regarding IBN that will be discussed in the following.

8.1. Future work

First, as future work, we intend to extend our framework by incorporating further details related to a wider range of scenarios and applications. To do so, we plan to expand the intents template, add more fine-grained options and develop the formalized intent language expression to cover more parameters. The security area can be also covered using intents with an extension of the preliminary management framework shown by authors in [77].

Second, to increase the efficiency of the assurance phase, we look for the adoption of more sophisticated parameters and mechanisms. More specifically, several remediation actions can be stipulated according to the nature of the deterioration issue and its potential causes. In our work, as a preliminary implementation, we have already considered the deletion of the intent and the deployment of the slice once again considering the initial requirements of the intent. However, other technologies can be considered for this purpose, such as Artificial Intelligence and Machine Learning to predict users/applications behavior and consequently prepare remediation actions in an optimal manner, thus fully automating the configuration in the network.

Finally, regarding the performance evaluation of the framework, we plan to enhance our approach by investigating different network topologies in the SDN-based VNF, and testing the impact of the use of more realistic data sets and traffic patterns (e.g., Google workload).

8.2. Open challenges

The IBN concept has been recently presented by the IETF as a novel approach in network management to increase automation and simplify the process of network configurations. However, there are still several challenges and open issues that have to be investigated and addressed, which we discuss below [78].

8.2.1. Complexity

With the evolution of networks toward 6G and beyond, the complexity of network operation and management is increasing and accordingly business needs are changing making it mandatory for IBN to evolve in order to keep pace with the requirements of the new scenarios.

8.2.2. Integration with AI and machine learning

Although the integration with AI and Machine Learning technologies will improve the performance of IBN frameworks by increasing automation and flexibility, those technologies are still at their early stage and require further research efforts before reaching the maturity necessary for their integration with IBN.

8.2.3. Conflicting intents

The management of conflicting intents is one of the most important challenges that have to be considered given the impact that intents dependencies can have on the performance [79]. In fact, syntactically and semantically valid intent can still be infeasible due to inconsistency or conflicts with existing intents. Due to the complexity of service scenarios and network configurations, such dependencies are hard to predict and detect at the reception of intents requests and necessitate the implementation of sophisticated mechanisms to mitigate their effects.

9. Conclusions

In this paper, we presented an intent-based framework that allows for customized specifications and effective establishment of QoS-aware and SDN-enabled network slices toward flexible delivery of vertical services. Our approach offers a straightforward way for verticals to express their business and operational goals using a simple approach based on the NILE language while the technical details for their implementation are left to the network operation system. Our system includes an intent layer integrated with an ETSI NFV MANO platform thus providing the necessitated flexibility and dynamicity. The system is put in relation with different service scenarios where network slicing and SDN can foster the flexible delivery of vertical services. Moreover, intent specifications are detailed that reflect those service scenario goals. We have also reported a preliminary proof-of-concept implementation of the proposed intent layer. The obtained results show its feasibility and ability to easily deploy the intents while guaranteeing tenants objectives. As future work, we plan to improve the expressiveness of our intents by leveraging more sophisticated programming languages or libraries. We also envisage to add more features to the validation process to make the provision of the network slices more reliable.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work has been partially funded by the 5Growth Project (Grant No. 856709) and by the Department of Excellence in Robotics and Artificial Intelligence funded by MIUR, Italy to Scuola Superiore Sant'Anna.

References

[1] B. Bloching, et al., The digital transformation of industry - How important is it? Who are the winners? What must be done? 2015.
[2] X. Li, et al., Network slicing for 5G: Challenges and opportunities, IEEE Internet Comput. 21 (5) (2017) 20–27.
[3] V.A. Cunha, et al., 5 growth: Secure and reliable network slicing for verticals, in: Joint European Conference on Networks and Communications & 6G Summit, EuCNC/6G Summit, 2021, pp. 347–352.
[4] A. Hakiri, et al., Leveraging SDN for the 5G networks: Trends, prospects and challenges, 2015, preprint arXiv:1506.02876.
[5] C. Casetti, et al., Network slices for vertical industries, WCNCW, in: IEEE Wireless Communications and Networking Conference Workshops, 2018, pp. 254–259.
[6] A. Clemm, et al., Intent-based networking - concepts and overview, 2020, [Online] Available: https://bit.ly/2imukbf.
[7] E. Zeydan, Y. Turk, Recent advances in intent-based networking: A survey, in: IEEE 91st Vehicular Technology Conference, VTC2020-Spring, Antwerp, Belgium, 2020, pp. 1–5.
[8] Q. Sun, W. Liu, K. Xie, An Intent-driven Management Framework, [online] https://tools.ietf.org/html/draft-sun-nmrg-intent-framework-00.
[9] C. Li, et al., Intent classification, 2020, https://tools.ietf.org/pdf/draft-li-nmrg-intent-classification-03.pdf.
[10] F. Paganelli, M. Ulema, B. Martini, Context-aware service composition and delivery in NGSONs over SDN, IEEE Commun. Mag. 52 (8) (2014) 97–105, http://dx.doi.org/10.1109/MCOM.2014.6871676.
[11] M. Gharbaoui, et al., Implementation of an intent layer for SDN-enabled and QoS-aware network slicing, in: IEEE Conference on Network Softwarization, NetSoft, 2021.
[12] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J.J. Ramos-Munoz, J. Lorca, J. Folgueira, Network slicing for 5G with SDN/NFV: Concepts, architectures, and challenges, IEEE Commun. Mag. 55 (5) (2017) 80–87.
[13] Network functions virtualisation (NFV): Architectural framework, ETSI GS NFV 2 (2) (2013) V1.
[14] A. Israel, et al., OSM release three, Open Source MANO Tech. Overview (2017).
[15] Network functions virtualisation (NFV); management and orchestration, ETSI GS NFV-MAN 001 (2014) 1.1.1.
[16] F. Malandrino, C.F. Chiasserini, G. Landi, Service shifting: A paradigm for service resilience in 5G, IEEE Commun. Mag. 57 (9) (2019) 120–125, http://dx.doi.org/10.1109/MCOM.2019.1800986.
[17] G. Cuffaro, et al., Tenant-side management of service function chaining:architecture, implementation and experiment on a future internet testbed, in: Network Softwarization (NetSoft), 2019 1st IEEE Conference.
[18] L.M. Contreras, D.R. López, A network service provider perspective on network slicing, IEEE Softwarization (2018).
[19] B. Chen, et al., Edge computing in IoT-based manufacturing, IEEE Commun. Mag. 56 (9) (2018) 103–109.
[20] X. Li, et al., Adaptive transmission optimization in SDN-based industrial internet of things with edge computing, IEEE Internet Things J. 5 (3) (2018) 1351–1360.
[21] A.A. Abdellatif, et al., Edge computing for smart health: Context-aware approaches, opportunities, and challenges, IEEE Netw. 33 (3) (2019) 196–203.
[22] J. Baranda, et al., NFV service federation: enabling multi-provider ehealth emergency services, in: IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS, 2020, pp. 1322–1323.
[23] M.S. Munir, et al., Intelligent service fulfillment for software defined networks in smart city, in: International Conference on Information Networking, ICOIN, 2018, pp. 516–521.
[24] D. Z.Alotaibi, Isolation model for network slicing in 5G network, Arab J. Sci. Publ. (2020) 29–45.
[25] B. Martini, M. Gharbaoui, P. Castoldi, Intent-based zero-touch service chaining layer for software-defined edge cloud networks, Comput. Netw. 212 (2022) 109034.
[26] M. Gharbaoui, et al., An experimental study on latency-aware and self-adaptive service chaining orchestration in distributed NFV and SDN infrastructures, Comput. Netw. 208 (2022) 108880.
[27] Y. Wei, et al., Intent-based networks for 6G: Insights and challenges, Digit. Commun. Netw. (2020).
[28] A. Jacobs, et al., Refining network intents for self-driving networks, in: Proceedings of the Afternoon Workshop on Self-Driving Networks, SelfDN 2018, Association for Computing Machinery, 2018, pp. 15–21.
[29] A. Jacobs, et al., Hey, lumi! Using natural language for intent-based network management, in: USENIX Annual Technical Conference, USENIX ATC 21, USENIX Association, 2021, pp. 625–639.
[30] ISO/IEC 14977:1996 1996. Information technology – Syntactic metalanguage – Extended BNF. Standard.
[31] OSM, https://osm.etsi.org/.
[32] M. Gharbaoui, et al., Programmable and automated deployment of tenant-managed SDN network slices, in: IEEE/IFIP Network Operations and Management Symposium, NOMS, 2020, pp. 1–6.
[33] H. Khalili, et al., Network slicing-aware NFV orchestration for 5G service platforms, in: European Conference on Networks and Communications, EuCNC, 2019, pp. 25–30.
[34] OpenStack Official Site, https://www.openstack.org.
[35] Virtual Wall Web Site, https://doc.ilabt.imec.be/ilabt/virtualwall/index.html.
[36] S. Li, et al., Source routing with protocol-oblivious forwarding (POF) to enable efficient e-health data transfers, in: IEEE International Conference on Communications, ICC, 2016, pp. 1–6.
[37] B. Preveze, et al., SDN-driven internet of health things: A novel adaptive switching technique for hospital healthcare monitoring system, Wirel. Commun. Mob. Comput. J. (2022).
[38] F. Naeem, et al., SDN-enabled energy-efficient routing optimization framework for industrial internet of things, IEEE Trans. Ind. Inform. 17 (8) (2021) 5660–5667.
[39] M. Senouci, et al., Qoe-based network interface selection for heterogeneous wireless networks: A survey and e-health case proposal, in: IEEE Wireless Communications and Networking Conference, 2016, pp. 1–6.
[40] M.E. Khaddar, et al., Emerging wireless technologies in e-health trends, challenges, and framework design issues, in: International Conference on Multimedia Computing and Systems, 2012, pp. 440–445.
[41] CloudLab Web Site, https://www.cloudlab.us/.
[42] Open vSwitch Web Site, https://www.openvswitch.org/.
[43] Mininet Official Site, http://mininet.org/.
[44] https://iperf.fr/.
[45] E. Zeydan, Y. Turk, Recent advances in intent-based networking: A survey, in: 2020 IEEE 91st Vehicular Technology Conference, VTC2020-Spring, 2020, pp. 1–5.
[46] M. Behringer, et al., Autonomic Networking: Definitions and Design Goals, 2015, https://tools.ietf.org/html/rfc7575.
[47] B. Rusti, et al., 5G smart city vertical slice, in: IFIP/IEEE IM Workshop on Hot Topics in Network and Service Management, HotNSM, 2019.
[48] T.A. Khan, et al., Intent-based orchestration of network slices and resource assurance using machine learning, in: IEEE/IFIP Network Operations and Management Symposium, NOMS, 2020, pp. 1–2.
[49] A. Rafiq, et al., Intent-based end-to-end network service orchestration system for multi-platforms, Sustainability 12 (7) (2020).
[50] M. Kiran, et al., Enabling intent to configure scientific networks for high performance demands, Future Gener. Comput. Syst. 79 (2018) 205–214.
[51] W. Cerroni, et al., Intent-based management and orchestration of heterogeneous openflow/IoT SDN domains, in: IEEE Conference on Network Softwarization, NetSoft, 2017, pp. 1–9.
[52] S. Hares, NEtwork MOdeling (NEMO) language, https://tools.ietf.org/html/draft-xia-sdnrg-nemo-language-03.
[53] V. Heorhiadi, et al., Intent-driven composition of resource-management SDN applications, in: Int. Conf. Emerg. Netw. Exp. Technol., CoNEXT, 2018, pp. 86–97.
[54] B. Lewis, et al., Using P4 to enable scalable intents in software defined networks, in: Proc. IEEE 26th Int. Conf. Netw. Protocols, ICNP, 2018, pp. 442–443.
[55] D. Sanvito, et al., Enabling external routing logic in ONOS with intent monitor and reroute service, in: Proc. IEEE Conf. Netw. Softwarization Workshops, NetSoft, 2018.
[56] K. Abbas, et al., Slicing the core network and radio access network domains through intent-based networking for 5G networks, Electronics 9 (10) (2020).
[57] N.F.S. de Sousa, et al., Intent-based orchestration of network slices and resource assurance using machine learning, in: Workshop de Gerência e Operação de Redes e Serviços, 2019.
[58] L. Pang, et al., A survey on intent-driven networks, IEEE Access 8 (2020) 22862–22873, http://dx.doi.org/10.1109/ACCESS.2020.2969208.
[59] T. Subramanya, R. Riggio, T. Rasheed, Intent-based mobile backhauling for 5G networks, in: 12th International Conference on Network and Service Management, CNSM, 2016, pp. 348–352.
[60] F. Aklamanu, et al., Intent-based real-time 5G cloud service provisioning, in: 2018 IEEE Globecom Workshops, GC Wkshps, 2018, pp. 1–6, http://dx.doi.org/10.1109/GLOCOMW.2018.8644457.
[61] A. Leivadeas, et al., VNF placement problem: A multi-tenant intent-based networking approach, in: 24th Conference on Innovation in Clouds, Internet and Networks and Workshops, ICIN, 2021.
[62] L. Contreras, et al., Transport Slice Intent, https://tools.ietf.org/html/draft-contreras-nmrg-transport-slice-intent-00.

[63] F. Esposito, et al., A behavior-driven approach to intent specification for software-defined infrastructure management, in: IEEE Conference on Network Function Virtualization and Software Defined Networks, NFV-SDN, 2018, pp. 1–6.
[64] E.J. Scheid, et al., INSpIRE: Integrated NFV-based intent refinement environment, in: Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage., IM, 2017, pp. 186–194.
[65] 5G-Transformer, http://5g-transformer.eu/.
[66] X. Li, et al., Service orchestration and federation for verticals, in: IEEE Wireless Communications and Networking Conference Workshops, WCNCW, 2018, pp. 260–265, http://dx.doi.org/10.1109/WCNCW.2018.8369008.
[67] A. de la Oliva, et al., 5G-transformer: Slicing and orchestrating transport networks for industry verticals, IEEE Commun. Mag. 56 (8) (2018) 78–84.
[68] Matilda, https://www.matilda-5g.eu/.
[69] 5G-EVE, https://www.5g-eve.eu/.
[70] W. Nakimuli, et al., Automatic deployment, execution and analysis of 5G experiments using the 5G EVE platform, in: IEEE 3rd 5G World Forum, 5GWF, 2020, pp. 372–377.
[71] H. Wang, et al., Bring your own controller: Enabling tenant-defined SDN apps in IaaS clouds, in: IEEE Conf. on Computer Communications, IEEE INFOCOM 2017, 2017, pp. 1–9.
[72] R. Munoz, et al., Integrated SDN/NFV management and orchestration architecture for dynamic deployment of virtual SDN control instances for virtual tenant networks [invited], IEEE/OSA J. Opt. Commun. Netw. 7 (11) (2015) B62–B70.
[73] Y. Han, et al., An intent-based network virtualization platform for SDN, in: 12th International Conference on Network and Service Management, CNSM, 2016, pp. 353–358.
[74] G. Aujla, et al., Optimal decision making for big data processing at edge-cloud environment: An SDN perspective, IEEE Trans. Ind. Inform. 14 (2) (2018) 778–789.
[75] G. Aujla, et al., SAFE: SDN-assisted framework for edge–cloud interplay in secure healthcare ecosystem, IEEE Trans. Ind. Inform. 15 (1) (2019) 469–480.
[76] R. Chaudhary, et al., Network service chaining in fog and cloud computing for the 5G environment: Data management and security challenges, IEEE Commun. Mag. 55 (11) (2017) 114–122.
[77] B. Martini, et al., Pushing forward security in network slicing by leveraging continuous usage control, IEEE Commun. Mag. 58 (7) (2020) 65–71, http://dx.doi.org/10.1109/MCOM.001.1900712.
[78] Y. Wei, M. Peng, Y. Liu, Intent-based networks for 6G: Insights and challenges, Digit. Commun. Netw. 6 (3) (2020) 270–280.
[79] S. Mwanje, et al., Intent-driven network and service management: Definitions, modeling and implementation, TechRxiv (2021) https://doi.org/10.36227/techrxiv.17075450.v1.

B. Martini is an Associate Professor at Universitas Mercatorum, Rome, Italy and an Affiliate Researcher with Sant'Anna School of Advanced Studies, Pisa, Italy. Formerly she was Head of Research with the CNIT National Laboratory of Photonic Networks and Technologies (PNT Lab) and worked for two large telco companies, Italtel and Marconi Communications (currently Ericsson). Her research interests include network virtualization and orchestration in SDN/NFV/5G environments, service platforms for next-generation networks, network control/management architectures, and security solutions for multi-domain IP/optical net-works and NFV deployments. She is the coordinator of Ph.D. Programme in "Big Data and Artificial Intelligence" and serving with chairing roles in several international conference in her research field (ICIN, NetSoft, SDN-NFV). She has been involved in several national/EU research projects, the recent ones 5GPPP 5GEX, 5GTRANSFORMER, and 5GROWTH, and in several FIRE projects (OFELIA, FED4FIRE+,TRIANGLE, and 5GINFIRE) with leading roles. She has co-authored more than 100 papers in international journals and conference proceedings.

M. Gharbaoui is an Assistant Professor at the Scuola Superiore Sant'anna, Pisa, Italy. She received her Ph.D. degree in Innovative Technologies of Information & Communications Engineering and Robotics in 2012 from the Scuola Superiore Sant'anna, Pisa. Her main research interests are in the field of software-defined networking, network function virtualization, network orchestration, the development and the implementation of service-oriented architectures and service management for smart cities. She has been involved in several EU projects and has co-authored more than 50 papers appeared in international journals and conferences.

P. Castoldi has been a professor at Scuola Superiore Sant'Anna, Pisa, Italy since 2001. He was abroad at Princeton University (USA) overall about two years in 1996, 1997, 1999, 2000. He is currently a leader of the "Net-works and Services" research area at Scuola Superiore Sant'Anna, Pisa. His research interests cover telecommunications networks and system both wired and wireless, and more recently reliability, switching paradigms and control of optical networks, including application-network cooperation mechanisms for cloud networking. He is an IEEE Senior Member and he is an author of more than 400 international publications.