Computer Networks 212 (2022) 109034

Contents lists available at ScienceDirect

Computer Networks

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

Intent-based zero-touch service chaining layer for software-defined edge cloud networks

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

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

ARTICLE INFO

Keywords:
IBN
Service chaining
Edge networking
SDN
NFV

ABSTRACT

Edge Computing, along with Software Defined Networking and Network Function Virtualization, are causing network infrastructures to become as distributed clouds extended to the edge with services provided as dynamically established sequences of virtualized functions (i.e., dynamic service chains) thereby elastically addressing different processing requirements of application data flows. However, service operators and application developers are not inclined to deal with descriptive configuration directives to establish and operate services, especially in case of service chains. Intent-based Networking is emerging as a novel approach that simplifies network management and automates the implementation of network operations required by applications.

This paper presents an intent-based zero-touch service chaining layer that provides the programmable provision of service chain paths in edge cloud networks. In addition to the dynamic and elastic deployment of data delivery services, the intent-based layer offers an automated adaptation of the service chains paths according to the application's goals expressed in the intent to recover from sudden congestion events in the SDN network. Experiments have been carried out in an emulated network environment to show the feasibility of the approach and to evaluate the performance of the intent layer in terms of network resource usage and adaptation overhead.

1. Introduction

Thanks to 5G (and beyond) mobile networks, Internet of Things (IoT) and a large set of smart devices (e.g., smart glasses, smart cars), novel vertical applications are emerging in different industry fields (e.g., telemedicine, hyper-connected smart cities, and industrial automation) that will improve several aspects of society and human lives. In this context, pervasive cloud deployments extended to the edge (i.e., Edge Computing (EC)) are essential to augment computing capabilities, run distributed services and assure the interactive behaviour that such emerging applications require [1].

Different operational and architectural design have been conceived for implementing EC in different contexts and with different targets, e.g., Multi-Access Edge Computing (MEC) [2], Fog Computing [3], Cloudlet [4]. Established in telecommunication field, MEC considers providing cloud capabilities to the access network (i.e., base stations or Point of Presence (PoP)) so as to improve the quality of offered services (e.g., real-time Quality of Service (QoS) through adaptive data throughput closer to the users) [5]. A correlated trend is Network Function Virtualization (NFV) with network functions (e.g., firewall, Network Address Translation (NAT), Deep Packet Inspection (DPI)) deployed as virtual appliances (i.e., Virtual Network Functions (VNFs)) and flexibly provisioned in distributed edge telco clouds thereby enabling innovative network service delivery models for telco service providers through dynamic service chaining [6]. In this context, service-centric control structures for edge networks would be desirable to effectively establish and maintain service chains across edge clouds according to expressed requirements [7].

In all EC schemes, Software-Defined Networking (SDN) is considered an attractive network solution to simplify the management of the network, better utilize network resources and facilitate virtualization within the network. In particular, SDN provides more efficient and agile procedures for the establishment of data delivery paths connecting specified sequences of virtual service or network components, i.e., dynamic service chaining [8]. SDN also brings additional benefits to the EC architecture in terms of service-centric networking [7,9] and support of a northbound interface for third-party network

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

https://doi.org/10.1016/j.comnet.2022.109034
Received 6 October 2021; Received in revised form 26 April 2022; Accepted 7 May 2022
Available online 18 May 2022

applications to support EC services in highly dynamic ecosystem of users, devices, and data communication requirements [10]. However, the semantic gap between the business and the operational goal of application providers and the network delivery potential necessitates the underlying network to constantly (and consistently) adapt, protect, and inform across all strands of the service-oriented landscape [11]. For this reason, Intent-Based Networking (IBN) [12] comes into play as a novel approach in network management to (i) enable a loosely-coupled interworking between applications (i.e., vertical applications or service management applications) and network operators, and (ii) further foster the development of third-party applications thanks to abstractions and semantics not available at the SDN northbound interface. Indeed, application developers are not inclined to specify low-level technical parameters (e.g., Virtual Local Area Network (VLAN) tags, forwarding rules, connection points) needed to realize their business and operational goals in a specific application domain. Nevertheless, SDN remains a prominent building block in the migration towards 5G (and beyond) and intents support at the network edge.

In this paper, an intent-based and zero-touch service chaining layer for software-defined EC is presented, allowing for adaptive service chain paths and offering an effective interaction with the edge computing-enabled applications. More specifically, the proposed solution provides an Intent Layer with autonomy and assurance features for:

- dynamic enforcement and configuration of service chain paths through SDN capabilities established at the network edge and starting from high-level requests expressed by applications through intent-based northbound REpresentational State Transfer (REST) interface. This Application Programming Interface (API) allows applications to request the set-up and tear-down of data delivery services throughout chains of virtual functions using application-oriented semantics (i.e., prescriptive rather than descriptive directives) thus avoiding to deal with technology-specific low-level implementation details (fulfilment phase).
- autonomous adaptation of service chain paths according to the user intents and without requiring the user to report the intent deviations from the desired outcome and to specify the detailed technical steps for how to achieve that outcome (i.e., zero-touch). Instead, the Intent Layer triggers the required configuration changes in the network in order to maintain the intent and assure the desired outcome leveraging the continuous monitoring of the current status on network nodes and links during the whole lifecycle of the intent (assurance phase). As a reconfiguration trigger we considered the overload of a significant subset of switches according to a load balancing criteria.

A number of research works recently addressed novel solutions for IBN in different contexts, namely network automation [13], service orchestration [14], multi-domain networking [15] and data plane programming [16,17]. Regarding the adoption of IBN in the EC context, very few works tackled this issue and are limited to the use of the intents in specific scenarios (e.g. vehicular networks [11,18]). On the other hand, numerous works investigated the use of IBN techniques in SDN environments [19–21]. Although the proposed approaches go into the direction of seamless and automated provisioning of network paths, most of them still rely on complex network configurations that hinder the interactions with the IBN framework. This is in contrast with the straightforward template-based approach proposed in this work, which aims at facilitating the deployment of service chains in edge computing networks. Few works tackled the adoption of IBN in Service Function Chaining (SFC) contexts [14,22,23] where only some aspects were addressed while in this work all the intent lifecycle management features across edge clouds are implemented and assessed. Finally, the functionalities of the proposed approach are in line with the current trends and initiatives in both dynamic service chaining and IBN carried out by standardization bodies such as Internet Engineering Task Force (IETF) [24,25].

This work has been inspired by authors previous work [26,27] on the orchestration of dynamic service chains in SDN networks, and by [28] on how to express slice intent requests through a template-based approach. In this paper, the previous proposal for dynamic service chaining in SDN networks in [26,27] is further enhanced by designing and implementing an intent layer for service chaining including zero-touch assurance. The proposal on intent-based networking beyond network slicing presented in [28] is also extended with an intent layer for edge networks for service chaining. The new features introduced by the Intent Layer offer a service-oriented support for applications to interact with the network aiming at the enforcement of the service chaining paths and a more dynamic reaction to performance deviations so as to assure service data delivery performance. Indeed, the continuous monitoring of the network status during the assurance phase automatically guarantees the high availability of the service chains.

The remainder of the paper is organized as follows. Section 2 gives a background on intent-based networking. Section 3 presents the reference scenario that motivates the use of an Intent Layer in SDN-enabled edge computing infrastructures. Section 4 details the main components of the intent-based framework architecture, while Section 5 highlights the features brought by specific operations of the proposed Intent Layer. Section 6 evaluates the performance of the presented approach under different assumptions. Section 7 discusses the state of the art and recent standardization initiatives. Finally, Section 8 concludes the paper.

2. Background on intent-based networking

The IBN concept represents a new approach to network management conceived by the IETF 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 [12].

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 *fulfilment* and *assurance* operations in a closed loop workflow. More specifically:

- *fulfilment* 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

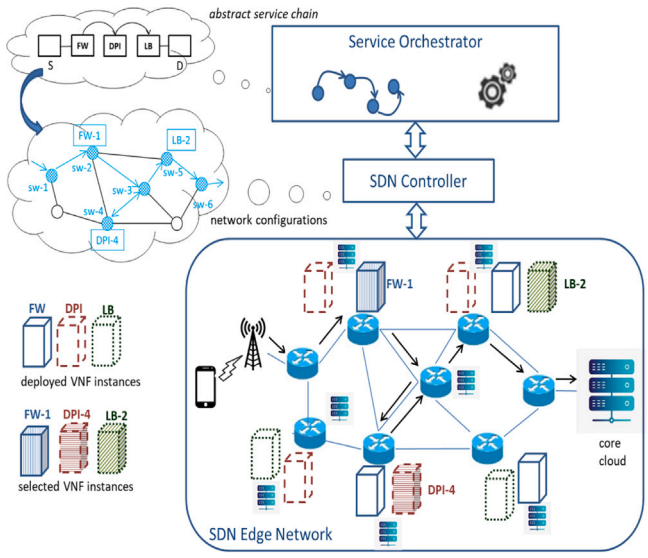

Fig. 1. Service chaining in NFV/SDN-enabled edge network and cloud infrastructure.

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 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 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 through closed loop and through continuous learning to reduce costs through dynamic network operation and maintenance [29].

3. Reference scenario

In this section, the reference scenario that motivates the use of an Intent Layer in SDN-enabled edge network and computing infrastructures for the benefit of vertical applications is presented.

The network scenario is shown on the bottom of Fig. 1 and reproduces a typical set-up of a SDN-based network infrastructure in edge computing service scenarios (either Fog Computing, Cloudlet or MEC) where virtualized service or network functions are deployed as virtual appliances over distributed edge computing nodes (i.e., micro-sized servers in cloud platforms) co-located with and interconnected by SDN-enabled network nodes managed via an SDN controller. Multiple instances of each kind of virtualized service or network functions are also considered that are operated in those geographically-distributed servers as replications to allow for (i) keeping the service quality at the highest level anywhere at the edge, (ii) enlarging service coverage over a wider geographical area, and (iii) running back-up services for high-critical applications [10]. As an example, in Fig. 1 multiple instances of VNFs (i.e., firewalls, deep packet inspectors and load balancers) are shown over distributed edge servers. This distributed service deployment scenario is peculiar of edge computing set-up where service requests are typically fulfilled as a concatenation of dynamically selected service/network function instances (i.e., dynamic service chaining) as part of end-to-end services delivery to users.

The complexity of such a combined scenario of edge clouds and network requires cross-layer control mechanisms to orchestrate virtual resources and cloud-like services in such a distributed environment. For this reason, the reference scenario also includes a Service Orchestrator shown on top of Fig. 1. In general terms, service orchestrators are in charge of operational and functional processes involved in designing, creating, and delivering end-to-end application services (e.g., e-health, face recognition, entertainment) or network services (e.g., virtual Radio Access Networks (RANs) and core networks). In case of application services, service orchestration is performed by Service Delivery Platforms usually inspired by Service Oriented Application (SOA) principles [30] and horizontally-integrated Service Overlay Network (SON) architectures, such as Next Generation SON (NGSON) [31]. In case of network services, Service Orchestrators are from the NFV ecosystem. Indeed, NFV is another key technology that supports service orchestration [32], through the NFV Orchestrator (NFVO) following the European Telecommunications Standards Institute (ETSI) Management and Orchestration (MANO) specifications [33]. In either cases, service orchestration is based on specifications to define a composite service as a flow of functional capabilities (i.e., service components or network functions) that need to be sequentially invoked to deliver the end-to-end service (e.g., *abstract service chain specification*) [34]. Accordingly, service chain requests are generated and issued to the network layers for path set-up.

This deployment scenario affects the service function chaining problem since it requires a novel service-centric structure and service control layers for edge networks to effectively establish service chains across edge clouds in a dynamic environment featured by multiple available service instances with different and variable loads over time. Indeed, as opposed to host-centric solutions, service-centric operations in edge networks prevent end-systems (e.g., user devices) to be aware of the service end-points to resolve their Internet Protocol (IP) address dynamically (e.g., based on context information) since relying on specific service-aware components to effectively track the locations of service instances and select the most proper one [7]. It is agreed that SDN can provide several benefits in this scenario thanks to software-based and centralized control over the network. In particular, thanks to the SDN capability to have a general view of the network and to act programmatically on network nodes, the SDN controller can (i) track the locations of service instances under its coverage [35], (ii) better select the most appropriate ones for the incoming service request, and (iii) accordingly handle connectivity among service components or network functions so as to provision service chains (service chain path set-up) [36].

As shown in Fig. 1, the chain composed of a firewall (FW), a deep packet inspector (DPI) and a load balancer (LB) is established by identifying the FW-1, DPI-4 and LB-2 as proper instances and by enforcing the proper flow rules on switches sw-1 to sw-6 so as to set-up the chain paths.

Within SDN, a northbound interface (NBI) has been designed to provide a common interface between the SDN controller and the wide range of network applications that can be developed on top. Recently, additional abstractions have been devised at the SDN controller NBI to specify connectivity requests between two end-hosts [37]. However, there is an inherent semantic gap between the way the service chains are handled by service orchestrators (i.e., abstract service chain specifications) and the way SDN controllers handle connectivity and the flow rules set-up across the switches. Indeed, the NBI is conceived for network specialists to mainly handle packet-level forwarding and data monitoring and not to handle multi-hop logical connectivity among virtualized appliances on overlay networks. On the other hand, service developers are not inclined to specify descriptive configuration directives and continuously check low-level technical parameters of forwarding rules settings, topology and protocols (e.g., VLAN tags, connection points) since they are more focused on realizing their business and operational goals (e.g., service delivery and reliability).

On top of that, network operators aim at addressing customers goals driven by their own optimization process on resource usage and at automating operations for network services and connectivity set-up to lower operational costs. The matching between these two dispositions is possible through higher-level interfaces to foster the development of third-party applications and to provide the connectivity services with abstractions for service chaining not available at SDN northbound interface.

From the above considerations, an intent-based approach is proposed in this work supported by a middleware layer (i.e., Intent Layer) between Service Orchestrators and SDN controllers as part of a service orchestration process (either at application or at network service level) for dynamic service chaining in a distributed deployment scenario at edge. In particular, in the following section we present an Intent Layer for the dynamic service function chain enforcement (including assurance) across a set of edge clouds interconnected via SDN and hosting multiple service instances. If the service function chaining problem is typically tackled into the edge clouds (i.e., within computing/data center networks) [38], in this work an extension of such a problem across distributed edge clouds is considered involving SDN and intents to address service function chain paths in a dynamic and application/service-oriented way [39].

4. Intent layer design for edge networks

In this section, the intent framework for SDN edge cloud networks is presented aimed at the provisioning of adaptive service data paths to applications that are therefore unburdened from using prescriptive directives. Fig. 2 presents the intent-based framework in terms of functional design and the key aspects of the overall proposed architecture. The framework extends a SDN-enabled Network Layer with an Intent Layer that automatically handles the SDN edge network resource management capabilities (e.g., data forwarding, load information), thus allowing for a flexible and efficient configuration of network paths while continuously satisfying the requirements specified in the intent. The Intent Layer elaborates intents from an Application Layer that represents any entity that is interested in automatically deploying service chains with specific throughput guarantees in a SDN-based edge network and computing infrastructure, i.e., Service Orchestrators. In order not to deal with specifications or constraints on selection, allocation, concatenation of required resources (e.g., mapping to VLAN tags, connection points), they express prescriptive directives to the Intent Layer in terms of desired outcome and operational or business goal to achieve, i.e., intents. This is in accordance with the principle that applications ask the network *"what to do"* and not *"how to do it"* [40].

The main components of the Intent Layer are described below. Although the presented building blocks and interactions are valid for a generic intent, for their description the case of a service chain intent is mainly considered.

- *Intent Manager*: exposes a NBI to handle the intents received from the Application Layer and takes care of any operation to fulfil the intent requests. More specifically, it performs the *Translation* operation to parse the intent and derive the operational steps necessary to meaningfully configure the network according to the user's goals and requirements expressed in the Intent. This can be achieved through different approaches such as the usage of deep neural networks to understand human specific language, the adoption of Yet Another Next Generation (YANG) and Topology and Orchestration Specification for Cloud Applications (TOSCA) based models to represent intents [41], etc. The Intent Manager also performs the *Verification* operation to verify the translated intents can be executed in the network according to its current state (i.e., *execution verification*). If the intent can be executed, it is admitted and the *Intent Manager* triggers the *Configuration and Deployment Engine* to enforce it in the SDN edge network. The *Intent Manager* is also triggered by the Adaptation Module in case the *validity verification* check is not passed during the intent lifecycle in order to trigger re-optimization or remediation actions to recover the expected performance as required by the expressed intent. First, the Intent Manager checks again the eligibility of the intent by verifying the feasibility of its deployment according to the new status of the network. As at the time of the original intent request, it makes sure that there are enough resources available to answer the request (i.e., an intent can be directly blocked if for example all the switches in the network are overloaded) and guarantees that the deployment of the intent with its specific requirements (e.g., amount of throughput) will not impact the existing intents in the system (i.e., the redirection of the intent will not cause the congestion of other switches.) If the intent request is eligible, the Intent Manager then forwards the request to the Configuration and Deployment Engine to calculate and enforce a new (part of) service chain path.
- *Configuration and Deployment Engine*: responsible for coordinating the provisioning actions of the end-to-end service data paths along the chain to enforce the intent in the network upon triggering by the Intent Manager. The forwarding of the packets is required across multiple segments composing the end-to-end delivery path where a segment is a path along the switches starting from a VNF (or a source endpoint) and terminating at the next VNF (or the final destination endpoint) in the logical chain. Thus, a number of different provisioning actions need to be coordinated under the same workflow of service chain set-up to address the intent request. Each provisioning action aims at the set-up of each segment, so that their combination allows to accomplish the whole end-to-end path setup of the service chain expressed in the intent. The *Configuration and Deployment Engine*, other than acting for the intent fulfilment, also acts for re-enforcing (part of) service chain along a different path in the network if remediation actions are required. In either cases, the *Configuration and Deployment Engine* interacts with the SDN controller through appropriate APIs to handle the low-level directives (i.e., Openflow messages [42]) to install the forwarding rules throughout the network switches for each delivery path that is needed for the end-to-end service chain set-up.

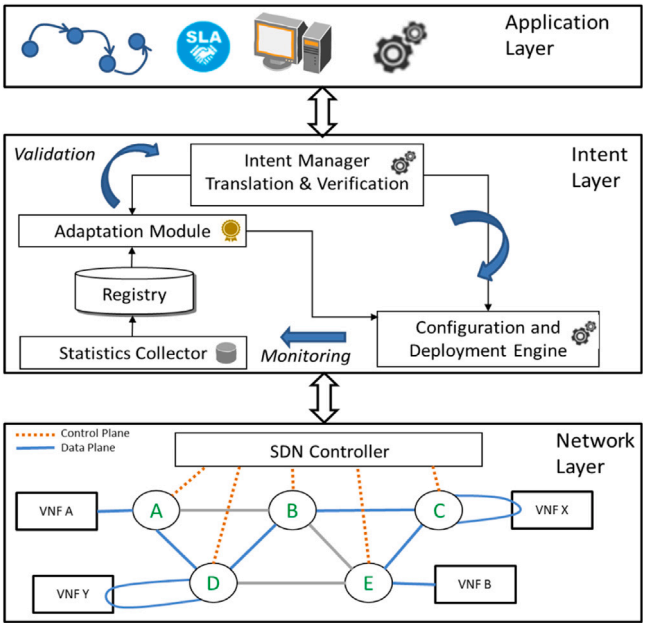

Fig. 2. Intent framework: Functional design and building blocks.

- *Adaptation Module*: continuously verifies the overall status of the network through the polling of OpenFlow statistics and supervising operations to maintain the performance of established intents (i.e., *validity verification*) in line with specified requirements, despite possible changes in the network status due to dynamic service requests (e.g., steady increase of traffic load). In case of intent validity failure (e.g., a switch congestion degrades data delivery throughput required for the service chain intent), it is responsible of triggering the *Intent Manager* for starting re-optimization procedures and remedial actions eventually leveraging the *Configuration and Deployment Engine* to implement them in the SDN network through, e.g., re-establishment or redirection of data delivery paths.
- *Statistics Collector*: interacts with the SDN controller to retrieve network statistics related to the switches composing the data plane. Different metrics can be considered, such as bandwidth, jitter, delay, or even more sophisticated and consolidated metrics to derive the switch status (e.g., average switches load). The statistics data are then made available to the *Adaptation Module* to decide if remediation or re-optimization actions are needed.
- *Registry*: contains service and operational data on network nodes and delivery paths. In particular, it contains a list of the available virtual functions instances with related information (e.g., type, network location in terms of IP prefix, Data Path Identifier and port of the switch they are connected to) and a list of the established data delivery (segment) paths with additional information (e.g., IP addresses of end-points, identifier and port number of intermediate network nodes) realizing (part of) running service chain instances.

As result of the interworking among the described components, the following phases underpin the overall Intent Layer operation:

- *Awareness*: consists in the collection of OpenFlow statistics (and related data processing) performed by the *Statistic Collector* through the SDN Controller NBI to derive the current network status. Based on network state awareness, remedial actions can be triggered as part of the assurance phase. In this work, we chose the switches load as a metric for the validation mechanism since it directly impacts the throughput and thus the Quality of Experience (QoE) of users/applications. It is also strictly related to other metrics since overloaded switches will start discarding packets (increase of packet loss) and network congestion will cause a delay increase. In Section 5.2 more details are given regarding the OpenFlow monitoring process and the switch load estimation supporting the Awareness phase.
- *Fulfilment & Verification*: consists in the set of actions performed by the *Intent Manager* to (i) derive the operations necessary to meaningfully configure the network according to the user's goals and requirements expressed in the Intent (*Translation*) and (ii) validate the compliance of the underlying network with the Intent requirements and hence verify if the Intent can be accepted based on the current network status (*Verification*). In this work, the intents are issued through *Intent Manager* NBI and expressed through a template from which the Intent Manager extracts the set of parameters necessary for the configuration of the service chain (e.g., source, destination, VNFs) and the verification check is performed based on the availability of network resources (e.g., intent throughput does not overload the switches according to a certain threshold on current switch load). If the verification is passed, the Intent is enforced in the network leveraging the *Configuration and Deployment Engine*. In Section 5.1 the Intent NBI is described along with the Intent template. In Section 5.3 further details are given on the actions performed by the *Intent Manager* and the steps to enforce the Intent into the network.

Table 1
REST API offered by the Intent Layer.

Intent type	HTTP method	Description
Simple	POST	Sets-up a path between the endpoints supplied in the request message.
	DELETE	Deletes the path.
Composite	POST	Sets-up a path between the endpoints supplied in the request message that traverses a given number of network functions.
	DELETE	Deletes the composite path.

- *Assurance & Validation*: consists in the set of actions for the continuous check if the intents statuses are fully compliant with the users requirements/goals originally expressed in the Intents. Indeed, due to the dynamic scenario of incoming intent requests with unknown arrival patterns (i.e., dynamic service requests), the resulting network status and resource allocations (e.g., established traffic flows) may change, even unexpectedly. Hence, the Intent Layer may have to deal with unpredictable situations that may compromise/degrade the performance of established intents. For this reason, reconfiguration actions may be required to maintain the performance of established intents despite those changes in the underlying network status. In this regard, different criteria (and correspondingly different performance indicators) can be considered to trigger reconfiguration actions (e.g., reliability, QoS/QoE, load balancing) corresponding to as many different resource management goals (e.g., minimize SLA violations, improve cost- and energy-efficiency, maximize request acceptance rate) while assuring the delivery of the expected level of performance with intents defined objectives. More specifically, the throughput metric can be used to trigger reconfigurations towards addressing reliability and load balancing (computed at node level) and QoS/QoE (computed at flow-level), the network latency metric between edge nodes can be consider to address QoS/QoE, notifications on failures can be considered to handle network reliability criteria. The overall process is governed by the Adaptation Module that continuously polls the collected network statistics and process and store performance metrics to trigger The Intent Manager to perform reconfiguration actions, if needed, regarding the execution of the Intent. In this work, we consider the load of a significant subset of switches as a relevant performance metric and the exceeding of a specified load threshold as a reconfiguration trigger according to a load balancing criteria as part of resource management operation policies. More specifically, upon the overload of one or more switches, the Adaptation Module triggers the Intent Manager to perform remediation/re-optimization actions that consist in the redirection of the paths traversing the overloaded switch(es) to other available switches in the network. In Section 5.4 additional details and workflows are given on how the Adaptation Module acts in cooperation with the Intent Manager in this phase.

5. Intent northbound interface and layer operations for service chaining

In this section, the characteristics of the intents for service chains are highlighted (5.1) and the description on how the three different phase operations are run in the proposed implementation of the Intent Layer for service chaining is detailed (5.2) to (5.4).

5.1. Intent northbound interface for service chaining

The interaction between the Application Layer and the Intent Layer is handled by a RESTful interface. REST is a resource-oriented architectural style for networked systems, which is considered a best practice

for building distributed hypermedia systems and web service interfaces. Indeed, REST is also widely adopted in the software-defined networking area.

In order to allow applications express their operation goals, two types of intents were defined that are differentiated according to the parameters expressed in the request. More specifically, *simple intents* only require the specification of the source and destination endpoints to setup a path with a specific throughput, while *composite intents* require the specification of the endpoints as well as the number and type of virtual functions that must be traversed by the traffic along the network path. Table 1 summarizes the set of operations exposed by the Intent Layer to the Application Layer.

In this regard, a template is considered that has to be filled out to specify the operation goals in terms of intents without taking care of the low-level implementation details. The template-based approach offers a high level abstraction on network resources and network behaviours and allows users to easily specify their goals, which will then be translated into a set of network configurations realizing the service chaining. New requirements can be simply added as new parameters in the template.

```
<Intent Id="IntentId">
    <Parameters>
        <Source_IP="10.0.3.1"/>
        <Destination_IP="10.0.11.1"/>
        <Chain>
            <Length=2/>
            <Node_Id="Firewall"/>
            <Order=1/>
            <Node_Id="DPI"/>
            <Order=2/>
        <Chain/>
        <Throughput="5"/>
    </Parameters>
</Intent>
```

Listing 1: Example of an intent template

Once the request is correctly translated by the *Intent Manager*, the parameters for its fulfilment are extracted and sent to the *Configuration and Deployment Engine* which constructs a JavaScript Object Notation (JSON) based message that is sent to the Network Layer.

The extract presented in Listing. 1 shows an example of the template adopted in this work for a composite path setup requesting that all the traffic originating at a node with IP address 10.0.3.1 and terminating at a destination node with IP address 10.0.11.1 should pass through a network function chain made by a firewall and a DPI. The *Throughput* parameter set to 5 specifies the amount of traffic that must be guaranteed between the two endpoints during the whole service duration.

5.2. Awareness: SDN-based network monitoring and load estimation

A background monitoring of the network nodes is necessary to (i) obtain operational data related to the actual utilization of the switches and/or links, and (ii) adequately support the *Configuration and Deployment Engine* and *Adaptation Module* operations based on the current operational status of network nodes. Within SDN, the OpenFlow (OF) protocol allows for the live-manipulation of the flow tables of OF-enabled switches as well as the retrieval of real-time data statistics for monitoring purpose. In particular, in this work OF statistics relative to the traffic load from the switches in the network are periodically collected: (i) per-flow received/transmitted packets/bytes, (ii) flow duration, and (iii) per-port received/transmitted packets/bytes/errors [43]. Then the per-port bytes counters of each switch are used to assess

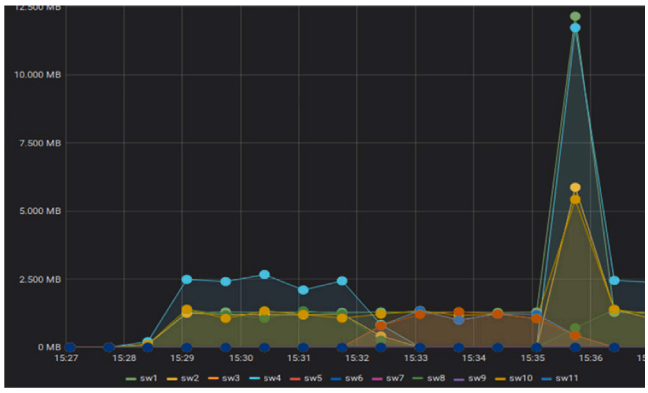

Fig. 3. Screenshot of the Grafana system — Status of switches load.

its load and its capability to support the required packet processing and exchanges. The difference of received/transmitted bytes between 2 consecutive queries is computed and consequently the traffic rate at the switch ports by dividing by the duration of the polling time. Based on this information, consolidated trends of traffic load at switches ports are derived [44]. More specifically, for each switch s, at the port connected to the link l, 2 consecutive counter values for transmitted and received bytes (i.e., *Tx_Bytes* or *Rx_Bytes*) at t_{i-1} and t_i are collected, then the traffic rate at the link l is calculated as follows:

$$Link\_Load\_In_i(s,l) = \frac{[Rx\_Bytes(s,l,t_i) - Rx\_Bytes(s,l,t_{i-1})] * 8}{(t_i - t_{i-1})}; \quad i = 1,2,\dots,n \tag{1}$$

$$Link\_Load\_Out_i(s,l) = \frac{[Tx\_Bytes(s,l,t_i) - Tx\_Bytes(s,l,t_{i-1})] * 8}{(t_i - t_{i-1})}; \quad i = 1,2,\dots,n \tag{2}$$

The *Statistics Collector* then processes the retrieved statistics to obtain the operational status of the switches as follows:

$$Switch\_Load_i(s) = \sum_l Link\_Load\_In_i(s,l) + \sum_l Link\_Load\_Out_i(s,l) \tag{3}$$

The collected statistics can be visualized using *Grafana*, an open source graphical representation tool [45]. In the example shown in Fig. 3 a network topology composed of 11 switches and a situation where already few intents have been processed and enforced by framework is considered.

For each established intent, 1MB of User Datagram Protocol (UDP) traffic is sent between the two endpoints composing the chain. In the screenshot, the status of the switches in terms of average load is displayed. It can be noticed that at a given time, not all the switches have the same load and that their status dynamically changes over the time. More specifically, if the switches load threshold is for example fixed to 5MB, it cen be clearly observed that from 15:27 to 15:35 all the switches loads are below the threshold. Few seconds after 15:35, the load of switches $2, 4, 8$, and 10 suddenly increases due to the setup of new incoming intents and the flow of their related traffic. Since the load of those switches exceeds the threshold, a remediation action is triggered, i.e., a path redirection mechanism involving less loaded switches. From the figure, it can be clearly observed that, after the redirection is completed, the load of those switches drops below the threshold thus performing a better distribution of the traffic over all the switches available in the network.

5.3. Fulfilment and intent verification

In this subsection, details on how the Intent Layer components contribute in the fulfilment of intents are given. The case of *composite intents* is considered.

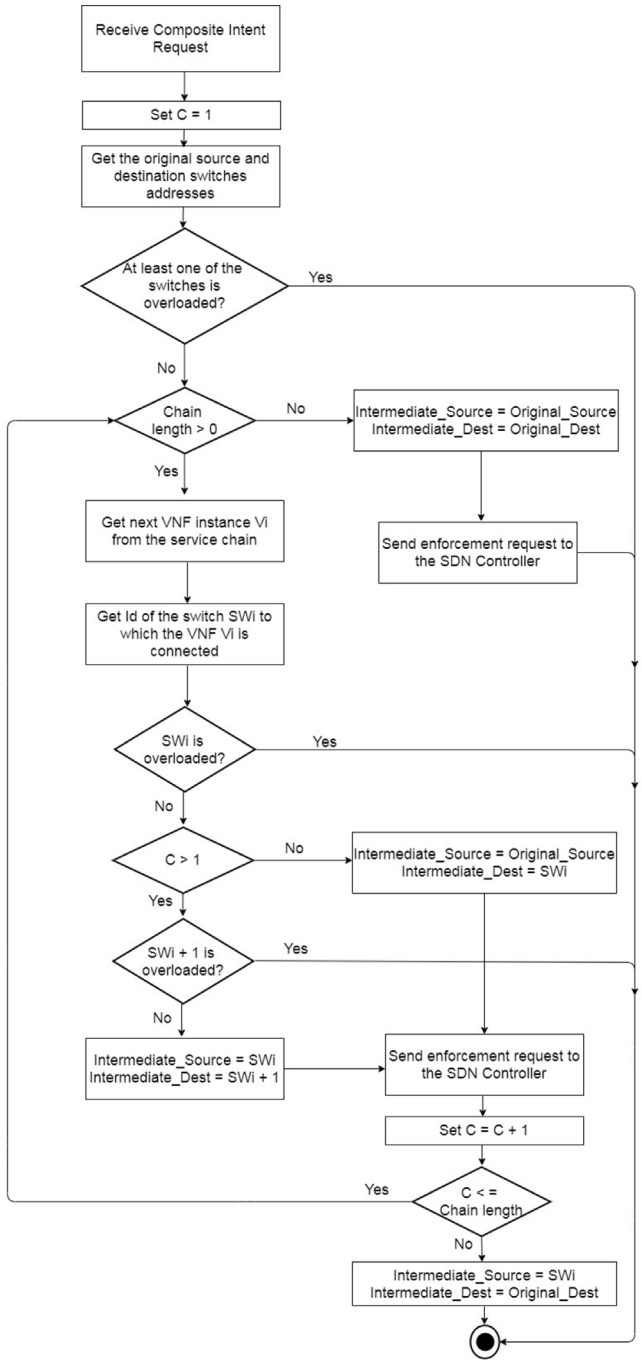

Fig. 4. Flowchart of the fulfilment phase.

In addition to the *Intent Manager*, the fulfilment phase mainly involves the *Configuration and Deployment Engine*, the *Statistics Collector* and the SDN controller to enforce the intent in the network after the verification that the intent can be supported. More specifically, as shown in the flowchart depicted in Fig. 4, the fulfilment operates as follows.

After the translation of the template and the extraction of the intent's parameters, the Intent Manager verifies the feasibility of the deployment of the intent according to the current network status (Verification). It starts by retrieving the status of the switches directly connected to the endpoints and capable of satisfying the request requirements (i.e., source and destination endpoints, virtual functions to be traversed), called "original source" and "original destination" switches in the flowchart, and checks their current load. If at least one of those switches is overloaded, the request is rejected.

Otherwise, it forwards the request to the *Configuration and Deployment Engine*, which for all the VNFs composing the service chain, interacts with the SDN controller to select the appropriate switches for the request and then splits the chain into a set of segments where each segment is composed of an "intermediate source switch" directly connected to one VNF and an "intermediate destination switch" connected to the successive VNF in the chain. The switches composing each segment are then also checked for their current load through several interactions between the *Intent Manager* and the *Configuration and Deployment Engine*. If at least one of switches is overloaded, the request is rejected. Otherwise the intent is enforced in the network by setting the necessary flow rules on the set of eligible switches along the path.

5.4. Assurance and intent validation

In this subsection, details are given on how the Intent Layer components contribute in automatically validating (i.e., zero-touch management) each enforced intent to maintain its status during its whole life-time in accordance with the expressed goals. The case of *composite intent* is considered.

In addition to the *Intent Manager,* the assurance phase mainly involves the *Adaptation Module* and the *Statistics Collector* to decide if actions are needed to maintain the intent despite data rate degradation or delay increasing. Indeed, after its fulfilment, the intent can deviate from the desired outcome due to dynamic network conditions, e.g., increase in the switches load. Therefore intent status is continuously monitored to guarantee its agreed parameters while responding to the continuous changes of the network. This is part of the *validity verification* operation running in background to assure the intents remain in line with the outcomes expressed by the application (i.e., zero-touch and fully automated operation).

More specifically, the validation operates as follows. Thanks to the monitoring process performed in background, the *Adaptation Module* periodically checks the operational data of all the switches and retrieves the list of currently active paths. For each switch that is connected to a cloud platform, if the average load is higher than a given threshold, the switch is considered as overloaded. The *Adaptation Module* triggers then the *Intent Manager* to perform corrective actions regarding the execution of the intent. For this purpose, once a congestion notification is received, the *Intent Manager* interacts with the *Coordination and Deployment Engine* to trigger the redirection of the active data delivery paths passing through the overloaded switch to other switches available (i.e., not overloaded) in the data plane.

In Fig. 5 a flowchart that summarizes the different steps executed during the validation process is presented. More specifically, the *Adaptation Module* periodically checks the collected statistics, which are relative to the status of the switches present in the network. The *Adaptation Module* examines the switches one by one (i.e., "More switches?") and elaborates their statistics to get the overall load of each switch. If the average load of one or more switches in higher than a given threshold, it triggers the *Intent Manager* to perform remediation actions. The *Intent Manager*, checks again the eligibility of the intent, and if the new deployment is feasible triggers the *Coordination and Deployment Engine* to enforce the new network configuration. This latter gets the list of flow entries installed in the switch and deletes them one by one (i.e., "More flow entries?"). Then, from the list of deleted paths, the source and destination IP addresses of each path are obtained and all the traffic (i.e., "More paths?") is redirected to other unloaded switches in the network where the flow rules are re-installed.

It is worth pointing out that the *execution verification* may result in the need to block some intent requests from the applications because of lack of resources. On the other hand, the *validity verification* aims at maintaining established intents despite dynamic network status

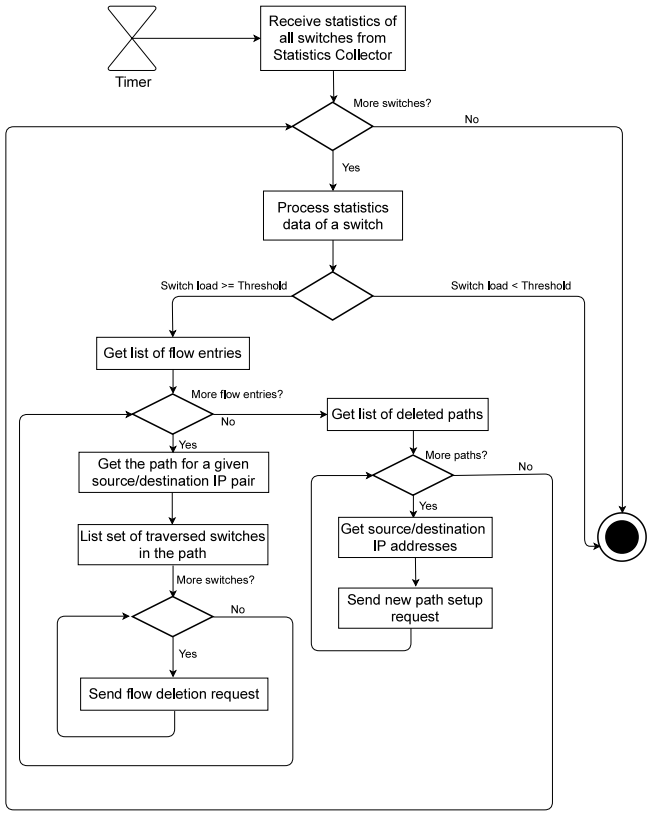

Fig. 5. Flowchart of the validation phase.

(e.g., increasing network load). Hence, intent verification and validation are also resource management operations since overall resulting in and taking advantages from a better resource utilization. Indeed, intent processing combined with effective resource management and control capabilities are beneficial not only to fulfil at best the incoming intent requests but also to increase the possibility to fulfil upcoming/future intent requests while assuring the running ones.

6. Performance evaluation

In this section, a set of results obtained after carrying out a number of experiments is presented to explore the performance of the presented automated deployment and show its effectiveness.

6.1. Emulation settings

In order to evaluate the performance of the proposed framework, a virtualized environment is adopted where a prototype of the proposed Intent Layer is implemented. The Intent Layer components are executed and configured to interact with a Network Layer emulated using Mininet, a network emulator that allows to deploy a SDN-based network [46]. As a reference network, the Abilene topology is used [47] which is a high-performance backbone network created by the Internet2 community and composed of 11 nodes connected in a mesh topology. In each node of the topology, an OpenFlow-enabled emulated switch is deployed. Mininet typically uses the default Linux bridge or Open vSwitch running in kernel mode to switch packets across interfaces. As a result, the OpenFlow switches created by Mininet provide the same packet delivery semantic that would be provided by a hardware switch. A subset of these switches are connected to emulated cloud platforms that contain the virtual functions while the remaining switches are simply forwarding switches. Moreover, a real SDN controller, Floodlight [48], is used for the setup of the service chains paths and traffic steering. In particular, for the selection of the appropriate switch, the dijkstra's shortest-path algorithm provided by the SDN controller is exploited to select the closest instance for each service specified in the chain. In this algorithm, the switches are associated with no weight. Moreover, all the nodes in the topology are randomly chosen to behave as a source/destination of the service chain requests. If not specified otherwise, the VNFs performing the same type of network functions are assumed to be present in each cloud platform, which releases us from the use of any placement algorithm since the focus of this paper is on the service chains establishment through intent-based operations. In addition, to emulate the traffic flowing in the network, a UDP traffic generated from the Iperf tool [49] is sent from each source host to each destination host specified in the requests. The amount of traffic sent is equal to the throughput value expressed in the intent and randomly varies within the interval $[1\ \text{Mbps}, 10\ \text{Mbps}]$.

In the following, two types of tests were conducted. First, in Sections 6.2 and 6.3, to demonstrate the validity and feasibility of the approach, a simple API that takes the intent template as input and sends it to the Intent Manager was implemented. After the fulfilment of the intent (setup of the service chain path in the network), the Iperf tool was used to generate UDP traffic equal to the amount specified in the intent request. The requests follow a deterministic Inter-arrival Time (IAT) equal to 10 s. In Section 6.4, the performance of the Intent Layer verification and validation mechanisms is assessed focusing more on the network load and the behaviour of the Adaptation module. For this purpose, a more dynamic pattern for the intent requests is considered (the requests follow a Poisson distribution characterized by an inter-arrival and holding times exponentially distributed with an average of $1/\lambda$ equal to 20 s and $1/\mu$ varying within the range $[5\ \text{s}, 60\ \text{s}]$ respectively). Also during these tests, UDP traffic was generated using the Iperf tool. Results were collected with a confidence interval of 5% at a confidence level of 95%.

6.2. Performance of intent enforcement in edge cloud environment

In this paragraph, the SDN Network Layer performance is evaluated while enforcing composite service chain intents and when different pervasiveness levels of VNFs in the network clouds are adopted. For this reason, the verification operation is deactivated and the validation mechanism is not adopted, and only the intent enforcement in the network nodes is considered which is the operation that is mainly affected by network and cloud deployment set-up. The number of requested virtual functions in a chain is fixed to 3, which is a reasonable number (not too low and not too high [50,51]). This is only a design choice and not a limitation of the prototype, which is able to consider chains with different lengths. The number of switches connected to a cloud platform is also varied from 3 to 10 out of 11 available switches. Each cloud platform contains a fixed number of virtual functions equal to 3.

In Fig. 6 the flow setup time is evaluated, defined as the total time required to setup the flow entries in all the switches across the service chain paths. Results show that the setup time is quite independent from the number of switches connected to the cloud platforms. The small variations are due to the total path length (i.e., in terms of number of segments) which might differ from one request to the other and which in some cases necessitates traversing other switches (i.e., forwarding switches) thus increasing the setup time.

Fig. 7 plots the average number of flow entries installed in the switches as a function of the number of switches connected to cloud platforms. Results show that the number of flow entries considerably decreases by increasing the number of switches connected to cloud platforms, which can be explained as follows. Data delivery path requests are characterized by a source, a destination and a chain of virtual functions that are generated randomly. Identifying a feasible path results in identifying a shortest path that traverses the set of switches connected to the type of VNF specified in the request. Since

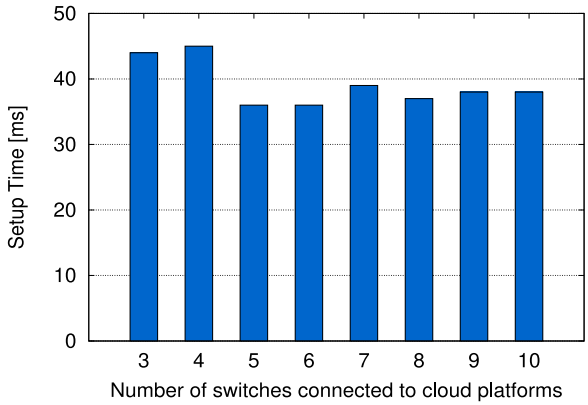

Fig. 6. Flow setup time vs. number of switches connected to cloud platforms.

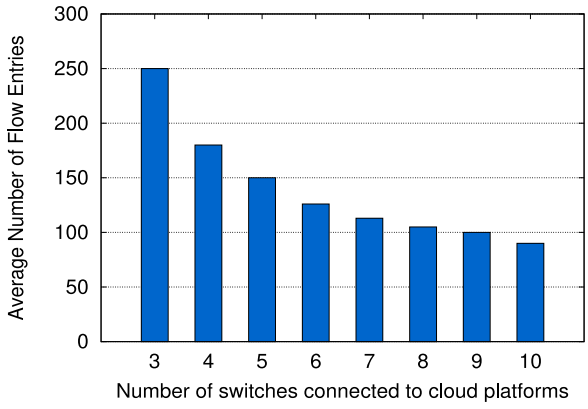

Fig. 7. Number of flow entries vs. number of switches connected to the cloud platforms.

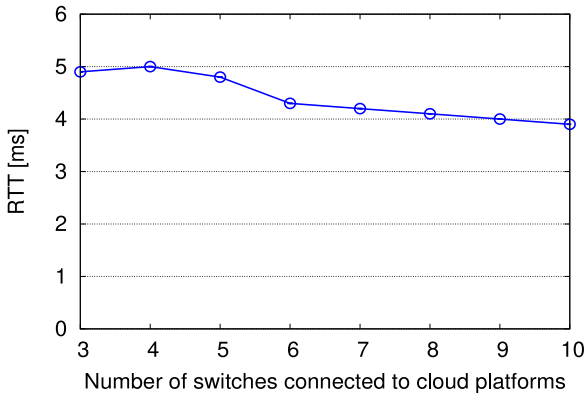

Fig. 8. RTT for a different number of switches connected to the cloud platforms.

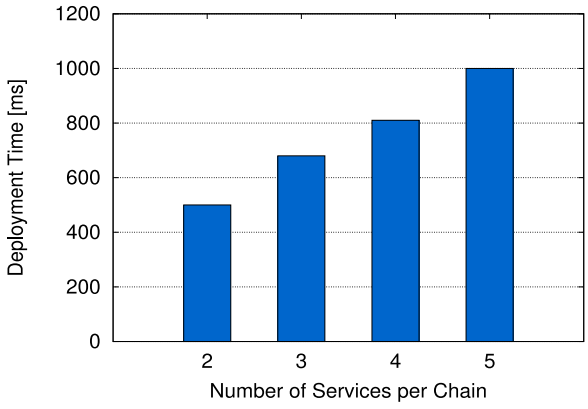

Fig. 9. Intent deployment time vs. chain's length.

that the cloud platforms connected to the switches are assumed to contain all the possible types of VNFs, increasing the number of available cloud platforms and spreading them on different switches decreases the number of paths traversing the same switch for a given VNF, which alleviates the load (i.e., in terms of number of flow entries) on the switches.

Fig. 8 plots the Round Trip Time (RTT) as a function of the number of switches connected to cloud platforms. The RTT refers to the time required for a packet to travel from a specific source to a specific destination and then gets back again. To do so, the Ping tool was used to measure the RTT between each source/destination couple of every intent request successfully fulfilled and then the average value was calculated. Results show that increasing the number of switches connected to the cloud platforms slightly decreases the RTT. Such behaviour is typical of edge networks where the service is brought closer to the customers through the proliferation of cloud platforms hosting the VNFs. In fact, given that the shortest path algorithm has been used for the selection of the VNFs, having larger number of switches with cloud platforms provides a higher number of alternative paths. However, the gain is limited (1 ms in the best case), which implies that the deployment of a relatively small number of switches connected to the cloud platforms might be acceptable thanks to VNFs replications.

This result shows the importance of a trade-off between the number of the edge clouds platforms deployed in the network and the QoE expressed in terms of end-to-end delay experienced by users requests. Augmented Reality (AR) applications for example in different fields such as healthcare, entertainment and education require low latency, which can be brought down by reducing the number of hops. On the other hand, the deployment cost of the edge cloud platforms and the energy efficiency aspects such as their power consumption are also important parameters to be taken into consideration.

6.3. Performance of intent-based service layer

Secondly, the performance of the Intent Layer considering different numbers of VNFs in the chain (i.e., chain length) was evaluated. To this purpose, the number of switches connected to cloud platforms was fixed to 5 and the chain length varied from 2 to 5. Again, the verification operation is deactivated and the validation mechanism is not adopted.

In Fig. 9 the overall intent deployment time is plotted as a function of the service chain's length. The deployment time refers to the time elapsed between receiving the intent request from the Application Layer and its complete fulfilment at the Network Layer. It mainly consists into two components: (i) the intent processing time, which includes the translation time, the time necessary for checking the database, the time necessary for the communication between the *Configuration and Deployment Engine* and the controller, etc.; and (ii) the path setup time, which is mainly relative to the flow rules installation time. The experiment was conducted as follows. Several intents with different service chains length (going from 2 to 5) and different sources/destinations values were sent. Each time an intent request was sent, a timer was triggered to start measuring its deployment time. After the fulfilment of the intent (setup of the service chain path in the network), the timer was stopped. In this plot, only the time spent by the Intent Manager to reply for successful intents is considered. Results show that by considering more services in one request, the deployment time increases mainly for the increase of the number of switches to be traversed that consequently necessitate more interactions and more flow entries to be installed.

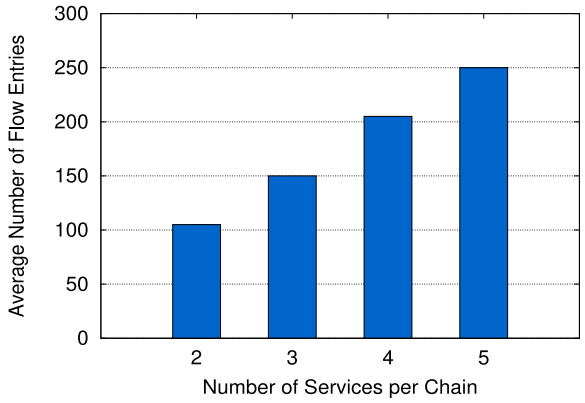

Fig. 10. Number of flow entries vs. chain's length.

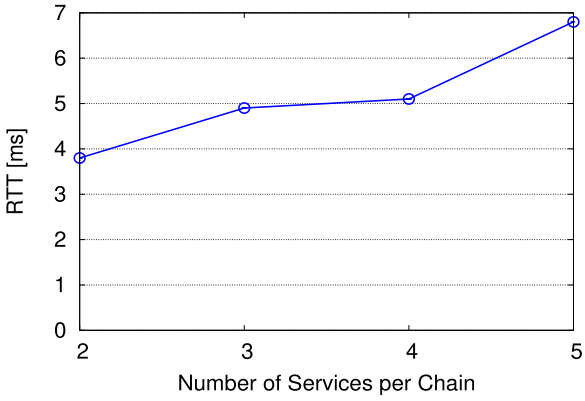

Fig. 11. RTT vs. chain's length.

The plot in Fig. 10 shows that the number of flow entries installed in the switches increases as the number of required services in a chain increases. In fact, since the number of switches connected to cloud platforms is limited to five, increasing the number of virtual functions in one request also increases the number of traversed switches (from 2 to 5) which requires the setup of more flow entries to forward the traffic along the data delivery path.

Fig. 11 plots the average RTT as a function of the different lengths of the service chains considered in all the requests. The VNFs are assumed to be transparent (i.e., no packet modification is performed) and the VNFs performing the same type of network functions are present in each cloud platform, which releases us from the use of any placement algorithm. Once the intent request was successfully fulfilled, the ping tool was used to measure the RTT from each source to each destination specified in the intents, and the obtained average values were plotted. As expected, it can be observed that the RTT experienced by the packets increases as the number of services in the chain increases. In fact, when having more services in the chain, the number of traversed switches that need to be configured (i.e., setup of flow entries) increases which raises the RTT.

6.4. Performance of the intent layer verification & validation

After focusing on the performance of intent processing and enforcement operations, in this paragraph the impact of the verification and validation mechanisms on the rate of fulfilled intents is assessed in relation to the effectiveness of resource utilization in the SDN edge network.The impact of the verification and validation mechanisms is also assessed in terms of further processing load that they required to be implemented. More specifically, the blocking probability (BP) is

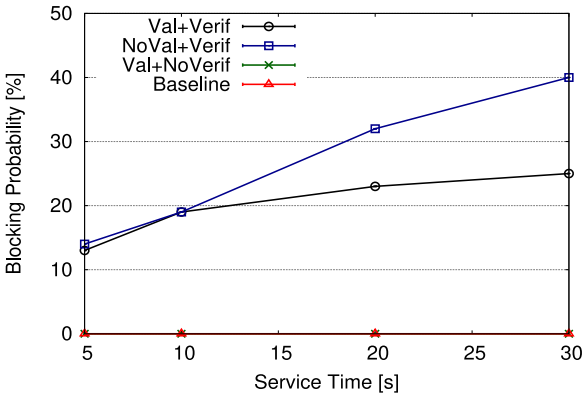

Fig. 12. Blocking probability.

calculated as the probability that one request is not accepted by the *Intent Manager* since the network in its current state cannot support the execution of the intent. In this work, the load status of the switches was considered as a metric for the network congestion. A switch is considered overloaded if its average load is higher than a given threshold. We also provide the average switches load and the overhead on the Intent Layer expressed in terms of exchanged messages and number of performed redirections. The results are compared with the basic Intent Layer operation where no verification and validation mechanisms are adopted (i.e., only intent translation and enforcement are performed).

For each test, a sequence of 100 composite intent requests has been generated. Each request is characterized by a random source, a random destination and 3 virtual functions. The number of switches connected to a cloud platform is fixed to 5. The requests follow a Poisson distribution characterized by an inter-arrival and holding times exponentially distributed with an average of $1/\lambda$ and $1/\mu$, respectively. $1/\lambda$ is fixed to 20 s while $1/\mu$ varies within the range [5 s, 60 s], thus achieving the desired load computed as λ/μ and expressed in Erlang. The holding time corresponds to the time necessary for maintaining active a flow. Also for these tests, a UDP traffic is generated using the Iperf tool and sent from each source host to each destination host specified in the requests. The amount of traffic sent is equal to the throughput value expressed in the intent. The switches average load threshold is fixed to 20 MB.

Fig. 12 plots the blocking probability and shows that, when the verification feature is considered, at low loads, BP is low since the switches are not overloaded, which minimizes the risk for an intent to be rejected independently from the adopted mechanism. By increasing the load, BP increases in the no validation case while it remains almost stable but lower when the validation mechanism is adopted. In fact, once the average load of a switch reaches the threshold, the already installed flows are deleted and redirected to less congested switches. At the arrival of a new request, the current load is already balanced on all the available switches which decreases its probability to be rejected thus allowing to accommodate more intents.

In Fig. 13 the average service time is fixed to 20 s and the average load of the switches connected to cloud platforms is plotted. Results show that with respect to the baseline (no verification and no validation are performed), the introduction of at least one of the two mechanisms enhances the performance. In fact, in the baseline high disparities can be noticed among the switches in terms of average load where, for example, switches 2 and 4 are almost doubly loaded with respect to switches 6 and 10. Moreover, the adoption of the path redirection or the blocking features separately introduces an improvement in the results with respect to the baseline. This is confirmed by the load balancing of the current traffic among all the available switches. The path redirection presents slightly better results since it is performed periodically and does not depend on the arrival rate of the requests. Finally, as

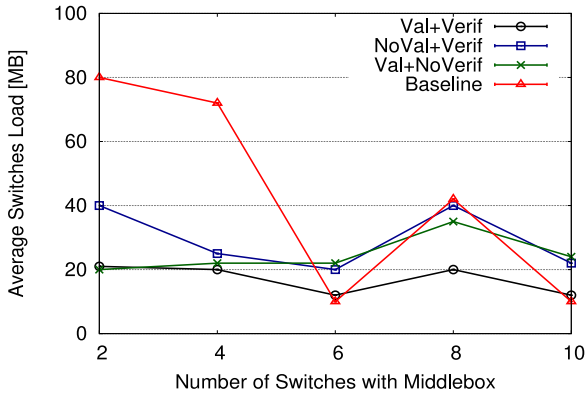

Fig. 13. Average Load in the switches connected to cloud platforms.

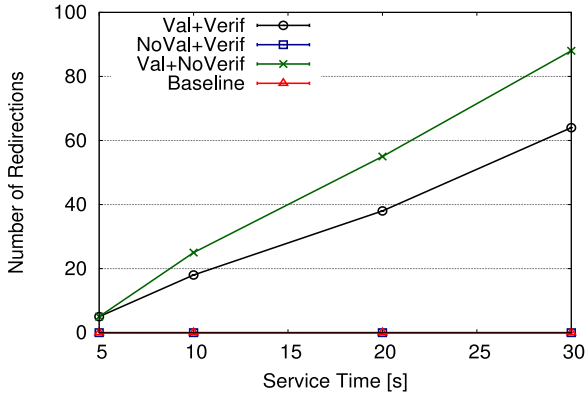

Fig. 14. Number of redirections for different loads.

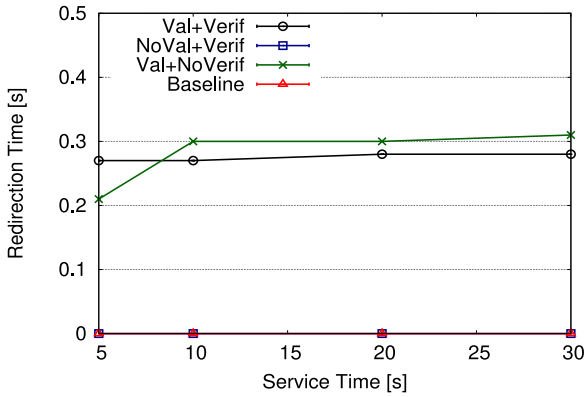

Fig. 15. Redirection time.

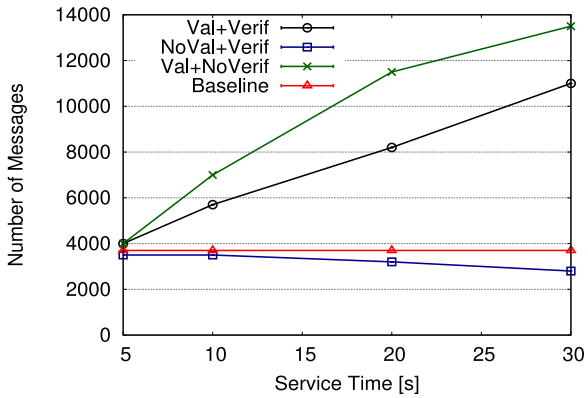

Fig. 16. Number of messages exchanged between the controller components.

expected, the combination of the two features further enhances the performance which results in an almost equal distribution of the load among the switches.

Load balancing is an important design goal especially for multimedia applications such as live broadcast, video on demand (VoD) and real-time conferencing. In fact, those applications have high requirements in terms of reliability and real-time performance which can be addressed through load balancing since it allows to maintain a high throughput while minimizing the packet loss due to the overload of one or multiple switches in the network.

In the following graphs, the overhead on the Intent Layer of the verification and validation mechanisms is evaluated. In Fig. 14, the number of performed redirections is plotted, which increases linearly as the network load increases. In fact, at high loads more switches become overloaded more frequently which, as expected, increases the number of the redirections in order to balance the load and then limit the switches congestions. Moreover, results show that the number of redirections is lower when the verification mechanism is adopted. In fact, in such a case the overhead is minimized since no more requests are accepted when the network is congested.

Fig. 15 plots the redirection time as a function of the network load. Results show that the time necessary to the controller for deleting a path traversing an overloaded switch and setting it up through a different sequence of switches is independent from the current load. The redirection time mainly depends on the characteristics of the environment in which the SDN controller is running and the network topology is emulated (e.g., CPU and RAM of the virtual machines hosting them).

Fig. 16 plots the number of messages exchanged between the different components of the Intent Layer and the SDN controller. When the validation mechanism is not adopted, the number of messages is constant and is relative to the setup/deletion of the flows which corresponds to the number of data delivery requests. On the contrary, as expected, the redirection feature increases the number of messages exchanged between the Intent Layer and the Network Layer since every time a redirection is performed, a new path is calculated and new flow entries are set-up which necessitates further communication messages (e.g., path delivery set-up, path tear-down, access to the database, etc.). It is worth highlighting that the overhead is higher when the verification is not performed since all the requests are accepted independently from the current network load which increases the congestion and then the number of redirections.

7. State of the art

Several works in literature show that the adoption of SDN, possibly in combination with an orchestration layer, provides increasing flexibility and scalability to dynamic service chaining [52]. Moreover, the emerging IBN concept combined with software-defined networks pave the way for the automation and efficiency of edge computing scenarios. In this section, the above concepts are discussed in three related research areas. Then, the contribution of this work is highlighted as a result of the reported analysis of related works.

7.1. SDN-based service chaining

The use of SDN in service function chaining is considered as a promising technology to programmatically enforce data delivery paths in the network and thus to dynamically steer application/service data flows across a set of service functions/virtual functions taking part in the service chains [52]. In this area of investigation, [53] proposes the NIMBLE system that uses SDN-based traffic steering operations for

middleboxes management and compositions. The key idea is based on the establishment of tunnels between all switches pairs in the network, which significantly reduces the number of flow entries and allows to simply forward packets among the different middleboxes present in the network. Similar to what we do in this work the middleboxes deployment is based on a software architecture for configurable routers (Click modular router) while relying on cloud providers or middleboxes placed in virtual machines VMs is considered as a possible extension of the work. [54] proposes the StEERing framework for dynamically steering data flows across a specified chain of middleboxes while leveraging SDN and the OpenFlow protocol. The framework supports efficient forwarding for a large number of applications and subscribers. As proposed in this work, in their simulations, authors adopt the Abilene topology and distinguish between switches connected to services and simple forwarding switches, while the experimental testbed is quite limited since composed of 4 middleboxes and 4 OpenFlow switches. [55] presents a prototype of an SDN-enabled node that, given a new user device connected to one of its physical ports, is able to dynamically instantiate the user-specific forwarding graph by forcing the user data to traverse the required set of network functions over a user-specific path set-up across instances of programmable OpenFlow switches (i.e., Logical Switch Instances). Upon the arrival of a new network function forwarding graph, an LSI is activated on-the-fly to steer traffic among the required VNFs deployed either as DPDK processes or in docker containers, in contrast with our approach where VNFs are already deployed in the emulated cloud platforms. Traffic isolation features for service chaining are also considered in [56] where a novel approach and a prototype is presented to integrate network services into a service chaining system. The network isolation of service instances allows a simplified deployment process. As in this work, during the proof-of-concept, the service chaining relied on Mininet and custom scripts for creating the network and the service instances. However, unlike in this proposal where the SFC deployment is fully automated, in [56] middleboxes still require a minimal configuration which is costly and error-prone. On the other hand, the orchestration of network and cloud-based NFV resources is a relevant research areas, especially in combination with SDN control plane solutions [57]. Indeed, a coordinated control of both cloud- and network-layer resources is recommended in order to provide adaptive service data delivery and adequate user service experiences [58]. In our previous works, concatenated connectivity in computing networks (i.e., into data-center/cloud networks) was also investigated [59] and we also contributed to an end-to-end service chaining including connectivity in edge clouds [60] or within network slicing for industry 4.0 [61]. Latency, adaptability and availability requirements were also investigated in [62]. In this area of research, [63] proposes an orchestrator for resilient service function chaining operations in cloud networks able to recover from service function failures by activating a stand-by VNF instance and, accordingly, by rerouting the network path of the chain to include the new instance. In [64] authors present a SDN controller design to address the service chain orchestration aiming at dynamically handling multiple data flows, at differentiating traversed network functions (i.e., forwarding graphs) to address different user SLA, and at steering the data traffic throughout nodes, accordingly. They also used a monitoring system based on packet inspection to verify the meeting of the data flow to the SLA. In [65] authors present an SDN orchestrator for dynamic chaining of computing and communication resources while assuring better than best effort data delivery among virtual functions (i.e., VMs) in Cloud Data Centers. Moreover, a set of chaining strategies are proposed that consider the current load of switches and servers for resource selections. [38] authors present a distributed service chaining approach that coordinates among the VNFs selection, placement and routing operations. The simulated network relied on a common data center topology where the focus is mainly on an efficient VNF instances selection to improve intra-cloud resources utilization and minimize links load ratio.

7.2. Software-defined edge computing

The proliferation of edge devices and the increasing requirements in terms of minimization of service delay and energy consumption has participated in the popularity of Edge Computing, which allows for bringing computational infrastructures and thus services closer to the end users. However, along with its benefits, edge computing brings also some technical challenges and complexities that can be faced through the adoption of SDN [66].

More specifically, SDN can help in improving the performance experienced by the users of edge computing systems by exploiting the OpenFlow capabilities to monitor service utilization and manage the load on edge servers. In [67] authors propose an edge computing framework that allows for dynamic route calculation to suitable edge servers for real-time vehicle monitoring. The framework is based on the integration of EC and SDN and the obtained results show its efficiency in terms of resources utility and delay. In [68] authors build a software-defined vehicular edge networking structure that achieves load-balancing by the virtue of the global load status information. The proposed algorithm allows to meet the requirements of computation and transmission delay for various vehicular tasks.

SDN also allows to flexibly handle users handover to avoid any performance disruptions when an edge device moves from a cloudlet to the other. Such operation is performed through dynamic flows redirection [69] or VMs live migration [70]. On the other hand, when coupled with edge computing, SDN helps in providing an efficient IoT service orchestration and contributes in solving the challenges related to complex IoT management [71]. In [72], authors present a framework for IoT that employs an edge computing layer of Fog nodes controlled and managed by an SDN network. The proposed framework relies on distributed SDN controllers and OpenFlow switches that in concordance with a load balancing algorithm achieves higher efficiency in terms of latency and resource utilization. In [73] authors exploit the benefits of SDN to support the use of large-scale fog-enabled vehicular network services by investigating the design principles and evaluating a traffic accident rescue use case. The obtained results clearly show how SDN contributes in easing the fog/cloud network integration and improves user experiences and the efficiency of vehicular communications.

However, despite all the research work performed within the software-defined edge computing area, there is only few works that tackle the hiddenness of all the network configuration complexities from the end users, while further simplifying their access to the edge computing services as we do in this work through the implementation of the Intent Layer. Among those, [11] presents an intent-based network control framework designed for data dissemination in the vehicular edge computing environment. In their work, authors use the Open Network Operating System (ONOS) controller along with an AI-based optimization engine and confirm how the use of the IBN approach allows for a significant preservation of the desired QoS requirements (i.e., energy efficiency, latency reduction).

Finally, it is worth pointing out that generic cloud data centers are not pretty in scope of this paper since they are more focused on cross-edge service function chaining in Edge Computing scenarios where routing (and network control in general) play a more key role to justify the use of SDN [38]. Indeed, in this work, a distributed environment of services interconnected with a generic mesh network topology is considered that provides multiple options for network connections supporting service chain paths.

7.3. Intent-based networking

The use of IBN techniques in SDN environments has been lately investigated in numerous works. More specifically, [19] presents an IBN approach for service routing and QoS control on the Koren-SDI infrastructure. Authors demonstrate the efficiency of IBN-driven routing in ensuring service provisioning and guaranteeing the seamless fulfilment

of QoS demands by changing traffic routes dynamically through a Machine Learning (ML) based approach used to select the best paths. The presented solution is based on the prediction of future link utilization while in this work the current network status is considered as part of the verification operation of the intent before processing it. In [21], authors propose Alviu, an SD-WAN network orchestrator, which allows for a dynamic intent-based configuration of Wide Area Networks (WAN) that assures end-to-end network slicing. The proposed orchestration solution is based on an SDN controller which abstracts the network equipments to be managed and facilitates their configuration. Although an IBN approach is adopted, the network characterization is declared through a set of descriptors, in contrast with the simplified template-based approach presented in this work. Moreover, no intent validation mechanism is adopted to ensure the automation and self-assurance of the intent-based platform as proposed in this paper. In [20] an intent-based NBI for end-to-end service management and orchestration across multiple technological domains is proposed. The general approach has been tested in an SDN-based testbed to show the benefits taken from a fully automated infrastructure. In [74], an intent-based NBI is designed to enable cross-domain end-to-end service management. The particular use case of an IoT infrastructure deployment has been considered for the experiment validation of the NBI. Finally, the Intent Framework offered byONOS allows to easily express the connectivity requirements in an SDN network through the specification of endpoints, constraints and actions [37]. Although simple, the framework still requires technical skills for the network configuration.

Other research initiatives regard the definition of descriptive interfaces, which is an essential aspect of IBN since it allows the accurate definition of services characteristics and network-related requirements to enable automated service chaining deployment and optimization [75]. To this purpose, an intent-based approach has been considered in [76] where authors claim the necessity to consolidate infrastructure-independent abstractions for specifying network functionalities and use an efficient and scalable virtualization platform to translate abstract specifications into concrete infrastructure configuration primitives. [77] presents a prototype that provides an intent-based service request API that allow the description of services in a more declarative manner (i.e., in terms of intentions and strategies). Finally, [78] proposes a NBI for intent declaration based on a behaviour-driven approach. Intents are specified in plain text and then translated into pre-compiled network policies before converting them into low-level rules by an SDN controller, which makes network configuration easier and paves the way for a simple and more expressive interface for intent-driven networking.

Few research works tackled the adoption of IBN in SFC contexts. In [22], a framework called OpenPATH is developed for the deployment of NFV applications with a special focus on traffic steering among NF service chains. The OpenPATH implementation meets some requirements of intent-based networking (e.g., an intuitive API for representing SFC policies,automatic deployment of the SFCs). However, unlike this work which focuses on the intent layer, the main focus of OpenPATH is on SFC placement and packet forwarding while some aspects of the IBN lifecycle are not addressed such as intents verification and assurance phases. [23] presents an intent conflict resolution scheme to avoid conflicts among intents declared by multiple users. As in this work, the intent template is simply composed of a source, destination, an SFC and a priority constraint. However, authors only deal with the validity and reliability of the conflict resolution scheme (intent verification) leaving intent execution and assurance details to future works. In [14] a proof-of-concept implementation of an intent-based northbound interface for VNFs dynamic service chaining is presented. Also in this work, the NBI takes as input a template including the source, destination and VNFs list in the service chain and then forwards the intent request to a Virtual Infrastructure Manager (VIM), which interacts with an SDN controller to steer traffic among the VNFs. The framework also allows for remediation actions through the update of the SFCs. However, the update can only be performed upon a specific request while in this work we handle congestion issues automatically as part of the lifecycle management of the intent.

Finally, the functionalities of the approach proposed in this work are in line with the current trends and initiatives in both dynamic service chaining and IBN carried out by some standardization bodies. The IETF is promoting the adoption of an intent-based interface to enable applications to easily describe their requirements for network connectivity to the network management systems [79]. Moreover, the IETF is defining a set of control functionalities and interfaces required for governing service function chains, e.g., enable/disable operations, service chain modification [24]. In particular, a set of control functions are foreseen to establish, maintain and recover service chains while exploiting monitoring status information to detect, e.g., service function unavailability or data delivery degradations along paths bound to a given service chain. Moreover, in [25] the need of addressing the high availability of the service function chains and of the connecting paths is stated as addressed in this work.

8. Conclusions

In this paper, an intent-based service chain layer is presented that allows for the dynamic deployment of service chains paths through the SDN capabilities of edge cloud networks. The proposed approach is flexible and scalable since it offers a simple way for users/applications to express their goals in a high-level manner using a straightforward template-based approach while the technical details for their implementation are left to the network operation system. In this way, the structured intent facilitates the translation of the requirements into configuration directives while being close enough to the natural language used by operators and home users. In addition to the easiness of the service chains paths provisioning, the adaptive features provided by the Intent Layer offer the possibility to activate recovery actions when data delivery degradations are detected thus meeting the requirements specified in the intent. As a further indication of the extensibility of the framework, in the Performance Evaluation section the impact of the variation of the service chains length and the number of cloud platforms in the network were assessed showing that those modifications do not impair existing system functions. On the other hand, for real edge-based applications to be deployed, the results highlighted the necessity of a trade-off between the pervasiveness of the edge cloud computing platforms in the network and the QoE perceived by the users. This proposal presents a step forward in the adoption of intent-based networking in edge cloud networks, which is still not a well explored research area. However, the proposed framework is still under development and some aspects still need to be investigated such as (i) the adoption of more sophisticated parameters for the assurance phase (single flow performance instead of/or in addition to the switches overload), (ii) the improvement of the intents template to include further details strictly related to the edge cloud context and provide a formalized intent language expression, and (iii) the investigation of different network topologies and traffic patterns.

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.

Acknowledgement

We would like to thank Ahmed Ali Mohammed and Hagos Lemlem Adhane for their contribution in the Performance Evaluation section of this paper.

This work was carried out in the framework of the Department of Excellence in Robotics and Artificial Intelligence funded by MIUR to the Scuola Superiore Sant'Anna.

References

[1] L.U. Khan, et al., Edge-computing-enabled smart cities: A comprehensive survey, IEEE Internet Things J. 7 (10) (2020) 10200–10232, http://dx.doi.org/10.1109/JIOT.2020.2987070.

[2] Mobile edge computing (MEC); framework and reference architecture, in: ETSI, ETSI GS MEC 003 1.1.1, 2016.

[3] R.K. Naha, S. Garg, D. Georgakopoulos, P.P. Jayaraman, L. Gao, Y. Xiang, R. Ranjan, Fog computing: Survey of trends, architectures, requirements, and research directions, IEEE Access 6 (2018) 47980–48009, http://dx.doi.org/10.1109/ACCESS.2018.2866491.

[4] M. Muniswamaiah, T. Agerwala, C.C. Tappert, A survey on cloudlets, mobile edge, and fog computing, in: 8th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/7th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom), 2021, pp. 139–142, http://dx.doi.org/10.1109/CSCloud-EdgeCom52276.2021.00034.

[5] A. Filali, A. Abouaomar, S. Cherkaoui, A. Kobbane, M. Guizani, Multi-access edge computing: A survey, IEEE Access 8 (2020) 197017–197046, http://dx.doi.org/10.1109/ACCESS.2020.3034136.

[6] M. Gharbaoui, et al., Resource orchestration strategies with retrials for latency-sensitive network slicing over distributed telco clouds, IEEE Access (2021) 1, http://dx.doi.org/10.1109/ACCESS.2021.3115173.

[7] A. Sathiaseelan, et al., SCANDEX: Service centric networking for challenged decentralised networks, in: Proceedings of the Workshop on Do-It-Yourself Networking: An Interdisciplinary Approach, Association for Computing Machinery, 2015, pp. 15–20.

[8] G. Castellano, et al., A model-based abstraction layer for heterogeneous SDN applications, Int. J. Commun. Syst. 32 (17) (2019) e3989, http://dx.doi.org/10.1002/dac.3989.

[9] T. Braun, et al., Service-centric networking extensions, in: Proceedings of the 28th Annual ACM Symposium on Applied Computing, Association for Computing Machinery, 2013, pp. 583–590.

[10] A.C. Baktir, A. Ozgovde, C. Ersoy, How can edge computing benefit from software-defined networking: A survey, use cases, and future directions, IEEE Commun. Surv. Tutor. 19 (4) (2017) 2359–2391, http://dx.doi.org/10.1109/COMST.2017.2717482.

[11] A. Singh, et al., Intent-based network for data dissemination in software-defined vehicular edge computing, IEEE Trans. Intell. Transp. Syst. 22 (8) (2021) 5310–5318.

[12] A. Clemm, et al., Intent-based networking - concepts and overview, 2020, [Online] Available: https://Bit.Ly/2ImukBF.

[13] T. Szyrkowiec, et al., Automatic intent-based secure service creation through a multilayer SDN network orchestration, IEEE/OSA J. Opt. Commun. Netw. 10 (4) (2018) 289–297.

[14] F. Callegati, et al., Performance of intent-based virtualized network infrastructure management, in: IEEE International Conference on Communications (ICC), 2017, pp. 1–6.

[15] S. Arezoumand, et al., MD-IDN: Multi-domain intent-driven networking in software-defined infrastructures, in: Proc. Int. Conf. Netw. Service Manage. (CNSM), Tokyo, Japan, 2017.

[16] 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.

[17] 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, http://dx.doi.org/10.1109/NOMS47738.2020.9110302.

[18] H. Liao, et al., Learning-based intent-aware task offloading for air-ground integrated vehicular edge computing, IEEE Trans. Intell. Transp. Syst. 22 (8) (2021) 5127–5139.

[19] T. Khan, et al., Intent-based networking approach for service route and QoS control on KOREN SDI, in: Proceedings of IEEE Conference on Network Softwarization (NetSoft), 2021, pp. 1–5.

[20] G. Davoli, et al., Intent-based service management for heterogeneous software-defined infrastructure domains, Int. J. Netw. Manage. 29 (1) (2019).

[21] R. Perez, et al., Alviu: An intent-based SD-WAN orchestrator of network slices for enterprise networks, in: Proceedings of IEEE Conference on Network Softwarization (NetSoft), 2021, pp. 1–5.

[22] P. Krishnan, et al., OpenPATH: Application aware high-performance software-defined switching framework, J. Netw. Comput. Appl. 193 (2021).

[23] J. Zhang, et al., A conflict resolution scheme in intent-driven network, in: IEEE/CIC International Conference on Communications in China (ICCC), 2021, pp. 23–28.

[24] M. Boucadair, et al., Service function chaining (SFC) control plane components & requirements, 2016, https://datatracker.ietf.org/doc/html/draft-ietf-sfc-control-plane-06.

[25] B. Sarikaya, et al., Service function chaining: Subscriber and service identification use cases and variable-length NSH context headers, 2018, https://datatracker.ietf.org/doc/html/draft-sarikaya-sfc-hostid-serviceheader-07.

[26] A. Mohammed, et al., SDN controller for network-aware adaptive orchestration in dynamic service chaining, in: IEEE NetSoft Conference and Workshops (NetSoft), 2016, pp. 126–130.

[27] B. Martini, et al., SDN controller for context-aware data delivery in dynamic service chaining, in: Proceedings of the 1st IEEE Conference on Network Softwarization (NetSoft), 2015, pp. 1–5.

[28] M. Gharbaoui, et al., Implementation of an intent layer for SDN-enabled and QoS-aware network slicing, in: IEEE 7th International Conference on Network Softwarization (NetSoft), 2021, pp. 17–23, http://dx.doi.org/10.1109/NetSoft51509.2021.9492643.

[29] Q. Sun, W. Liu, K. Xie, An intent-driven management framework, [online] https://tools.ietf.org/html/draft-sun-nmrg-intent-framework-00.

[30] T. Erl, SOA, Principles of Service Design, Prentice Hall, 2008.

[31] IEEE standard for service composition protocols of next generation service overlay network, in: IEEE Std 1903.2-2017, 2018, pp. 1–54, http://dx.doi.org/10.1109/IEEESTD.2018.8365908.

[32] Network functions virtualisation (NFV); management and orchestration, in: ETSI, GS NFV-MAN 001 1.1.1, 2014.

[33] Network functions virtualisation (NFV): Architectural framework, ETSI GS NFV 2 (2) (2013) V1.

[34] 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.

[35] S. Charpinel, et al., SDCCN: A novel software defined content-centric networking approach, in: IEEE 30th International Conference on Advanced Information Networking and Applications (AINA), 2016, pp. 87–94, http://dx.doi.org/10.1109/AINA.2016.86.

[36] B. Martini, F. Paganelli, A service-oriented approach for dynamic chaining of virtual network functions over multi-provider software-defined networks, Future Internet 8 (2) (2016) 24.

[37] ONOS intent framework, 2021, https://wiki.onosproject.org/display/ONOS/Intent+Framework.

[38] M. Ghaznavi, et al., Distributed service function chaining, IEEE J. Sel. Areas Commun. 35 (11) (2017) 2479–2489.

[39] Z. Zhou, Q. Wu, X. Chen, Online orchestration of cross-edge service function chaining for cost-efficient edge computing, IEEE J. Sel. Areas Commun. 37 (8) (2019) 1866–1880, http://dx.doi.org/10.1109/JSAC.2019.2927070.

[40] A. Clemm, et al., Intent-based networking - concepts and definitions, 2021, [Online] Available: https://Tools.Ietf.Org/Pdf/Draft-Irtf-Nmrg-Ibn-Concepts-Definitions-05.Pdf.

[41] F. Paganelli, et al., Network service description model for VNF orchestration leveraging intent-based SDN interfaces, in: IEEE Conference on Network Softwarization (NetSoft), 2017, pp. 1–5, http://dx.doi.org/10.1109/NETSOFT.2017.8004210.

[42] OpenFlow switch specification, [online] https://opennetworking.org/sdn-resources/openflow-switch-specification/.

[43] N. McKeown, et al., Openflow: Enabling innovation in campus networks, ACM SIGCOMM Comput. Commun. Rev. 38 (2) (2008) 69–74.

[44] M. Gharbaoui, et al., Cloud and network orchestration in SDN data centers: Design principles and performance evaluation, Comput. Netw. 108 (2016) 279–295, http://dx.doi.org/10.1016/j.comnet.2016.08.029.

[45] Grafana labs, [online] https://grafana.com/.

[46] http://mininet.org/.

[47] Abilene topology, [online] https://web.archive.org/web/20080113120821/http://abilene.internet2.edu/.

[48] Project floodlight, [online] https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview.

[49] https://iperf.fr/.

[50] D. Borsatti, et al., Performance of service function chaining on the OpenStack cloud platform, in: 14th International Conference on Network and Service Management (CNSM), 2018, pp. 432–437.

[51] H. Chai, et al., A parallel placement approach for service function chain using deep reinforcement learning, in: IEEE 5th International Conference on Computer and Communications (ICCC), 2019, pp. 2123–2128, http://dx.doi.org/10.1109/ICCC47050.2019.9064448.

[52] A.M. Medhat, T. Taleb, A. Elmangoush, G.A. Carella, S. Covaci, T. Magedanz, Service function chaining in next generation networks: State of the art and research challenges, IEEE Commun. Mag. 55 (2) (2017) 216–223.

[53] Z. Qazi, et al., Practical and incremental convergence between SDN and middleboxes, Open Netw. Summit (2013).

[54] Y. Zhang, et al., Steering: A software-defined networking for inline service chaining, in: Int. Conf. on Network Protocols (ICNP), IEEE, 2013, pp. 1–10.

[55] I. Cerrato, et al., User-specific network service functions in an SDN-enabled network node, in: 3rd European Workshop on Software Defined Networks, 2014.

[56] J. Blendin, et al., Position paper: Software-defined network service chaining, in: Third European Workshop on Software Defined Networks, 2014, pp. 109–114, http://dx.doi.org/10.1109/EWSDN.2014.14.

[57] F. Callegati, et al., SDN for dynamic NFV deployment, IEEE Commun. Mag. 54 (10) (2016) 89–95.

[58] W. Cerroni, et al., Cross-layer resource orchestration for cloud service delivery: A seamless SDN approach, Comput. Netw. 87 (2015) 16–32.
[59] B. Martini, et al., Experimenting SDN and cloud orchestration in virtualized testing facilities: Performance results and comparison, IEEE Trans. Netw. Serv. Manag. 16 (3) (2019) 965–979, http://dx.doi.org/10.1109/TNSM.2019.2917633.
[60] M. Gharbaoui, et al., Experimenting latency-aware and reliable service chaining in next generation internet testbed facility, in: 2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2018, pp. 1–4, http://dx.doi.org/10.1109/NFV-SDN.2018.8725783.
[61] C. Chang, et al., Performance isolation for network slices in industry 4.0: The 5growth approach, IEEE Access 9 (2021) 166990–167003.
[62] 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, http://dx.doi.org/10.1016/j.comnet.2022.108880.
[63] A. Medhat, et al., Resilient orchestration of service functions chains in a NFV environment, in: IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2016, pp. 7–12.
[64] F. Callegati, et al., Dynamic chaining of Virtual Network Functions in cloud-based edge networks, in: Proceedings of the 1st IEEE Conference on Network Softwarization (NetSoft), 2015, pp. 1–5.
[65] B. Martini, et al., An SDN orchestrator for resources chaining in cloud data centers, in: European Conference on Networks and Communications (EuCNC), 2014, pp. 1–5, http://dx.doi.org/10.1109/EuCNC.2014.6882628.
[66] A. Baktir, et al., How can edge computing benefit from software-defined networking: A survey, use cases, and future directions, IEEE Commun. Surv. Tutor. 19 (4) (2017) 2359–2391.
[67] S. Goudarzi, et al., Dynamic resource allocation model for distribution operations using SDN, IEEE Internet Things J. 8 (2) (2021) 976–988, http://dx.doi.org/10.1109/JIOT.2020.3010700.
[68] Z. Li, E. Peng, Software-defined optimal computation task scheduling in vehicular edge networking, Sensors 21 (3) (2021) http://dx.doi.org/10.3390/s21030955, URL https://www.mdpi.com/1424-8220/21/3/955.
[69] M. Peuster, et al., Let the state follow its flows: An SDN-based flow handover protocol to support state migration, in: 4th IEEE Conference on Network Softwarization and Workshops (NetSoft), 2018, pp. 97–104, http://dx.doi.org/10.1109/NETSOFT.2018.8460007.
[70] S. Secci, et al., Linking virtual machine mobility to user mobility, IEEE Trans. Netw. Service Manage. 13 (4) (2016) 927–940.
[71] W. Rafique, L. Qi, I. Yaqoob, M. Imran, R.U. Rasool, W. Dou, Complementing IoT services through software defined networking and edge computing: A comprehensive survey, IEEE Commun. Surv. Tutor. 22 (3) (2020) 1761–1804, http://dx.doi.org/10.1109/COMST.2020.2997475.
[72] A. Muthanna, A. A. Ateya, A. Khakimov, I. Gudkova, A. Abuarqoub, K. Samouylov, A. Koucheryavy, Secure and reliable IoT networks using fog computing with software-defined networking and blockchain, J. Sensor Actuator Netw. 8 (1) (2019) http://dx.doi.org/10.3390/jsan8010015, URL https://www.mdpi.com/2224-2708/8/1/15.
[73] J.C. Nobre, et al., Vehicular software-defined networking and fog computing: Integration and design principles, Ad Hoc Networks 82 (2019) 172–181, http://dx.doi.org/10.1016/j.adhoc.2018.07.016, URL https://www.sciencedirect.com/science/article/pii/S1570870518305080.
[74] 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.
[75] J. Wolfgang, et al., Research directions in network service chaining, in: IEEE SDN for Future Networks and Services (SDN4FNS), 2013, pp. 1–7.
[76] R. Cohen, et al., An intent-based approach for network virtualization, in: IFIP/IEEE International Symposium on Integrated Network Management (IM), 2013, pp. 42–50.
[77] N. Blum, et al., Insert an intent-based service request API for service exposure in next generation networks, in: 32nd Annual IEEE Software Engineering Workshop (SEW '08), 2008.
[78] 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, http://dx.doi.org/10.1109/NFV-SDN.2018.8725754.
[79] S. Hares, et al., NEMO (NEtwork MOdeling) language, October 2015, https://tools.ietf.org/html/draft-haresibnemo-overview-01.

B. Martini is a Head of Research with the CNIT National Laboratory of Photonic Networks and Technologies (PNT Lab) and an Affiliate Researcher with Sant'anna School of Advanced Studies, Italy. Before joining CNIT PNT Lab, she 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 an Adjunct Professor with Sant'Anna School of Advanced Studies and the University of Pisa, Italy. 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 a Research Engineer 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.