UML-based Modeling and Analysis of 5G Service Orchestration

Ashalatha Kunnappilly
Mälardalen University, Sweden
Email: ashalatha.kunnappilly@mdh.se

Peter Backeman
Mälardalen University, Sweden
Email: peter.backeman@mdh.se

Cristina Seceleanu
Mälardalen University, Sweden
Email: cristina.seceleanu@mdh.se

***Abstract*—The fifth generation of cellular wireless technology, 5G, bears the promise to transform the future network connectivity by providing seamless, low-latency and reliable interconnections between devices. In this paper, we focus on modeling and analyzing 5G service orchestration that deals with virtual network function placement, resource assignment and traffic routing, which are the building blocks of generating network slices catering to various application requirements. In order to ensure that a particular network slice works as stated by the application's service level agreement, it is essential that the constituent virtual network functions are placed in proper hosts, allocated adequate resources in terms of processing power, memory, bandwidth, and routed such that the constraints of the hosts and the network are met. This is a complex problem to solve if one considers the diverse set of requirements of 5G services.**

We tackle this problem by proposing a UML-based modeling and analysis framework, called UML5G Service Orchestration Profile, which allows one to describe 5G network slices and service orchestration via a specialized profile, and analyze associated quality-of-service requirements by checking constraints expressed in Object Constraint Language. Our framework allows a designer to model any candidate orchestration scheme for 5G networks and verify if the network function placement, resource assignment, and routing guarantee the application's quality-of-service requirements, at design time. We evaluate the framework on a prototype implementation of an orchestration algorithm that generates a multitude of allocation configurations that we automatically check against requirements formalized in Object Constraint Language. Our contribution facilitates modeling and design-time evaluation of network slicing and service orchestration schemes in 5G-based solutions.

***Index Terms*—5G, Network Slicing, Service Orchestration, UML 2.0, UML5G Service Orchestration profile**

I. Introduction

The **5G** technology has been proposed to address the shortcomings of 4G networks in terms of congestion and support of heterogeneous applications, by utilizing a less crowded, higher bandwidth (up to 20Gbps) spectrum that can achieve a connection density as large as 1 million devices per square kilometer, and meet a latency of as low as 1 ms in radio access network [1]. A 5G network is divided into the so-called *network slices* that are independent logical networks on a shared infrastructure, which can be optimized to serve the higher bandwidth, low latency or enhanced broadband requirements [2]. In general, a network slice consists of a number of *Virtual Network Functions* (VNFs) that need to be interconnected or chained, according to a *VNF Forwarding Graph* (VNFFG), to fulfill application requirements and meet specific network constraints in order to realize a particular use-case scenario [3] belonging to, for instance, the health, automotive or media domains, respectively. Since each VNF has a resource requirement in terms of processing power and storage, and chaining them results in additional overheads in terms of bandwidth capacity and connection latency, it is crucial that these VNFs are adequately placed onto available hosts, such that the application requirements are met [4], [5].

There are now standardized slices defined for enhanced Mobile Broadband (eMBB), massive-machine-type communication (mMTC), ultra-reliable low latency communication (uRLLC), and Vehicle to Everything (V2X), which can cater for the specific requirements of targeted applications [6]. However, as already mentioned, in order to benefit from the potential of 5G network slicing, one needs to apply *service orchestration* schemes [7] that allow different slices to efficiently share the available network resources (processing units, storage, and bandwidth) to achieve different quality-of-service (QoS) requirements. Since service orchestration consists of decisions regarding VNF placement on hosts, but also resource assignment to each VNF, and traffic routing over network links, the problem is complex due to the impact that decisions have on one another [5], [8], which might end up in breaching requirements. A promising way of capturing such impacts is by specifying the system in a well-understood modeling language, and automatically generating and checking associated constraints of possible orchestration schemes resulting from applying given algorithms.

In this paper, we address the above by proposing a UML 2.0-based modeling framework, named **UML5G Service Orchestration (UML5G-SO)**. The latter is based on a UML Profile [9] that we define, which allows one to model network slicing and service orchestration depicting VNF placement, implicit assignment of required resources to the VNFs and traffic routing. In addition, we demonstrate our approach with the UML-based Specification Environment (USE) tool [10], in which we model and check candidate allocations against requirements captured in the Object Constraint Language (OCL), in order to identify the set of feasible ones, at design time. We carry out the experimental evaluation of the method on a prototype implementation of a greedy algorithm for service orchestration, as well as of an enumerative solution of generating VNF allocations and routing schemes.

The rest of the paper is organized as follows. Section II

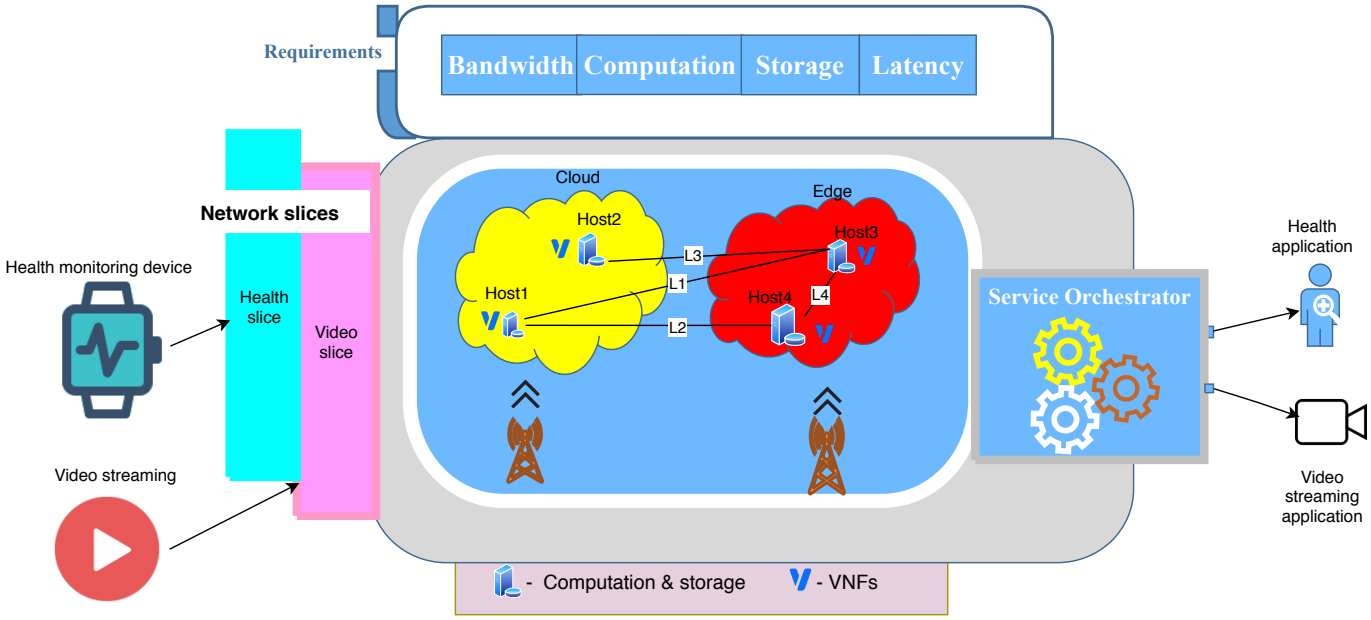

Fig. 1. Use Case Description

details the problem statement. In Section III, we overview the preliminaries of UML 2.0 modeling, OCL and the USE tool. In Section IV, we present our UML 2.0 profile allowing the modeling of network slicing and service orchestration in 5G systems. Section V shows the specification and validation of the system by logic-based analysis using the USE tool, the experimental evaluation, and a brief discussion of the gained insights. Section VI compares our contribution to related work, whereas VII provides the concluding remarks with some directions of future work.

II. Problem Statement

The main goal of the paper is to provide a modeling and analysis framework of 5G orchestration schemes. As a running example, we consider a system with a set of computing hosts deployed in the edge and cloud, respectively, which needs to support applications from various domains, e.g., health and media, via network slices. The hosts are characterized by their *processing power*, *storage*, *capabilities* to execute VNFs (interchangeably called services henceforth), and *support* of mobile edge computing (MEC). We assume that edge hosts are MEC capable, whereas cloud hosts are not. The host nodes form an overlay network where different nodes are connected via links. In this paper, we consider only virtual links defined by bandwidth capacity and latency, respectively. Throughout the paper, we consider a 5G network slice as a chain of VNFs. The VNFs have respective resource requirements in terms of computation and storage, execution time, and a constraint of whether they require mobile edge computing, hence need to be placed in host devices with such capability.

Given the above scenario, in order to satisfy the application requirements on latency and bandwidth, it is essential that the VNFs are placed in adequate hosts respecting the constraints, are assigned enough resources on hosts respecting their requirements, and subjected to a traffic routing scheme across the links, which respects the VNF chaining sequence and the link constraints of capacity and delay. This problem is called *service orchestration*, and it is difficult to solve due to the potential impact that VNF placement, implicit resource assignment, and traffic routing may have on one another, deeming the final orchestration scheme not suitable for particular applications.

Hence, it is essential that orchestration solutions need to be designed such that QoS guarantees can be provided, especially in worst-case scenarios where maximum system capacities are exploited. To achieve this, one needs to verify if the VNF placement and the routing scheme (service orchestration scheme) fulfill the application requirements.

In this paper, we tackle this problem and provide a UML-based modeling and analysis framework, named **UML5G-SO**, which assists designing such service orchestration schemes by facilitating design-time checks in OCL, to evaluate if the scheme guarantees the application requirements.

Use case description. The 5G-based system under consideration is depicted in Fig. 1. We examine a scenario with two applications, namely a health-monitoring application on a health band that sends alarms in case of health-parameter deviations, and a video streaming application. The two applications have their specific real-time and bandwidth requirements, so we assume that they use two 5G network slices, respectively, a *health slice*, and a *video slice*. The health slice needs to deal with the communication from the user equipment (health band) to the hospital in case of health emergencies. In most cases, the health emergencies are critical and the hospital needs to be alerted as soon as possible (real-time), yielding requirements on the maximum allowed communication latency. In addition, after acknowledging the alarm, the doctor/caregiver may in

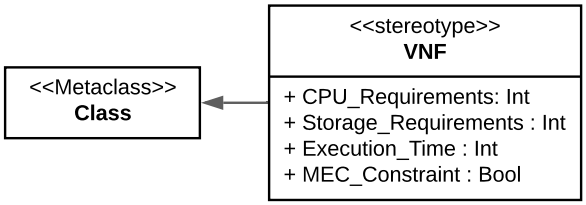

Fig. 2. An example of a stereotype definition in an UML Profile

some cases request video conferencing with the patient, which translates into requirements on bandwidth as well. However, latency requirements are more stringent for such applications, hence we consider the health slice as an instance of the standard uRLLC slice, with certain bandwidth requirements. The video slice, in comparison, is used by video-streaming users, thus it has very high requirements on bandwidth. However, there are also soft requirements on latency for this slice, to ensure that the video streaming experience is pleasant. Consequently, we consider the video slice as an instance of the eMBB slice, with soft real-time latency requirements.

In our case, the health slice is a VNF chain consisting of VNF1, VNF2, VNF3 and VNF4, and the video slice is a VNF chain of VNF5 and VNF6. The VNFs can be those defined in the 5G architecture [2] (e.g., Core Access and Mobility Management Function (AMF), Data network (DN) etc.), or domain-specific VNFs. The order in which VNFs are chained is defined by the *VNF Forwarding Graph* (VNFFG). In general, a VNFFG can have many different structures, but for simplicity we consider in this paper only VNFFG where VNFs are sequentially connected, named VNF Forwarding Sequential Graph (VNFFSeq). In the use case, we assume that we have two cloud hosts (Host1 and Host2) and two edge hosts (Host3 and Host4). As shown in Fig. 1, Host1 and Host3 are connected via virtual link L1, Host1 and Host4 via L2, Host2 and Host3 via L3, and Host3 and Host4 via L4.

In order to capture service orchestration in such a 5G-based system, we need to be able to model the allocation of the VNFs, namely VNF1 to VNF6, to the various hosts, Host1 to Host4, together with a routing scheme for each slice. To accomplish this, in Section IV we present our **UML5G-SO** framework that we apply on this use case. Further, in order to analyze the feasibility of candidate orchestration configurations for our system, we show how to check if the VNFs are allocated such that the resource constraints are met, and also if the selected routing scheme meets the end-to-end latency and bandwidth requirements.

III. Preliminaries

In this section, we briefly overview the UML 2.0 *Profile Diagram, Object Diagram*, OCL constraints, and the USE tool.

A. UML Profile, Class, and Object Diagrams

UML 2.0 profiles are structural diagrams that offer the possibility to extend UML 2.0 by defining *stereotypes*, *tagged values*, and *constraints* for capturing domain-specific concepts [9]. For instance, in Fig. 2, we present a *stereotype*, VNF, extending the UML 2.0 metaclass, Class. The stereotype VNF is associated with a set of properties (tag definitions) to represent the VNF's CPU requirements, Storage Requirements, Execution Time and MEC constraints. When this stereotype is applied to a specific model element, the values taken up by these tag definitions are referred to as tagged values. We have not represented any constraints for our VNF stereotype. In this work, we define a new **UML5G-SO** profile that facilitates the modeling of service orchestration of 5G-based systems.

UML 2.0 class diagrams provide visualization of the classes that extend our **UML5G-SO** stereotype. In order to represent an instance of our use case at a particular time, we use the *UML 2.0 object diagram*.

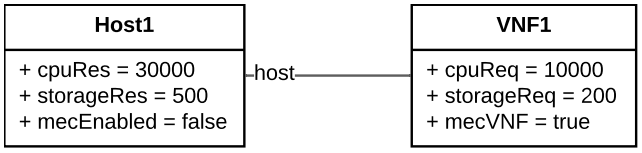

Fig. 3. Example of simple system

B. OCL and the USE tool

The Object Constraint Language (OCL) [11] is a declarative language in which constraints over UML models can be specified. For instance, one can specify that an attribute value must lie within a certain range. In this work, we use OCL to capture Boolean conditions which, if satisfied by a certain orchestration configuration, guarantee that the quality of service is met. As an example, consider the UML specification of a host and an allocated VNF in Fig 3.

In this system, we would like to enforce that the storage and CPU requirements are met by the available resources of the host, and if the VNF requires MEC-capabilities then the host must have them. We can formalize this as constraints:

$$cpuRes \geq cpuReq \tag{1}$$

$$storageRes \geq storageReq \tag{2}$$

$$mecEnabled \vee \neg mecVNF \tag{3}$$

In Fig. 3, we can see that the first two constraints are satisfied while the last one is not, indicating that the system is not valid. We do not delve into the details of the OCL syntax here, instead we simply note that it allows for expressing all constraints presented in the paper. The USE tool [10] is a software program that enables one to check OCL constraints automatically. We apply it to show, as a proof of concept, that it is possible to automate the verification of the 5G models of service orchestration.

IV. UML5G-SO Modeling Framework

In this section, we overview our proposed **UML5G Service Orchestration (UML5G-SO)** framework, which enables us to model and analyze 5G service orchestration schemes. We start

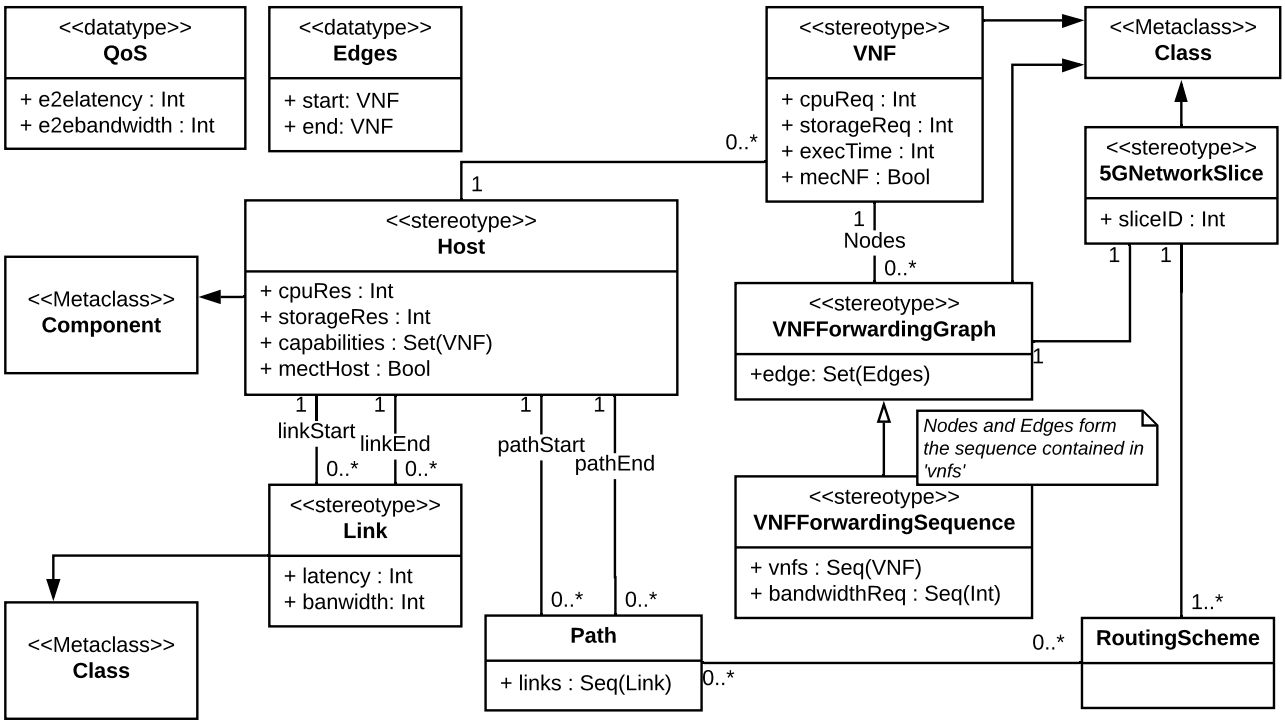

Fig. 4. UML5G Service Orchestration Profile

by presenting a formalization of the system definitions of our framework. Thereafter, we present the UML5G-SO profile, which can be used for modeling 5G service orchestration configurations in any tool that supports UML. Next, we illustrate the modeling approach by applying the UML5G-SO profile to our use case (described in II), and provide the system-specific class diagram and the object diagram views, respectively. In the forthcoming section, we also introduce the analysis part of our framework, using the USE tool.

A. System Definitions

We begin by proposing a mathematical description of a 5G service orchestration system, which we use as a basis for our UML modeling and the subsequent OCL constraint formalization.

Definition 1: (5G Service Orchestration System) We define a *5G Service Orchestration System* as the following tuple:

$$5GSOSys \triangleq \langle \mathcal{F}, \mathcal{V}, \mathcal{H}, \mathcal{L}, \mathcal{S} \rangle,$$

where each component is defined as follows:

1) $\mathcal{F} = \{f_1, \ldots, f_n\}$, $n \in \mathbb{N}$, is the set of *network functions*, which are available in the 5G system.
2) $\mathcal{V} = \{v_1, \ldots, v_n\}$ is the set of *virtual network functions*, where a VNF is defined as: $v_i = \langle nf_i, cpuReq_i, storReq_i, mecVNF_i, execTime_i \rangle$, $\forall i \in [1, n]$. Here, $nf_i \in \mathcal{F}$ represents the network function that the VNF caters for, $cpuReq_i, storReq_i \in \mathbb{N}$ represent v_i's minimum CPU and storage requirements, respectively, $mecVNF_i$ indicates whether the VNF v_i has a MEC constraint such that it can only be executed in a MEC-compatible host, and $execTime_i$ defines the execution time of v_i.
3) $\mathcal{H} = \{h_1, \ldots, h_n\}$ is the set of *hosts* in the system, where $h_i = \langle cpuRes_i, storRes_i, capabilities_i, mecHost_i \rangle$, $\forall i \in [1, n]$. Here, $cpuRes_i, storRes_i \in \mathbb{N}$ are constants representing the CPU and storage resources available at host h_i, respectively, $capabilities_i \in 2^{\mathcal{F}}$ is a set of network functions that h_i is capable of executing, and $mecHost_i$ is a Boolean variable indicating whether h_i is MEC-enabled or not.
4) $\mathcal{L} = \{l_1, \ldots, l_n\}$ is the set of *links* connecting hosts, where $l_i = \langle h_{start_i}, h_{end_i}, llat_i, lbw_i \rangle$, $\forall i \in [1, n]$. Here, $h_{start_i}, h_{end_i} \in \mathcal{H}$ denote the start host and the end host of l_i, respectively, $llat_i$ denotes l_i's latency, and lbw_i is the bandwitdh provided over l_i.
5) $\mathcal{S} = \{s_1, \ldots, s_n\}$ is the set of *5G network slices*, where $s_i = \langle sliceId_i, slat_i, sbw_i, VNFFSeq_i \rangle$, $\forall i \in [1, n]$. Here, $sliceId_i$ refers to the slice identification number and $slat_i, sbw_i \in \mathbb{N}$ are the required end-to-end latency and bandwidth of slice s_i, respectively. The tuple element $VNFFSeq_i = \langle v_1^i, v_2^i, \ldots, v_j^i \rangle$, with $v_j^i \in \mathcal{V}$ represents the sequence of particular VNFs that are chained in slice s_i, indicating what VNFs need to be used and in what order.

Given a particular 5G service orchestration system, there are many possible *candidate configurations*, that is, allocations of VNFs on hosts, assignments of resources, and routing schemes, which may or may not fulfill all requirements.

Definition 2: (Candidate Configuration) We define a candidate configuration (CC) as the following tuple:

$$CC_{5GSOSys} \triangleq \langle \mathcal{A}, \mathcal{R} \rangle,$$

where each component is defined as follows:

1) $\mathcal{A} : \mathcal{V} \rightarrow \mathcal{H}$ is the *VNF allocation*, a function assigning one host to each VNF, which decides on which host each VNF is to be executed.
2) $\mathcal{R} : \mathcal{S} \times \mathcal{H} \times \mathcal{H} \rightarrow \mathcal{P}$ is the *routing scheme*, assigning for each slice and pair of hosts, a path to use for communication. Here, $\mathcal{P} = (p_1, \ldots, p_n)$, where $p_i = (l_1^i, \ldots, l_n^i)$, $l_i^i \in \mathcal{L}$, $\forall i \in [1, n]$. Intuitively, given a routing scheme $\mathcal{R}$, $\mathcal{R}(s_1, h_1, h_2)$ returns the sequence of virtual links traversed when slice s_1 is chaining its selected VNFs between host h_1 and host h_2.

For simplicity, when defining routing schemes for a CC, we only include paths that are relevant (i.e., paths between hosts of allocated VNFs located next to each other in a VNFSeq chain).

Example 1: To help understanding, we now exemplify the above definitions using our 5G use case. The input system is $UC = \langle \mathcal{F}_{UC}, \mathcal{V}_{UC}, \mathcal{H}_{UC}, \mathcal{L}_{UC}, \mathcal{S}_{UC} \rangle$ with:

- $\mathcal{F}_{UC} = \{A, B, C\}$
- $\mathcal{V}_{UC} = \{v_1, v_2, v_3, v_4, v_5, v_6\}$, where:
  $v_1 = \langle B, 1000, 100, true, 5 \rangle$,
  $v_2 = \langle A, 200, 50, true, 5 \rangle$,
  $v_3 = \langle A, 1000, 100, false, 5 \rangle$,
  $v_4 = \langle C, 500, 70, false, 10 \rangle$,
  $v_5 = \langle A, 1000, 100, true, 5 \rangle$,
  $v_6 = \langle C, 1000, 500, false, 10 \rangle$
- $\mathcal{H}_{UC} = \{h_1, h_2, h_3, h_4\}$, where
  $h_1 = \langle 30000, 500, \{A, C\}, false \rangle$,
  $h_2 = \langle 15000, 900, \{A, C\}, false \rangle$,
  $h_3 = \langle 2000, 300, \{A, C\}, true \rangle$,
  $h_4 = \langle 1000, 150, \{A, B\}, true \rangle$
- $\mathcal{L}_{UC} = \{l_1, l_2, l_3, l_4\}$, where:
  $l_1 = \langle h_1, h_3, 4, 100 \rangle$, $l_2 = \langle h_1, h_4, 2, 100 \rangle$,
  $l_3 = \langle h_2, h_3, 10, 200 \rangle$, $l_4 = \langle h_3, h_4, 5, 200 \rangle$
- $\mathcal{S}_{UC} = \{s_1, s_2\}$, where:
  $s_1 = \langle 1, 60, 2, (v_1, v_2, v_3, v_4) \rangle$, $s_2 = \langle 2, 100, 4, (v_5, v_6) \rangle$

A possible CC is the following allocation and routing scheme:

- $\mathcal{A}(v_1) = h_2$, $\mathcal{A}(v_2) = h_3$, $\mathcal{A}(v_3) = h_4$,
  $\mathcal{A}(v_4) = h_2$, $\mathcal{A}(v_5) = h_3$, $\mathcal{A}(v_6) = h_1$
- $\mathcal{R}(s_1, h_2, h_3) = l_3$,
  $\mathcal{R}(s_1, h_3, h_4) = l_4$,
  $\mathcal{R}(s_1, h_2, h_3) = l_3$,
  $\mathcal{R}(s_1, h_4, h_2) = l_3, l_4$,
  $\mathcal{R}(s_2, h_3, h_1) = l_1$

In the above CC, we can see that v_1 is allocated to host h_2, v_2 is allocated to h_3, and when slice s_1 chains VNF v_1 and v_2 it uses the path consisting of the single link l_3.

Given a 5G SO system and a candidate configuration, it is possible to check whether all constraints in terms of QoS are met. In the remainder of the paper, we describe how the SO system and candidate orchestration configuration can be modeled in UML, using our UML5G profile, as well as how the constraints can be formalized in OCL, to allow for the automatic verification of a given configuration.

B. The UML5G Service Orchestration Profile

The UML5G Service Orchestration Profile proposed in this paper provides a UML 2.0-based framework intended for modeling and analysis of service orchestration schemes in 5G-based systems. The profile is depicted in Fig. 4. The stereotype *Host* is defined to specify the hosts in our overlay network and it extends the *UML Metaclass::Component*. We have also defined stereotypes to specify *5GNetworkSlice, VNF, VNFForwardingGraph*, and *Link*, by extending the *UML Metaclass::Class*. The stereotypes are constructed to match the system definitions given in subsection IV-A. For instance, our network slice stereotype has attributes for specifying the *slice id*, and is associated with a set of *QoS requirements* that the slice should meet[1]. Each *5GNetworkSlice* has an association to a *VNFForwardingGraph* (VNFFG). In our profile, we have a specialization of the VNFFG, that is, *VNFForwardingSequence*, which restricts the VNF chaining to a sequence that represents one of the common ways of chaining VNFs. However, the profile can also be extended with other mechanisms of VNF chaining, for instance, branching. We have also defined datatypes to specify the QoS attributes of the slice and to specify the edges of our *VNFForwardingSequence*.

C. Use-Case Modeling with Class and Object Diagrams

In this section, we demonstrate our UML5G-SO profile to model the use case overviewed in Section II. In order to use the profile for our specific example, we first identify how the profile can be applied to match our use case; we show this by using the class diagram description as follows.

Example 2: The class diagram depicted in Fig. 5 shows how our use case supports modeling of two categories of hosts, namely *Edge* and *Cloud*, connected via *virtual links*. In addition, we keep the network functions abstract, naming them *VNF-A, VNF-B* and *VNF-C*. In addition, in this use case, we consider two categories of *5GNetworkSlice*: *eMBB* and *uRLLC*, with VNFs chained as sequences, represented by *VNFFSeq*. Our modeling framework describes potential routing schemes *RoutingScheme* and *Path*. It should be noted that the class diagram in Fig. 5 respects all the associations and multiplicities depicted in the UML5G-SO Profile Diagram of Fig. 4.

Given the class diagram, we now present a snapshot of our use-case system at a particular time instance, using an object diagram representation. The object diagram provides the complete system view at a particular time. It consists of our input, that is, the set of network slices, hosts, VNFs and

[1]It is important to emphasize that this stereotype can be extended with parameters specifying slice life cycle, geographical coverage, etc., but since we do not use these parameters in our system analysis in the current work, we have restricted to the above definition.

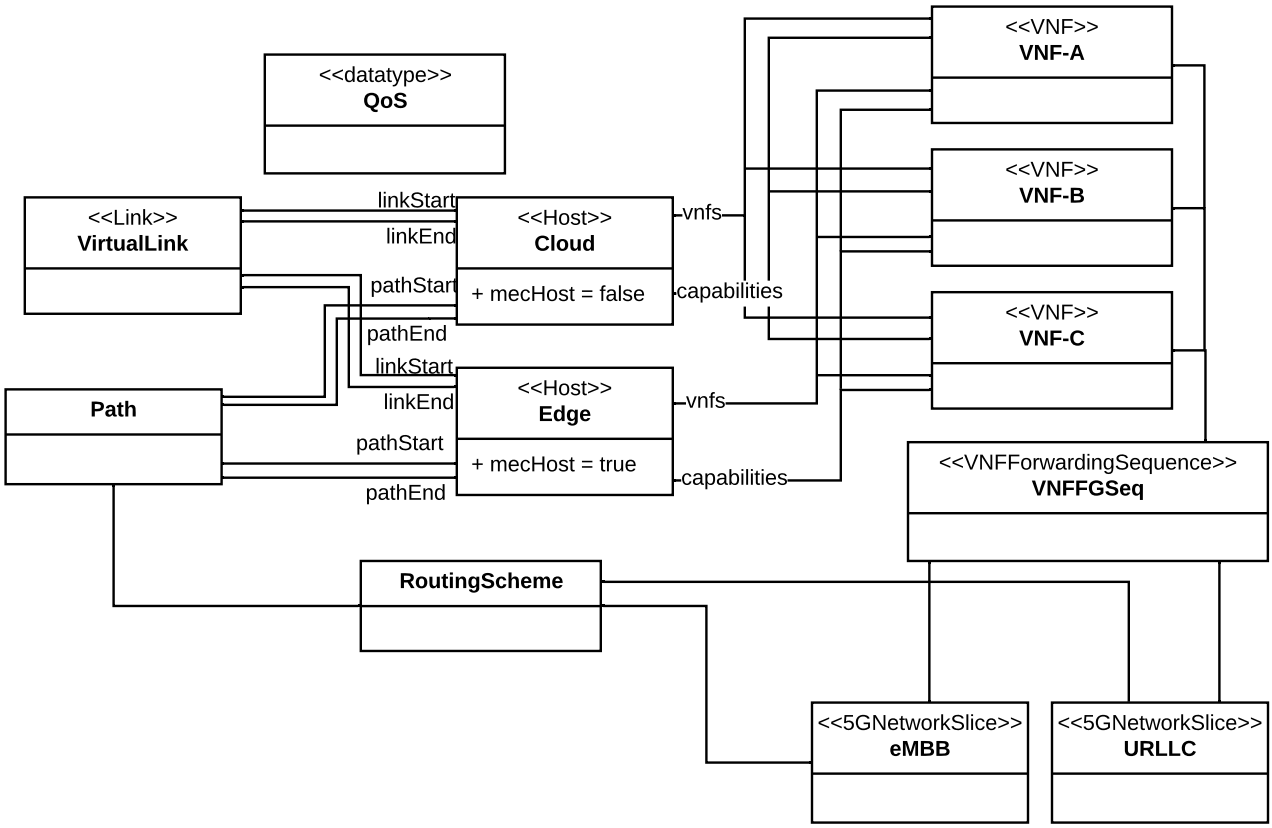

Fig. 5. Class Diagram (It inherits all associations and multiplicities depicted in Fig. 4)

their forwarding graphs, and it depicts a possible candidate configuration with an allocation and routing scheme for each slice, respectively.

Example 3: The object diagram representation of our use case is depicted in Fig. 6. We consider two different applications (Health and Video streaming) with requirements on bandwidth requirement and latency, and requesting two slices - *HealthSlice* and *VideoSlice*, which are of type *uRLLC* and *eMBB*, respectively. The health slice is identified by an identity number, namely, *slice_id=1*; it has 4 VNFs, that is, *VNF1, VNF2, VNF3* and *VNF4*, an end-to-end latency requirement of at most 60ms in case of emergency alarms, and a bandwidth requirement of 100Gbps. Similarly, the video slice, with *slice_id=2*,/ has 2 VNFs, *VNF5* and *VNF6*, with an end-to-end latency and bandwidth requirements of 240ms and 50Gps, respectively. For the purpose of our study, we consider four hosts, out of which two are edge hosts and two are cloud hosts. Each host has respective constraints on processing, storage, as well as on which network functions it can execute, as depicted in Fig. 6. For instance, *Host1*, which is a cloud node, has a processing power 30000 GBps, a storage capacity of 500 GB and VNF capabilities that allow the execution of virtual network functions of type *VNF-A* and *VNF-C*. In addition, for each host, we have a Boolean constraint showing whether the host is MEC-capable or not. In our case, *Host3, Host4* are MEC capable, while *Host1, Host2* are not.

Moreover, as shown in Fig. 6, each of the VNFs has resource constraints, an execution time, and a MEC constraint, respectively. As an example, *VNF1* requires a processing power of 1000 GBps, storage of 100 GB, has an execution time of 5 ms, and a MEC constraint, implying that it can only be executed in MEC-capable hosts.

Furthermore, the *HealthSlice* and *VideoSlice* each have an associated VNFFSeq, *VNFFSeq1* and *VNFFSeq2*, respectively, as well as a routing scheme, *RS1* and *RS2*. Here, we illustrate via an example of the health slice. The *VNFFSeq1* depicts the VNF sequence, *VNF1, VNF2, VNF3, VNF4*, which is an ordered list of VNFs. In addition, *RS1* creates an association to *P1* and *P2*, connecting *Host1* and *Host2* through link *L1*, and *Host2* and *Host3* through links *L1, L2*, respectively. We show an example of modeling and verifying the VNF placement to hosts, which is generated by a slightly modified version of the minimum latency greedy algorithm [12], [4]. The modified version of our algorithm is shown as pseudocode, see Algorithm. 1. The output of the algorithm is a resource assignment and placement of VNFs to existing hosts, as well as a routing path minimizing the overall latency. One of the candidate configurations that we obtain by employing the algorithm in our use case is demonstrated in the object diagram representation of Fig. 6. As shown, our candidate configuration is the following: *VNF1* is placed on *Host2*; *VNF2*

on *Host1*; *VNF3,VNF4,VNF5* on *Host3* and *VNF6* in *Host4*, by respecting the resource constraints of both VNF and hosts. For *HealthSlice*, routing scheme *RS1* with paths *P1* and *P2* is chosen, and for *VideoSlice*, scheme *RS2* with *P3* is selected.

In the forthcoming section, we illustrate the analysis of our approach using the USE tool, wherein we check whether the respective greedy algorithm has generated VNF placement candidates that meet the implicit resource requirements of the VNFs, and routing schemes that allow fulfilling the application requirements.

V. Logic-based analysis using USE tool

A major benefit of an adequate modeling of the system is that it allows an automatic procedure to verify properties. In particular, we can check that the quality of service of each slice is guaranteed to hold. We begin by formalizing the constraints, such that we can leverage them and use existing tools to automatically check them on specified models. For readability, we introduce a set of auxiliary functions, as follows:

$$count(link, path)$$

gives the number of times $link$ occurs in $path$.

$$\mathcal{A}^{-1}(h)$$

is the inverse of the allocation function, that is, it gives all VNFs allocated to host h. Finally,

$$slicePath(s, i) = \mathcal{R}(s, \mathcal{A}(s.vnffseq[i]), \mathcal{A}(s.vnffseq[i+1])),$$

where $vnffseq[i]$ denotes the ith element of the sequence $vnffseq$, gives the path connecting the hosts of which the ith and $i+1$th VNFs of the VNF sequence of the slice s are allocated.

Moreover, we use '.' to denote elements of tuples. For example, if $s \in \mathcal{S}$ and $s = \langle sliceId, slat, sbw, VNFFSeq \rangle$, then $s.slat$ denotes $slat$ of s.

A. Constraints

1) LinkNotOverloaded: Each link must have sufficient capacity for its utilization:

$$\forall l \in \mathcal{L}$$
$$\sum_{s \in \mathcal{S}} \left(\sum_{i=1}^{|s.vnffseq|-1} count(l, slicePath(s, i)) \right) \cdot s.sbw \leq l.lbw$$

2) VNFBandwidthWithinBounds: There must be enough bandwidth capacity between each pair of chained VNFs:

$$\forall s \in \mathcal{S}, \forall i \in [1..|l.vnffseq| - 1]$$
$$(min_{l \in slicePath(s,i)} l.lbw) \geq s.sbw$$

3) CompatibleHost: Each VNF must be allocated to a host that has the capability to process the particular VNF network function:

$$\forall v \in \mathcal{V}$$
$$v.nf \in (\mathcal{A}(v)).capabilities$$

4) MECEnabled: If a VNF requires a MEC-enabled host, the allocation must respect that:

$$\forall v \in \mathcal{V}$$
$$\neg v.mecVNF \vee (\mathcal{A}(v)).mecHost$$

5) HostHasEnoughComputationResources:

$$\forall h \in \mathcal{H}$$
$$\left(\sum_{v \in \mathcal{A}^{-1}(h)} v.cpuReq \right) \leq h.cpuRes$$

6) HostHasEnoughStorageResources:

$$\forall h \in \mathcal{H}$$
$$\left(\sum_{v \in \mathcal{A}^{-1}(h)} v.storReq \right) \leq h.storRes$$

7) LatencyWithinBounds:

$$\forall s \in \mathcal{S}$$
$$\left(\begin{array}{c} \sum_{vnf \in s.vnffseq} vnf.execTime + \\ \sum_{i=1}^{|s.vnffseq|-1} \sum_{p \in slicePath(s,i)} \sum_{l \in p} l.llat \end{array} \right) \leq s.slat$$

We have formalized all constraints in OCL, and encoded them into the USE tool[10] together with the UML 2.0 profile from Sec. IV-B. Therefore, an object diagram representation of a particular 5G service orchestration system can also be encoded into the USE tool to automatically check that the particular instance meets all the above constraints. If all relevant constraints have been captured in OCL, this allows for an automatic procedure to verify a CC for a given 5G-based system.

B. Experimental Evaluation

To demonstrate how an automatic verification process could work, we consider a simple greedy algorithm, inspired by one of the deliverables in the EU H2020 5GTransformer project [4], and present how it is possible to check its output. Given a 5G-based system, the algorithm generates a candidate configuration, that is, an allocation and a routing scheme, by selecting, at each step, the choice that minimizes the latency. The algorithm iterates over all slices, and for each slice goes through the chain of VNFs. The first VNF, if not already allocated, can be placed anywhere. For each following VNF of the chain, the "closest" (i.e., with minimum possible latency from the host of previous VNF) as well as "possible" (i.e., with enough resources and capability for the considered VNF) host is chosen for allocation. When considering a host, to compute the minimum latency, if no previous path has been selected for routing, all possible paths are considered, and the one with the lowest latency is chosen and added to the routing scheme. The steps are repeated until all VNFs have been placed. If at any point no suitable host can be found, the algorithm returns failure (i.e., no backtracking is performed). The algorithm is shown in Alg. 1.

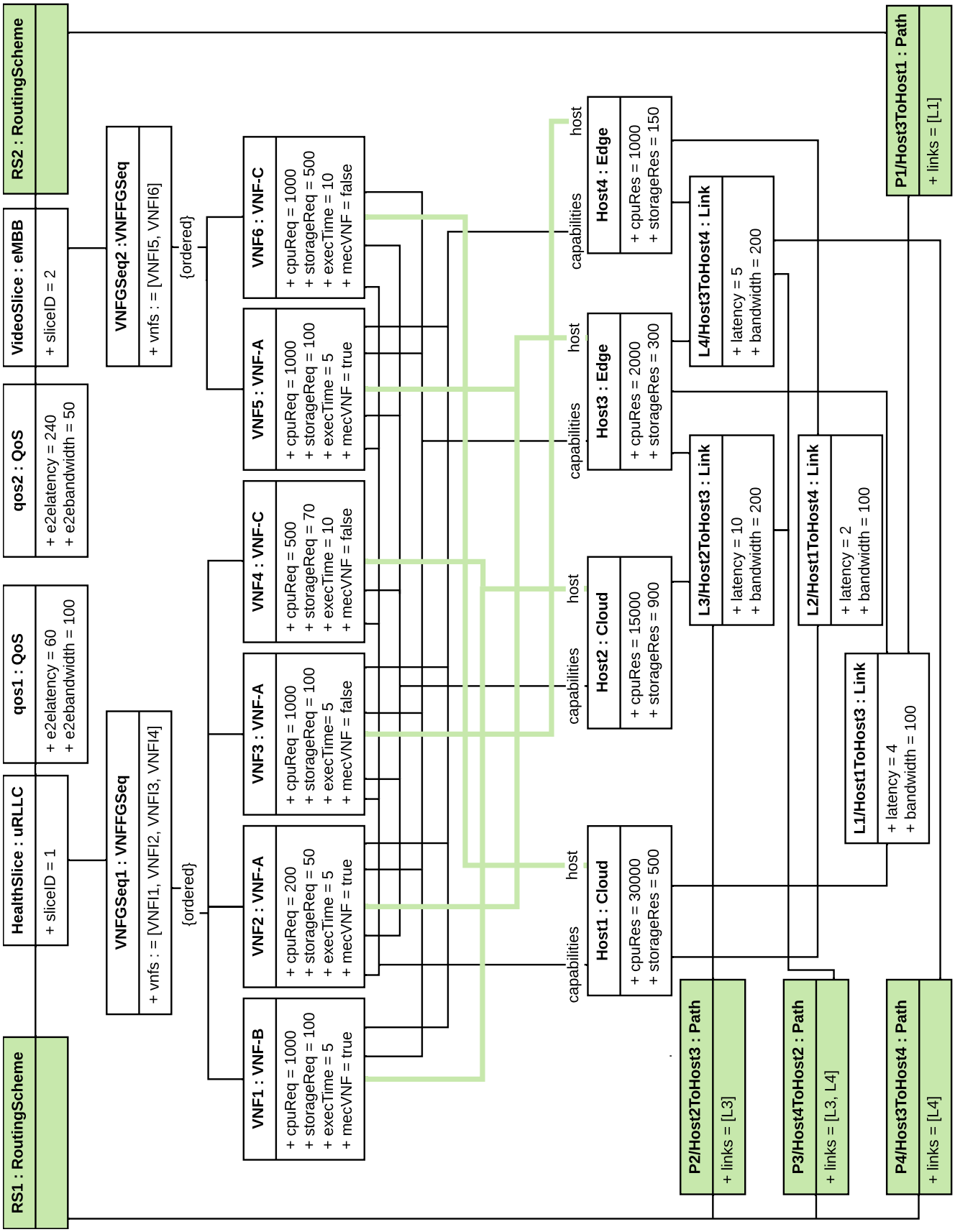

Fig. 6. Object Diagram (It inherits all associations and multiplicities depicted in Fig. 4)

```
Input  : Hosts, Slices, Links
Output: Allocation, Routing
initialization;
for s in slices do
    chain ← s.vnffseq
    if chain.head not allocated then
        allocate chain.head anywhere
        if no suitable host then
            return Fail
        end
    else
        // vnf_1 is allocated
        for (vnf_1, vnf_2) with (bwReq) in chain.tail do
            allocate vnf_2 to host with minimum latency
            if no suitable host then
                return Fail
            end
            add chosen path to routing scheme
            remove resources from resp. host and links
        end
    end
end
```

Algorithm 1: Greedy allocation algorithm.

Category	Count	Verified
Greedy	567	567
Enumerative	110	110
Infeasible	323	-

TABLE I
RESULTS FROM EXPERIMENTAL EVALUATION.

Example 4: The configuration presented in Ex. 1 has been obtained by running the greedy algorithm on the corresponding 5G-based system. We can use the USE tool, as mentioned above, to verify that the configuration indeed fulfills all constraints.

Furthermore, we can query the USE tool to give us the actual end-to-end latency of all the slices. We sum all of these together to obtain the objective value of the configuration. By generating all possible candidate configurations, keeping the ones fulfilling the constraints and querying the latency of each configuration, we can observe that in total there are thirteen actual solutions (combinations of allocation and routing scheme). The lowest possible total latency among the solutions is 61 ms, compared to the total latency of 74 ms obtained from the greedy algorithm.

As a second demonstration, we generate 1000 input systems randomly (each with two different network functions, four hosts, four links and two slices) with random capabilities, capacities and requirements. Due to the randomness, some of the inputs will have solutions, while some have contradictory constraints that are impossible to satisfy. For each input we run the greedy algorithm. In case that the greedy algorithm fails to find a solution, we enumerate all possible configurations to see if at least one exists. We then run the USE tool to verify the *greedy* or the *enumerative* solution. The resulting numbers are shown in Table I.

Note that this evaluation does not verify the greedy algorithm itself, but it allows one to verify particular generated orchestration configurations, ensuring that they are correct before actually using them. Moreover, the greedy algorithm has no special role in this, it was picked due to being quite straightforward. In general, any algorithm can be used.

Both experimental evaluations show how our UML5G-SO profile can be used with OCL constraints to automatically verify candidate configurations. Checking a particular candidate is done very quickly (< 20 ms), allowing for verification before, for example, deployment. Of course, generating all possible candidates is time consuming and not scalable for even slightly larger systems. However, this work establishes a baseline from which further expansion is possible. The profile uses a well-understood and industrially-adopted language of designing (or generating) input systems, as well as specifying candidate configurations, in addition benefiting from many tools that support UML modeling. This yields a connection between a user-friendly language, UML, and mathematical rigor.

VI. Related Work

While substantial work has been recently devoted to solving the service orchestration problem with respect to VNF placement and resource assignment in a 5G networking context [7], [13], [14], not much research focuses on providing modeling and analysis support for describing 5G-based orchestration systems and their solutions, and checking the latter against QoS requirements. Models, workflows, and tools to create and automatically verify service orchestration solutions in 5G-based systems, against QoS requirements, are still very limited or completely missing, and need to be established to fully support the development of 5G network applications.

Some initial approaches for this have been presented by the SONATA project [15], which provides a set of SDK tools that support service developers to create and ship new network services. This SDK also offers descriptor validation functionalities that go beyond simple syntax or schema checks, e.g., the automatic detection of loops in virtual network function forwarding graphs [16]. However, this work is limited to such checks and does not offer a logic-based design-time analysis of the orchestration scheme against latency, bandwidth and storage requirements.

In one of the deliverables of the EU H2020 project 5GTRansformer [7], which our greedy placement algorithm is inspired from, the contributors propose an approach to find an optimal solution to 5G VNF orchestration, by using a relaxation strategy that replaces binary variables by real variables bounded between 0 and 1, together with VNF placement heuristics in order to avoid getting infeasible solutions. In contrast, our approach generates orchestration candidates and verifies them automatically, hence eliminating the infeasible ones.

The architecture proposed in the 5GTango project [17] adds a verification and validation component to the network function virtualization reference architecture, allowing testing and verification of single network functions or entire network

services before they are deployed to production [18]. However, the endeavor does not offer a reusable modeling infrastructure, like our UML5G-SO profile does, which can be employed to build various diagrams for describing the structure and behavior of 5G orchestrators. Formal methods have also been used to verify VNFs and VNF chains against reachability and safety properties, in order to determine whether services are interfering, or are accessed by unauthorized users. Focusing on concrete VNF implementations, Marchetto et al. [19] propose a framework for verifying VNF chains automatically, by extracting verification models starting from a Java-based representation of a given VNF. VNF definitions are translated into formal verification models for different verification tools. Although this work offers a higher degree of assurance for VNF chains than ours, it is not focusing directly on service orchestration, also not on the set of QoS requirements that we consider in this paper. The closest to our work is the work of Papageorgiou et al. [20], in which the authors identify three main generic categories of 5G network slice models, namely, service-driven, resource-driven, and deployment-driven models. For these, the authors propose high-level models described in UML, representing the skeleton of the three generic categories. Specific detailed data models that belong to one of these categories should be possible to map to the respective skeleton. The core concept, the model structure, and prominent representative solutions for each of the designed models are explained and discussed. However, the work does not go beyond such modeling, nor provides any logic-based analysis of service orchestration, which we propose and demonstrate in this paper.

VII. Conclusions and Future Work

In this paper, we have proposed a UML-based modeling and analysis framework for 5G network slicing, named **UML5G-SO**, which enables one to model and analyze the feasible VNF placement and routing schemes respecting the VNF forwarding graphs of the slices. Our solution, unlike the ones available in literature, allows a 5G architect to model and analyze the orchestration scheme prior to implementation, by employing the user-friendly and industrially-adopted graphical modeling environment provided by UML. Our analysis framework not only provides the best possible VNF allocation and routing schemes, but also allows the engineers to eliminate the routing schemes that are not feasible. An important highlight of the framework is that no matter what service orchestration algorithm is chosen, the framework can model and analyze it, allowing a lot of flexibility and scalability options.

One of the limitations of our framework is that currently we deal with the system's design stage, hence considering only a static view of the system at each point of time. In the future, we want to expand our framework with modeling behavior that could be encountered at run time, such as flexible VNF sequencing, aperiodic events, possible component failures etc., and provide guarantees across all possible network configurations and extreme system traffic. We also plan to extend our UML5G-SO profile to serve the complete specification of a 5G system, for instance, by adding modeling support for physical and cloud-native network functions among others.

Acknowledgement

This work is supported by the EU Celtic Plus/Vinnova project, Health5G - Future eHealth powered by 5G, which is gratefully acknowledged.

References

[1] Guangyi Liu and Dajie Jiang. 5G: Vision and Requirements for Mobile Communication System towards Year 2020. *Chinese Journal of Engineering*, 2016:5974586, 2016.

[2] FCC Technological Advisory Council 5G IoT Working Group. 5g network slicing white paper. https://transition.fcc.gov/bureaus/oet/tac/tacdocs/reports/2018/5G-Network-Slicing-Whitepaper-Finalv80.pdf. Accessed: 2020-07-24.

[3] Tarik Taleb, Ibrahim Afolabi, Konstantinos Samdanis, and Faqir Zarrar Yousaf. On multi-domain network slicing orchestration architecture and federated resource control. *IEEE Network*, 33(5):242–252, 2019.

[4] 5G Transformer Project. http://5g-transformer.eu. Accessed: 2020-07-24.

[5] S Fichera, R Martínez, B Martini, M Gharbaoui, R Casellas, R Vilalta, R Muñoz, and P Castoldi. Latency-aware resource orchestration in sdn-based packet over optical flexi-grid transport networks. *Journal of Optical Communications and Networking*, 11(4):B83–B96, 2019.

[6] 5G evolution: 3GPP releases 16 & 17 overview. https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/5g-nr-evolution. Accessed: 2020-07-24.

[7] D4.1. Definition of service orchestration and federation algorithms, service monitoring algorithms. http://5g-transformer.eu/index.php/deliverables/. Accessed: 2020-07-24.

[8] Spyridon Vassilaras, Lazaros Gkatzikis, Nikolaos Liakopoulos, Ioannis N Stiakogiannakis, Meiyu Qi, Lei Shi, Liu Liu, Merouane Debbah, and Georgios S Paschos. The algorithmic aspects of network slicing. *IEEE Communications Magazine*, 55(8):112–119, 2017.

[9] Bruce Powel Douglass. *Real time UML: advances in the UML for real-time systems*. Addison-Wesley Professional, 2004.

[10] USE: UML-based Specification Environment. https://sourceforge.net/projects/useocl/. Accessed: 2020-07-24.

[11] Jordi Cabot and Martin Gogolla. Object constraint language (OCL): a definitive guide. In *International School on Formal Methods for the Design of Computer, Communication and Software Systems*, pages 58–90. Springer, 2012.

[12] S. Fichera, R. Martínez, B. Martini, M. Gharbaoui, R. Casellas, D. R. Vilalta, R. Muñoz, and P. Castoldi. Latency-aware resource orchestration in sdn-based packet over optical flexi-grid transport networks. *IEEE/OSA Journal of Optical Communications and Networking*, 11(4):B83–B96, 2019.

[13] Open Orchestrator Project and Linux Foundation (Open-O). https://www.open-o.org,2017. Accessed: 2020-07-24.

[14] Fraunhofer FOKU. OpenBaton. http://openbaton.github.io,2017. Accessed: 2020-07-24.

[15] SONATA project, "SONATA NFV: Agile Service Development and Orchestration in 5G Virtualized Networks". http://www.sonata-nfv.eu/. Accessed: 2020-07-24.

[16] W. Tavernier et al. D3.3. SONATA SDK final release, SONATA Project, Tech. Rep., 2017. http://sonata-nfv.eu/sites/default/files/sonata/public/content-files/deliverables/. Accessed: 2020-07-24.

[17] 5Gtango, "5Gtango project". https://www.5gtango.eu/. Accessed: 2020-07-24.

[18] E. Kapassa, M. Touloupou, P Stavrianos, G. Xylouris, and D. Kyriazis. Managing and optimizing quality of service in 5g environments across the complete sla lifecycle. *Advances in Science, Technology and Engineering Systems Journal (ASTESJ)*, Vol. 4(1):329–342, 2019.

[19] G. Marchetto, R. Sisto, M. Virgilio, and J. Yusupov. A Framework for User-Friendly Verification-Oriented VNF Modeling. In *Proc. of the IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 517–522. IEEE CS, 2017.

[20] A. Papageorgiou, A. Fernández-Fernández, S. Siddiqui, and G. Carrozzo. On 5g network slice modelling: Service-, resource-, or deployment-driven? *Computer Communications*, 149:232–240, 2020.