Customizable Function Chains: Managing Service Chain Variability in Hybrid NFV Networks

Hendrik Moens and Filip De Turck
Ghent University – iMinds, Department of Information Technology
Gaston Crommenlaan 8/201, B-9050 Gent, Belgium
e-mail: hendrik.moens@intec.ugent.be

***Abstract*—Using Network Functions Virtualization (NFV), network functions are virtualized and chained together to provide a network service. Often, a service chain can be allocated in multiple different ways, making use of different physical or virtual network functions, resulting in varying quality of service and deployment costs. In this article, we introduce customizable function chains, which model the services within a service chain and their variability, and present a management framework that can be used to deploy these services, making it possible to take service chain variability into account at runtime. We present and evaluate a model for resource allocation of network functions within NFV environments, focusing on a hybrid scenario where part of the network functions may be provided by dedicated physical hardware, and where part of the functions are provided using virtualized instances. We numerically study our approach using a small network provider scenario, on which a connectivity service is deployed, and analyze the benefit of the approach. In our evaluations, we find this approach results in a lower cost compared to an approach which does not support variability, both due to a reduction in the server use cost, and a reduction in failure cost when insufficient resources are present.**

***Index Terms*—Virtualization, NFV, Resource Allocation, Customization, SPLE**

I. Introduction

Using Network Functions Virtualization (NFV), Network Functions (NFs) can be migrated from costly hardware appliances to dynamically allocated virtualized instances deployed on generic servers using cloud technologies. This migration from Physical Network Functions (PNFs) to Virtual Network Functions (VNFs) increases network flexibility and scalability, as virtualized instances can be deployed and scaled on-demand using cloud technologies. A Service Function Chain (SFC) [1] describes the various NFs and how they interact to provide a complete network service.

In datacenters, high-capacity and high-speed networks are used to interconnect servers, making the specifics of the underlying network less important. In NFV deployments in networks outside of the datacenter, the importance of network constraints such as bandwidth and latency however increases, especially when NFs can be executed in multiple locations in the network, such as when multiple datacenters are present, or when physical devices spread throughout the network are part of the provided network service.

In practice, more expensive dedicated hardware often performs faster and more efficiently than virtualized instances, even though the latter are more flexible. As dedicated hardware is currently widely deployed, it is likely that hybrid NFV deployments will be common, where part of the NFs are provided by PNFs while others are provided using VNF instances. Using a mix of PNFs and VNFs also makes it possible to use "NFV bursts", where a base load is handled by efficient physical hardware, while variation in load is handled by dynamically instantiated VNFs. For these reasons, NFV management systems should support hybrid NFs that can be instantiated using both VNFs and PNFs.

When multiple PNFs or VNFs provide a similar functionality, there may be slight functional or qualitative differences. A software router or firewall may e.g. support newer protocol versions or more modern functionality than older, physical devices, but they may also be slower. In some cases, the functionality of different NFs may also overlap. It may e.g. be possible to configure a Deep Packet Inspection (DPI) NF to offer firewall functionality. For some SFCs these differences could be more constraining than for others, meaning they must always make use of specific NFs, while other SFC definitions could be more flexible, allowing for the use multiple different physical or virtual NFs with varying quality characteristics. This flexibility can be used to choose the most cost-efficient service chain configuration at runtime, making it useful to model these potential variations, and to take them into account during the resource allocation process. In addition, this approach also makes it possible to model a degraded, lower quality fallback version of the SFCs that can be used when service failures cannot be avoided due to network failures or capacity shortages.

To manage NFV services that support both PNFs and VNFs, and to model service chain variability, a network and service-aware NFV management system must be developed. This management system should allow for the existence of both physical and virtual NFs, and should also take into account SFC variability. The management system can then make use of this information to minimize costs resulting from resource utilization and service failures.

In this article, we introduce Customizable Function Chains (CFCs), an extension to SFCs that takes service chain variability into account, and define a management approach that is capable of deploying CFCs in hybrid NFV networks. To model CFC variability, we make use of Software Product Line Engineering (SPLE) [2] concepts. SPLE techniques are often used to develop highly customizable software, of which multiple variants can exist. Using this approach, the software is modeled as a collection of features, that can be included or excluded. By selecting and deselecting features, different software variants can be created. Thus, a CFC not only specifies the used NFs and their interconnections, but also a model describing the relations between the NFs. To allocate CFCs on a network, we present a Customizable Function Chain Placement (CFC-P) model, which we compare to a Service Function Chain Placement (SFC-P) model which does not take variability into account. We then evaluate both models in a simulated service provider network. The contribution of this article is threefold: (1) we describe how hybrid NFs, that can be realized using both VNFs and PNFs can be defined; (2) we describe how SPLE concepts can be applied to the specification and management of NFV function chains to increase their flexibility, resulting in CFCs; and (3) we formally define and evaluate models for allocating SFCs and CFCs using hybrid NFs in service provider networks.

In Section II, related work is discussed. In Section III we discuss an architecture for hybrid NFV resource management. Section IV describes how CFCs can be modeled and specified. We describe a model for CFC resource allocation in Section V. In Sections VI and VII we discuss the evaluation setup and results. Finally, in Section VIII we state our conclusions.

II. Related Work

Multiple recent works have focused on solving various (network) function placement problems in NFV networks. [3] presents a model for determining optimal gateway deployments in NFV-enabled mobile core networks. The proposed approach is however tailored specifically for this specific decision, and is aimed specifically at making decisions at the time of network dimensioning. In [4], a network function specification and placement approach is introduced. To specify SFCs, a context-free language is introduced which supports flexible definition of SFCs as a collection of modules. While the presented approach offers increased flexibility in the specification of SFCs, limited to re-ordering of NFs, it does not support hybrid NFs. [5] focuses on the allocation of virtual PDN gateways in carrier cloud networks, and focuses specifically on virtualized services, and not on physical or hybrid NFs. [6] focuses on the allocation of virtualized DPI engines within communication flows. [7] presents a dynamic NFV resource management system, capable of allocating VNFs within networks, and [8] introduces resource allocation strategies for managing NFV infrastructure, and focuses specifically on Virtual Machine (VM) allocation within datacenters, but neither take hybrid NFs or service adaptation into account. [9] introduces an Integer Linear Programming (ILP) formulation and heuristic algorithms for allocating SFCs defined as a topology of NFs within a network, but does not take hybrid NFs into account, and does not allow changing the NFs in the SFC and their topology based on the state of the network when the SFC is placed. In this article, we by contrast specifically focus on service chain resource allocation, where services can be deployed as a PNF, VNF or hybrid NF. In addition the deployed services can be adapted and deployed in multiple different ways in a controlled way using a feature modeling approach.

This article extends our previous work [10], where we introduced an approach for SFC resource allocation in hybrid NFV networks. While this approach makes it possible to support both PNFs and VNFs within NFV networks, this only works when both services are identical, limiting its applicability. In this article, we introduce CFCs, which make it possible to specify variability of service chains, solving this limitation. To manage CFCs, we introduce CFC-P, after which we compare the performance of CFC-P with that of SFC-P, which is based on the model from [10]. In addition, we present the architectural framework in which SFC-P and CFC-P can be used, and show how hybrid NFs and CFCs can be specified.

Resource allocation in NFV networks has some similarities to application placement approaches used within datacenters and clouds [11], specifically to network-aware application placement. Multiple publications [12]–[18] focus on either allocating collections of VMs, or on adding network-awareness to datacenter resource management algorithms. These works, however, focus specifically VM allocation within datacenter networks, meaning that often only datacenter-specific network topologies and hardware are considered. Therefore, these approaches are not suited for Wide Area Network (WAN) deployments, where a mix of general purpose hardware and dedicated hardware is present. This article contrasts with general datacenter management and application placement approaches by its focus on NFV resource allocation without any restrictions on the underlying network topologies, and its support for heterogeneous hardware. We define service chains that can contain NFs, which are managed by the service provider, and VMs, which are managed by clients. In addition, we also focus on how variability of the deployed service chains can be modeled and managed, making it possible to specify generic network services that can be implemented in multiple different ways at runtime.

SFC-P and CFC-P are similar to the problem of Virtual Network Embedding (VNE) in software defined networks [19]. VNE focuses on how virtual network requests, in the form of a collection of VMs and their interconnections can be deployed on physical networks. In this article we extend this embedding approach by defining a model that makes it possible to specify both VM requests and service requests, the latter resulting in service provider managed services that may be shared between multiple tenants. We also incorporate the notion of hybrid networks containing both physical devices offering services and virtualized services. The work presented in this article contrasts with existing VNE approaches as we also define the concept of service chain variability, making it possible to decide on the concrete service configurations at runtime to reduce management costs or to handle service degradations when insufficient resources are present.

The NFV Management and Orchestration (MANO) specification [20] supports the definition of both PNFs and VNFs, respectively using Virtualised Network Function Descriptor (VNFD) and Physical Network Function Descriptor (PNFD) elements in the VNF Forwarding Graph Descriptor (VNFFGD). While this approach supports the specification of services containing both PNFs and VNFs, there is no generic NF description element that defines the common functionality of network functions, meaning that no hybrid NFs can be specified. The specification also has no support for the specification of service chain customizability. In this article, we focus on how hybrid NFs and service chain variability can be modeled and managed, while aiming to define an approach which is compatible with the existing VNFD and PNFD elements.

TOSCA is a service modeling language which can be used to model and deploy SFCs [21]. By defining TOSCA templates, which can be instantiated using concrete types, limited support for variability can be provided as individual services may be offered using multiple concrete types. The standard however does not support more extensive customization, such as completely changing the topology of services in a service chain, and as a service specification language it requires a management system to determine optimal allocations. The work presented in this article is complementary to the goals of the TOSCA standard, focusing on determining optimal service configurations which could be represented using TOSCA and deployed using TOSCA deployment plans.

SPLE can be used to develop customizable Software-as-a-Service (SaaS) [22]–[26], but the focus of these works is generally on how this software can be developed and configured, and not on how it can be managed. In this article, we use SPLE principles to model runtime service chain variations. Using this approach it is possible to specify multiple valid service chain configurations using a single model, allowing the management system to choose the included NFs at runtime, and making it possible to reconfigure service chains to reduce costs or to manage service degradations.

In our previous work on the management of customizable SaaS [27]–[29], we developed an SPLE based approach for developing and managing customizable multi-tenant SaaS. The work focuses on managing small numbers of highly customizable applications within the context of a single cloud datacenter, which is achieved by splitting them up into components, and managing them using a Service-Oriented Architecture. This approach however focuses on the allocation of these components on homogeneous hosts within a single datacenter, where the underlying network is less important due to its high link capacities and low latencies. In this work, we similarly use an SPLE based approach to model CFC variability, which we use to support customizability within a single service chain, but we add support for network-awareness, which is needed in WAN networks. Furthermore, we generalize the approach making it possible to deploy both VMs and NFs on networks containing hosts and physical network devices.

III. NFV Resource Management Architecture

Figure 1 shows an overview of the NFV resource management system architecture, illustrating how the resource management algorithms fit within a larger management framework. The architecture consists of two layers: (1) a logically centralized CFC management and orchestration system, which is responsible for allocating and managing CFCs by determining where and how CFCs are allocated, after which it configures the NFs and routing system; and (2) a CFC infrastructure which runs the services. Both subsystems respond to different network requests: CFC traffic contains requests for the network service that must be routed

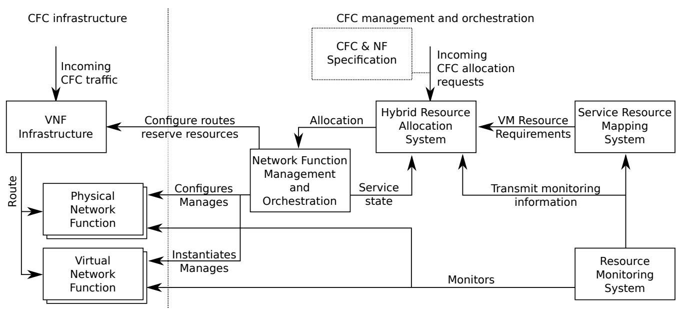

CFC infrastructure
CFC management and orchestration
CFC & NF Specification
Incoming CFC traffic
Incoming CFC allocation requests
VNF Infrastructure
Configure routes reserve resources
Allocation
Hybrid Resource Allocation System
VM Resource Requirements
Service Resource Mapping System
Network Function Management and Orchestration
Route
Physical Network Function
Configures Manages
Service state
Transmit monitoring information
Virtual Network Function
Instantiates Manages
Monitors
Resource Monitoring System

Fig. 1: A high level architectural overview of how a hybrid NFV resource management system can be constructed.

to the correct NFs, while CFC allocation requests are requests for the allocation and instantiation of a new network service that are handled by the CFC provisioning system.

CFC traffic is handled using the CFC infrastructure layer, which contains the VNFs, PNFs and the VNF infrastructure, on which these services are executed. The VNF infrastructure should provide functionality for configuring routes, such as those offered by existing SDN protocols. The CFC infrastructure layer corresponds to the NFV infrastructure and VNF subcomponents of the NFV reference architecture [30], but has been extended to support PNFs.

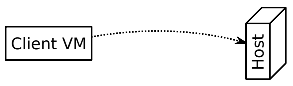
Incoming CFC allocation requests are handled by the CFC management and orchestration subsystem. This subsystem is responsible for determining how CFCs are deployed, and allocating the requested NFs using physical or virtual NF instances. Once the management system receives CFC allocation requests, they are handled by the following components:

1) The **hybrid resource allocation system** receives incoming CFC allocation requests and determines a correct allocation of NFs on the network. Depending on the CFC variability and the definition of the used NFs, the NFs may be allocated either using PNFs or VNFs.

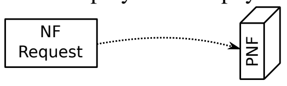
2) The **network function management and Orchestration subsystem** manages the PNFs, VNFs and infrastructure, and for deploying the computed resource allocation on this infrastructure. This component corresponds to the eponymous NFV reference architecture [30] subcomponent, but has been generalized to support managing both PNFs and VNFs.

3) The **service resource mapping system** is used to maintain a mapping between *server resources* and *service resources*. NFs provide resources specific to the provided service, such as e.g. the number of requests per second (rps) that the service can provide. While PNFs offer such service-specific resources, VM resource requirements are expressed in terms of server resources such as CPU and memory. To correctly provision resources, it is important that the management system is aware of the amount of server resources that are needed to provide a given amount of service resources. The service resource mapping system is responsible for storing an accurate mapping between server and service resources and updating it by monitoring the resource use of deployed VNFs.

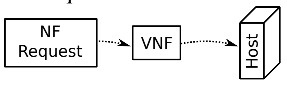
4) A **monitoring system** is needed to monitor the performance of the NFs and the network, ensuring the quality of the deployed CFCs is guaranteed, and enabling the hybrid resource



(a) Client VM deployment on physical nodes.



(b) NF request allocation on a PNF.



(c) NF request allocation on a VNF.

Fig. 2: Network asset allocation approaches.

TABLE I: The relation types used to structure feature models.

Relation	Definition
Mandatory(p, c) $p \leftrightarrow c$	If parent feature p is selected, child feature c must be included as well.
Optional(p, c) $c \rightarrow p$	The feature c may only be selected if p is selected.
Alternative$(p, \{c_1, c_2, \ldots, c_n\})$ $p \leftrightarrow c_1 \vee c_2 \vee \cdots \vee c_n$ $\forall i \neq j : \neg(c_i \wedge c_j)$	When p is selected, exactly one of the features c_i must be selected.
Or$(p, \{c_1, c_2, \ldots, c_n\})$ $p \leftrightarrow c_1 \vee c_2 \vee \cdots \vee c_n$	When p is selected, at least one of the features c_i must be selected.
Requires(f_1, f_2) $f_2 \rightarrow f_1$	For feature f_1 to be included, f_2 must be included as well.
Conflicts(f_1, f_2) $f_1 \rightarrow \neg f_2$	If f_1 is included, f_2 must not be included.

allocation system to re-allocate resources when changes in resource demand or other problems are detected.

IV. Modeling CFCs and NFs

A SFC defines a request for a complete network service which may be composed out of multiple NFs, and defines a service graph which shows how the NFs interconnect. To allow SFCs to define a large variety of network services, we assume that a client requesting it may also provide VMs that are part of the service chain. These client VMs must then be allocated on a host as part of the SFC allocation process. An SFC therefore contains a collection of *network asset requests* which can either be for VMs, or for NFs. These NF requests should then in turn be allocated using either a PNF or a VNF.

The key difference between client VM requests and VNF requests is that a VM that is part of an SFC is managed by the client who requests the service chain, appearing as a black box to the provider, while a VM that provides a VNF is provided and managed by the service provider, making it possible to share this VM between multiple clients who do not need to be aware of its existence. The provider-hosted VNFs can therefore be shared between multiple clients. Requested network assets may be allocated in three different ways depending on their type (illustrated in Figure 2: (1) client VMs must be deployed on physical server nodes, (2) NF requests can be allocated on PNF nodes, and (3) NF requests can be allocated on a VNF instance which is itself allocated on a physical server.

SFCs can contain hybrid NFs, but small differences between NFs may impact the feasibility and quality of some SFC

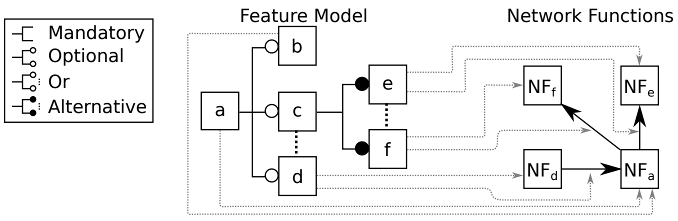


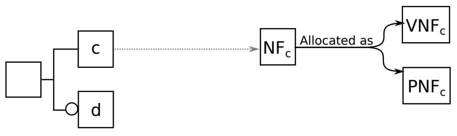
Fig. 3: An illustration of a feature model containing six features. The features a, d, e, and f are physical features associated with NFs. Including these features causes the corresponding NF to become active. Dotted arrows indicate the resource impact on NFs and links between NFs caused by the inclusion of features, while filled arrows show how traffic is forwarded between NFs if the link is included.

configurations. To manage SFC variation, we make use of SPLE principles. SPLE [2] is used to manage variability of applications and services, and can be used to define a large collection of software variants based on a single, shared codebase. To model the functionality that may be included and excluded in services, the concept of a *feature* is used. A feature represents a single functionality which can be either included or excluded in a variant of the application or service. All of the features that the service can include are contained in a *feature model*, which defines the relations between features, and which is used to determine which feature combinations result in a feasible service configuration.

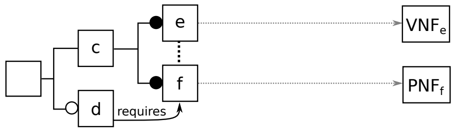
A CFC extends the SFC concept by defining a feature model instead of a service graph, making it possible to specify multiple alternative versions of a network service. The service chain itself is then composed by including and excluding features while taking their relations into account. This makes it possible to create multiple related service chains with differing functionalities using a common model. The inclusion of one feature may result in the inclusion or exclusion of other features, resulting in one or more possible valid configurations where every feature is either included or excluded. In SPLE, feature models are traditionally specified hierarchically using a limited set of relations to make it easier to define, manage and reason using them, which is why we also use a hierarchical approach in this article. Four hierarchical relation types, **Mandatory**, **Optional**, **Alternative**, and **Or**, and two non hierarchical relation types, **Requires** and **Conflicts**, are used[1]. These relations are defined in Table I, and an example feature model is shown in Figure 3.

When modeling CFCs using a feature model, we assume that every requested asset is associated with a feature within the feature model. The inclusion of the feature within the model then implies the inclusion of the associated asset. The feature model can then be used to determine the valid NF and VM combinations, and their network and resource demands. The inclusion of a feature may 1) result in the inclusion of specific asset requests, 2) impact the resource load of NFs and VMs, 3) result in an increase of the network demand between

[1]Note that the presented approach is not strictly dependent on these 6 chosen relations, as relations within the model may be expressed using any functionally complete set of propositional calculus operators.



(a) A feature implemented using a hybrid NF. The CFC model is unaware of the different implementation methods.



(b) A feature implemented by explicitly linking features within the model to a VNF or a PNF using an **Alternative** relation, making the CFC aware of the existence of different NF implementations.

Fig. 4: An illustration of how hybrid NFV scenarios can be modeled using CFCs.

TABLE II: Model parameters: network and service definition.

Symbol	Description
G	The graph $G = (N, E)$ representing the network.
N	The nodes within the network. This consists of a set of computational nodes N^c and a set of PNF nodes N^p.
E	The edges within the network.
E_n^{in}	The edges that are incoming in node $n \in N$.
E_n^{out}	The edges that are outgoing from node $n \in N$.
$C(e)$	The bandwidth capacity of an edge $e \in E$.
$\mathcal{NF}$	The collection of all NFs that exist within the model.
Γ^n	The resource types that are available on a node $n \in N$, or the resource types provided by a NF $n \in \mathcal{NF}$.
$C^\gamma(n)$	The resource capacity of a node or NF n. For computational nodes, $\gamma \in \Gamma^{VM}$. For PNF nodes offering an NF type s, $\gamma \in \Gamma^s$. A VNF of an NF type s provides $C^\gamma(n)$ resources of types $\gamma \in \Gamma^s$.
NC_n	The cost of using a node $n \in N$ during resource allocation.
NC_n^γ	The cost of using resources of type $\gamma \in \Gamma^n$ on node $n \in N$.
L_n^r	Node latency when a request r is executed on node n.
L_e	Edge latency of an edge e.

NFs and VMs, or 4) may even result in the inclusion or exclusion of quality constraints such as latency constraints. Within the feature model we discern two types of features: physical features and virtual features. A physical feature is directly associated with a specific asset request. Including the physical feature will therefore cause the associated NF or VM to be instantiated. Virtual features are not directly associated with a NF, but may still impact the configuration of NFs and VMs, and therefore their resource demand. They may also be used to add structure to the feature model. Figure 3 shows an illustrative feature model containing six features that map to four different NFs. Any SFC can be trivially converted into a CFC by creating a feature model containing all asset requests and only mandatory relations, making CFCs a superset of SFCs.

A hybrid NF can be specified within a CFC using two different approaches. Either a single NF, which can be allocated on either a VNF or on a PNF instance, can be defined. Alternatively, separate VNF and PNF instances may be defined, which are explicitly linked to the feature model using an **Alternative** relation. Using the latter model-based approach, no overarching hybrid NF is needed. Both approaches are illustrated in Figure 4. These approaches can be used to support a large number of different VNFs, such as when multiple service versions are deployed, and PNFs, such as when hardware from different vendors or different hardware versions are deployed, making it possible to represent multiple software versions and hardware variations. While both approaches can be used to represent hybrid NFs, they have different advantages and disadvantages. The feature model based approach ensures the CFC is aware of the various hardware and software NF implementations, increasing the model complexity, and consequently the number of possible configurations. This approach also makes it possible to define features that depend on a specific NF implementation. Using hybrid NFs, the complexity of the feature model decreases, but the model itself does not contain any information pertaining to the various NF implementations. Consequently, hybrid NFs are preferred when the physical and virtual implementations of the NF behave similarly and offer identical functionality, while the more complex feature model based approach may be needed otherwise.

While arbitrarily large feature models can be supported, this does increase computational and management complexity. In practice, only a limited number of software and hardware versions will be deployed within the network, limiting the number of variants that must be taken into account, therefore limiting the size of the required feature model. To manage larger and more complex feature models, automated management tools can be used that automatically convert flexible development-time models, which may support additional relation types, into larger, more rigid runtime models can be deployed at runtime. We have previously discussed how a similar automated transformation can be achieved for customizable component-based applications without network-awareness in [31].

V. Function Chain Resource Allocation

The CFC-P model addresses both how network services are composed (i.e. which NFs are used and how are they interconnected) and the service allocation (i.e. where in the network are the NFs deployed). We construct the CFC-P model in two steps. First, we describe an approach for allocating SFCs in hybrid NFV networks, without taking service variation into account. This model only solves the service allocation problem without addressing the service composition problem. Subsequently, we extend the SFC-P model and add awareness of service chain customizability by adding a feature model, allowing it to solve the service composition problem in addition of the service allocation problem. The CFC-P model builds on the SFC-P model by redefining some of the variables and adding additional constraints.

A. General model parameters

Both the SFC-P and CFC-P models require information about the topology of the network, and information about the PNFs and server nodes. These common parameters are shown in Table II. The network is represented as a graph $G = (N, E)$, consisting of a collection of nodes N that represent physical network nodes and edges E between these nodes. We assume that these edges can be either bidirectional

TABLE III: Model parameters: SFC-P.

Symbol	Description
$\mathcal{SFC}$	The SFCs that must be allocated on the network.
$R(C)$	The asset requests that are part of service chain $C \in \mathcal{SFC}$. $R^{VM}(C)$ contains only the VM requests, while $R^{NF}(C)$ contains the NF requests. $R(C) = R^{VM}(C) \cup R^{NF}(C)$.
$\mathcal{R}$	A collection containing all of the asset requests of all service chains in $\mathcal{SFC}$. $\mathcal{R}^{VM}$ and $\mathcal{R}^{NF}$ contain only the VM and NF requests respectively.
$D^\gamma(r)$	The resources of type γ needed for a request r. This request can either be an NF request or a VM request.
$D(r_1, r_2)$	The network demand between asset requests r_1 and r_2. The collection D contains all pairs of asset requests between which there is network demand.
FC_C	The cost of failing to allocate any of the services of a service chain $C \in \mathcal{SFC}$.

or unidirectional. Incoming and outgoing edges from a node $n \in N$ are represented by E_n^{in} and E_n^{out} respectively, and every edge e has a capacity $C(e)$. A node can be any device in the network, such as network switches, computational nodes on which VMs can be deployed, or PNFs such as e.g. routers, hardware firewalls, or access points. We make a distinction between computational nodes, contained in N^c, and PNF nodes N^p that offer a specific NF. A set, $\mathcal{NF}$ should contains all of the NFs that can be allocated.

Nodes $n \in N$ offer multiple types of resources, which are contained in the set Γ^n. The types of resources contained in this set are dependent on the type of the node: if a node is a computational node it will offer server resources, while if it is a PNF node it will offer service-specific resources. Similarly, for any NF $s \in \mathcal{NF}$ the offered service-specific resource types are contained in the set Γ^s. Every node $n \in N$ has a resource capacity $C^\gamma(n)$ for each of its offered resource types $\gamma \in \Gamma^n$. For every node, a usage cost NC_n, and a node resource use cost NC_n^γ can be specified. The former cost is incurred whenever a node n is used in an allocation, while the latter is incurred for every resource used on a node (e.g. a cost incurred for every CPU core used in a cloud node). These costs represent the cost of using the resource, and represent the cost of using cloud resources, energy use costs, licensing costs, opportunity costs, and possibly other costs. As some of these factors may vary over time, it is possible for these costs to vary over time, but it is assumed that they are constant during a single invocation of the resource allocation.

B. SFC-P

The required parameters to specify SFCs are shown in Table III. A collection $\mathcal{SFC}$ contains all service chains that must be allocated within the network. A service chain $C \in \mathcal{SFC}$ consists of a collection of asset requests, where the requested asset can either be an NF or a VM. The NF and VM requests of an SFC C are contained in the $R^{NF}(C)$ and $R^{VM}(C)$ collections respectively. All asset requests in an SFC are grouped in the collection $R(C) = R^{VM}(C) \cup R^{NF}(C)$. The collection $\mathcal{R}$ contains all asset requests in the model, aggregated over all of the service chains in $\mathcal{SFC}$. The resource demand of an asset request r is represented by $D^\gamma(r)$, and is specified for all resources $\gamma \in \Gamma^r$ that are provided by the requested asset. The network demand between requested VMs and NFs is represented by $D(r_1, r_2)$ for any two asset requests $(r_1, r_2) \in \mathcal{R}^2$ that are part of the SFC. When an SFC C can not be allocated because there is insufficient network or server capacity it may not be allocated. In this case, a service failure cost FC_C is incurred.

1) Resource management constraints: We define a binary decision variable, M_n^r, that determines whether an asset request $r \in \mathcal{R}$ is allocated on a node $n \in N$. If a service request may not be allocated on a given node, $M_n^r = 0$ is added as a constraint, making it possible to restrict the nodes on which a service can be executed. An integer decision variable IC_n^s specifies the number of times that an NF $s \in \mathcal{NF}$ is instantiated on a computational node $n \in N^c$. We use an integer variable as we assume that it is possible for there to be multiple instances of the same VNF on a single physical node. If this is not desired, an upper bound of 1 can be specified for the IC_n^s variable preventing this. Using these decision variables, and the parameters specified previously, a capacity constraint, shown in Equation (1), can be defined. This constraint specifies that the total resource use $U(n, \gamma)$ for all nodes $n \in N$ and resources $\gamma \in \Gamma^n$ should be less than the resource capacity $C^\gamma(n)$ of the node. The total resource use is computed as shown in Equation (2), and is specified separately for server nodes and physical nodes. For server nodes, the total amount of used resources is composed out of the resources used for handling VM requests, represented as U^{VM} and shown in Equation (3), and out of the resources used to allocate VNFs, represented as U^{VNF}, shown in Equation (4). For physical nodes, only the resources used by the allocated NFs need to be taken into account, as shown in Equation (5).

$$\forall n \in N : \forall \gamma \in \Gamma^n : U(n, \gamma) \leq C^\gamma(n) \tag{1}$$

$$U(n, \gamma) = \begin{cases} U^{VM}(n, \gamma) + U^{VNF}(n, \gamma), & \text{if } n \in N^c. \\ U^{PNF}(n, \gamma), & \text{if } n \in N^p. \end{cases} \tag{2}$$

$$U^{VM}(n, \gamma) = \sum_{r \in \mathcal{R}^{VM}} M_n^r \times D^\gamma(r) \tag{3}$$

$$U^{VNF}(n, \gamma) = \sum_{s \in \mathcal{NF}} IC_n^s \times D^\gamma(s) \tag{4}$$

$$U^{PNF}(n, \gamma) = \sum_{r \in \mathcal{R}^{NF}} M_n^r \times D^\gamma(r) \tag{5}$$

It is important to ensure that the IC_n^s decision variables take on the correct values. This is achieved using Equation (6), which ensures that the number of instances on a node, IC_n^s, offers sufficient service resources for the VNFs that are allocated on the node. This equation uses the total available service resources $T^\gamma(s, n)$ and the total needed service resources $N^\gamma(s, n)$ on the node. The expressions for $T^\gamma(s, n)$ and $N^\gamma(s, n)$ are shown in Equations (7) and (8) respectively.

$$\forall n \in N : \forall s \in \mathcal{NF} : \forall \gamma \in \Gamma^s : T^\gamma(s, n) \geq N^\gamma(s, n) \tag{6}$$

$$T^\gamma(s, n) = IC_n^s \times C^\gamma(s) \tag{7}$$

$$N^\gamma(s, n) = \sum_{r \in \mathcal{R}^{NF}} M_n^r \times D^\gamma(r) \tag{8}$$

When there are insufficient resources, such as when there is not enough network capacity to guarantee the required network throughput or when there is limited node capacity, it may be possible that not every SFC will be allocated. We define the binary decision variable Φ^C which takes on value 1 if the service chain $C \in \mathcal{SFC}$ is allocated and 0 otherwise. Equation (9) is used to ensure that a function chain is allocated when all of its requests are.

$$\forall C \in \mathcal{SFC} : \forall r \in R(C) : \sum_{n \in N} M_n^r = \Phi^C \tag{9}$$

2) Network constraints: To add network-awareness to the model, the flow between two requests $(r_1, r_2) \in \mathcal{R}^2$ over the edges $e \in E$ of the network is modeled using a collection of binary flow decision variables $F(e, r_1, r_2)$. Iff $F(e, r_1, r_2) = 1$, the edge e is used for the flow (r_1, r_2). These flow variables are subject to a flow conservation constraint, shown in Equation (10). For all nodes except the source and sink nodes, the incoming flow must equal the outgoing flow. For the source node, the flow must exceed the out flow, while for the sink node the opposite holds.

$$\begin{aligned} OUT(n, r_1, r_2) &= M_n^{r_2} + \sum_{e \in E_n^{out}} F(e, r_1, r_2) \\ IN(n, r_1, r_2) &= M_n^{r_1} + \sum_{e \in E_n^{in}} F(e, r_1, r_2) \\ \forall (r_1, r_2) \in D &: \forall n \in N : \\ &OUT(e, r_1, r_2) = IN(e, r_1, r_2) \end{aligned} \tag{10}$$

Two additional constraints, Equations (11) and (12), are added to make sure there is no incoming flow in source nodes or outgoing flow in sink nodes. These two constraints ensure that, when the node n is a source (sink) which is the case when $M_n^{r_2}$ ($M_n^{r_1}$) takes on value 1, the incoming (outgoing) flow must be 0. Finally, a capacity constraint, expressed in Equation (13), is needed to ensure that the total flow over an edge does not exceed the available edge capacity.

$$\forall (r_1, r_2) \in D\colon \forall n \in N\colon M_n^{r_2} + \sum_{e \in E_n^{out}} F(e, r_1, r_2) \leq 1 \tag{11}$$

$$\forall (r_1, r_2) \in D\colon \forall n \in N\colon M_n^{r_1} + \sum_{e \in E_n^{in}} F(e, r_1, r_2) \leq 1 \tag{12}$$

$$\forall e \in E\colon \sum_{(r_1, r_2) \in D} F(e, r_1, r_2) \times D(r_1, r_2) \leq C(e) \tag{13}$$

A second collection of network constraints result from network latency requirements. The latency between two asset requests r_1 and r_2 is defined by the latency incurred by the processing of the first asset request, referred to as the node latency, the node latency of the second request, and the network latency incurred by the communication between the two assets. Formally, node latency is expressed in Equation (14) and network latency is shown in Equation (15). These expressions depend on L_n^r, which represents the latency incurred when an asset request r is executed on a node n, and L_e, the latency of a network edge e. While L_n^r and L_e in practice depend on the load on the node and network link respectively, and could be specified in function of this resource utilization, this would greatly complicate the model specification and result in the model becoming quadratic. Therefore, we choose to provide L_n^r and L_e as input variables that represent estimated latencies based on the maximum capacity of the resource, erring on the side of caution when latency is critical. Combining both latency types, a service chain latency, shown in Equation (16), can be determined. When needed, this concept can be further generalized to support longer chains, containing multiple service chains as illustrated in Equation (17).

$$\mathcal{L}(r) = \sum_{n \in N} M_n^r \times L_n^r \tag{14}$$

$$\mathcal{L}^{net}(r_1, r_2) = \sum_{e \in E} F(e, r_1, r_2) \times L_e \tag{15}$$

$$\mathcal{L}(r_1, r_2) = \mathcal{L}(r_1) + \mathcal{L}^{net}(r_1, r_2) + \mathcal{L}(r_2) \tag{16}$$

$$\begin{aligned} \mathcal{L}(r_1, r_2, \ldots, r_n) = \mathcal{L}(r_1) + \mathcal{L}^{net}(r_1, r_2) + \mathcal{L}(r_2) + \\ \ldots + \mathcal{L}(r_{n-1}) + \mathcal{L}^{net}(r_{n-1}, r_n) + \mathcal{L}(r_n) \end{aligned} \tag{17}$$

These expressions can be used to add additional constraints to the model. This allows for the expression of latency constraints in the form of $\mathcal{L}(r_1, \ldots, r_n) < L^{max}$, with L^{max} representing a latency limit. One or more of these limitations can then be defined for any part of the service chain.

3) Optimization objective: We use an approach where the successful allocation of SFCs is subject to stringent constraints. Successful SFC allocation implies the satisfaction of these constraints, and therefore offers the needed quality guarantees for applications (i.e. capacity and latency guarantees). As it may not always be feasible for every SFC to be allocated successfully, it must be possible for some allocations to fail, which must result in a penalty that is taken into account during the optimization. The objective of the model is to minimize the total cost of the SFC allocations. This cost is composed of a resource utilization cost, CR, and a management cost associated with the failure to provision resources for an SFC, CF. This failure cost is incurred when the optimization fails to find an allocation where all required quality demands can be met, e.g. when the network capacity constraint can not be satisfied resulting in insufficient network throughput, when there is insufficient node capacity for supporting the network service, or when the deployed service would result in an unacceptable latency. Combining both costs, the optimization objective can be specified as shown in Equation (18).

$$\min\left(CR + CF\right) \tag{18}$$

$$CR = \sum_{n \in N} U_n \times NC_n + \sum_{\gamma \in \Gamma^n} U(n, \gamma) \times NC_n^{\gamma} \tag{19}$$

$$\forall n \in N : \forall r \in \mathcal{R} : U_n \geq M_n^r \tag{20}$$

$$CF = \sum_{C \in \mathcal{SFC}} FC_C \times (1 - \Phi^C) \tag{21}$$

The resource utilization cost consists of a static node use cost and a linear node resource use cost. The static node use cost is incurred whenever a node is used to allocate any asset. Therefore, this cost can be e.g. utilized to represent the energy cost of instantiating a node. By contrast, the linear node resource cost incurs a cost for every resource used on the node. This cost therefore increases as the resource utilization on a node increases, and may e.g. be used to incur a cost for every

TABLE IV: Model parameters: CFC-P.

Symbol	Description
$\mathcal{CFC}$	The CFCs that must be allocated on the network.
$\mathcal{F}_C$	The feature model of a service chain $C \in \mathcal{CFC}$. This model contains a collection of features $f \in \mathcal{F}_C$ and also defines the relations between these features.
$\mathcal{F}_C^{phy}$	This collection contains the physical features in the feature model of service chain $C \in \mathcal{CFC}$, i.e. the features that are directly linked to an asset request.
$FI^{\gamma}_{f_1}(f_2)$	The feature resource impact $FI^{\gamma}_{f_1}(f_2)$ defines the impact on the resource requirements for feature f_2 when feature f_1 is included.
$FI_{f_1}(f_s, f_t)$	The feature network impact $FI_{f_1}(f_s, f_t)$ defines the impact on the network demand between source associated with feature f_s and the sink linked with feature f_t.
FC_C	The cost of failing to allocate the entire service chain $C \in \mathcal{CFC}$.
FC^f_C	The cost of failing to provide a single feature $f \in \mathcal{F}_C$ that is part of a service chain $C \in \mathcal{CFC}$.

CPU core used in a data centre. The complete utilization cost computation is expressed in Equation (19). In this definition, the binary decision variable U_n is used to determine whether a node n is used. Equation (20) ensures that U_n takes on value 1 as soon as anything is allocated on the node n.

When insufficient network or hardware resources are present to accommodate all requests, some SFCs will fail to be allocated. Such service interruptions should be prevented, but if this is not possible, the cost of these failures should be minimized. Therefore, a service failure cost is associated with every SFC, and when it is not allocated this cost is incurred. Equation (21) shows how the total service failure cost can be computed. Note that this cost does not have to correspond to an exact monetary cost, but may be defined by a systems operator or the management system to achieve a desirable system state (e.g. to achieve service prioritization).

C. CFC-P

The general concepts of SFC-P can be extended to support CFCs. The parameters specific to the CFC-P are shown in Table IV. The general parameters specifying network and service information, listed in Table II, are still required. As the CFC-P extends the SFC-P, all of the SFC-P parameters from Table III must also be defined, but they are no longer inputs to the model as they are either transformed into decision variables, or alternatively the associated values are determined based on the CFC-P parameters.

A collection of CFCs is provided in the set $\mathcal{CFC}$. Every service chain $C \in \mathcal{CFC}$ has an associated feature model $\mathcal{F}_C$. This feature model contains a collection of features, f, and defines the relations between these features. The resource demand for a feature, and the network demand between features is dependent on which features are included. When a feature f_1 is included, it may impact the resource demand for another feature f_2 for a resource type γ using a given feature resource impact $FI^{\gamma}_{f_1}(f_2)$. The feature may also impact the network demand between two features f_s and f_t, resulting in a feature network impact $FI_{f_1}(f_s, f_t)$. When feature inclusion results in a non-zero impact on a feature or network link, this implies that the feature and link are part of the final service chain.

TABLE V: The constraints associated with the various feature model relations.

Relation	Constraint
$\mathbf{HardSelected}(f)$	$\Phi^C_f = \Phi^C$
$\mathbf{Mandatory}(f, c)$	$\Phi^C_f = \Phi^C_c$
$\mathbf{Optional}(f, c)$	$\Phi^C_f \geq \Phi^C_c$
$\mathbf{Alternative}(f, \{c_1, c_2, \dots, c_n\})$	$\Phi^C_f = \Phi^C_{c_1} + \Phi^C_{c_2} + \dots + \Phi^C_{c_n}$
$\mathbf{Or}(f, \{c_1, c_2, \dots, c_n\})$	$\Phi^C_f \leq \Phi^C_{c_1} + \Phi^C_{c_2} + \dots + \Phi^C_{c_n}$
	$\forall i = 0 \dots n : \Phi^C_f \geq \Phi^C_{c_i}$
$\mathbf{Conflicts}(f_1, f_2)$	$\Phi^C_{f_1} \leq 1 - \Phi^C_{f_2}$
$\mathbf{Requires}(f_1, f_2)$	$\Phi^C_{f_1} \leq \Phi^C_{f_2}$

To extend the SFC-P model, all of its parameters must be defined. First, feature models must be linked to asset requests. This is done by linking physical features to asset requests, as explained in Section IV and illustrated in Figure 3. Physical features, from the set $\mathcal{F}_C^{phy}$, are linked to asset requests. For a physical feature $f \in \mathcal{F}_C^{phy}$, the matching asset request is defined as $\hat{f}$. Using this mapping, the collection of all asset requests can be determined: $R(C) = \cup_{f \in \mathcal{F}_C^{phy}} \hat{f}$. The demand $D^{\gamma}(r)$ for requests and the network demand between these requests $D(r_1, r_2)$ are redefined as decision variables. The network demand decision variables are only defined if any feature exists that impacts the network between both requests. Finally, the chain fail cost FC_C is redefined as the cost of completely failing the service chain, while a new feature fail cost FC^f_C represents the cost of failing to provide a specific feature f.

For every CFC, a valid feature configuration must be determined. This is done by adding constraints for all relations that are contained in the feature model. These constraints make use of binary decision variables Φ^C_f, which are used to express whether the feature f is included in service chain C. Table V shows how all relation types can be modeled. The $\mathbf{HardSelected}(f)$ relation is used to express that the CFC can only be included if the feature f is included. This is e.g. always used for the root of the feature model. When a feature should be included, but may fail in some cases, a feature fail cost should be assigned to it. As shown in Equation (22), CFC features may only be included when the CFC itself is included.

$$\forall C \in \mathcal{CFC} : \forall f \in \mathcal{F}_C : \Phi^C \geq \Phi^C_f \tag{22}$$

When physical features are included in a CFC, the linked asset request must be included, as shown in Equation (23). The demand for these asset requests depends on the features that are included. Equation (24) shows how the feature resource impact is used to compute the demand for a given feature.

$$\forall C \in \mathcal{CFC} : \forall f \in \mathcal{F}_C^{phy} : \sum_{n \in N} M^{\hat{f}}_n = \Phi^C_f \tag{23}$$

$$\forall C \in \mathcal{CFC} : \forall f \in \mathcal{F}_C^{phy} : \forall \gamma \in \Gamma^{\hat{f}} : \\ D^{\gamma}(\hat{f}) = \sum_{f' \in \mathcal{F}_C} \Phi^C_{f'} \times FI^{\gamma}_{f'}(f) \tag{24}$$

1) Network constraints: The network demand, like the demand for asset requests, is dependent on the features included in the CFC. Equation (25) shows how the network demand can be computed using the feature inclusion variables.

$$\forall C \in \mathcal{CFC} : \forall (f_s, f_t) \in \left(\mathcal{F}_C^{phy}\right)^2 : \forall \gamma \in \Gamma^{\hat{f}} :$$
$$D(\hat{f}_s, \hat{f}_t) = \sum_{f \in \mathcal{F}_C} \Phi_f^C \times FI(\hat{f}_s, \hat{f}_t) \quad (25)$$

By using a feature model, there can be multiple possible feature configurations for a CFC, resulting multiple possible sets of included asset requests. This complicates the SFC-P flow conservation constraint (Equation (10)) as it depends on the source and sink nodes being both included or both excluded. To resolve this, a binary decision variable $FA(r_1, r_2)$ is specified which expresses whether a flow between two asset requests r_1 and r_2 is active. Equation (26) ensures the FA variables take on the correct value. Using these variables, a replacement flow conservation constraint must be redefined using Equations (27) and (28).

$$\forall C \in \mathcal{CFC} : \forall f \in \mathcal{F}_C^{phy} : \forall (s,t) \in (\mathcal{F}_C)^2 :$$
$$FI_f(f_s, f_t) \neq 0 \rightarrow \Phi_f^C \leq FA(\hat{f}_s, \hat{f}_t) \quad (26)$$

$$\forall (r_1, r_2) \in D : \forall n \in N :$$
$$OUT(n, r_1, r_2) \leq IN(n, r_1, r_2) + (1 - FA(r_1, r_2)) \quad (27)$$
$$IN(n, r_1, r_2) \leq OUT(n, r_1, r_2) + (1 - FA(r_1, r_2)) \quad (28)$$

Like SFCs, CFCs can also be subject to latency limitations. While these limitations can be expressed as separate constraints, independent from the feature model, these limitations can also be incorporated as part of the feature model. This can be achieved by defining a latency constraint which is only triggered when a given feature is present. The advantage of this approach is that quality differentiation with regards to latency becomes possible, and that failing to succeed in a stringent latency limit does not always necessarily lead to the failure of the entire network service. When the inclusion of a feature f limits the latency $\mathcal{L}(\hat{f}_1, \hat{f}_2, ..., \hat{f}_n)$ to a given limit L^{max} this can be expressed as shown in Equation (29). In this expression $\mathbf{M}$ represents a very large number, which must at least be larger than the sum of all latencies present within the network.

$$\mathcal{L}(\hat{f}_1, \hat{f}_2, \ldots, \hat{f}_n) \leq L^{max} + (1 - \Phi_f^C) \times \mathbf{M} \quad (29)$$

2) Optimization objective: The feature models provided by CFCs allow for more flexible network service specifications. While in SFC-P a service chain is either allocated in its entirety or not at all, CFC-P allows to make a distinction between individual feature failure and complete service failure, where the former results in a lower cost than the latter, making it possible to provide a degraded service chain when there is insufficient capacity to allocate the complete service. Therefore, an additional cost of feature failure *CFF* is defined. This cost per feature is expressed in Equation (30) and is used in the CFC-P objective function composed out of the resource utilization cost *CR*, and the failure costs *CF* and *CFF*, shown in Equation (31).

$$CFF = \sum_{C \in \mathcal{CFC}} \sum_{f \in \mathcal{F}_C} FC_C^f \times (1 - \Phi_f^C) \quad (30)$$

$$\min (CR + CF + CFF) \quad (31)$$

VI. Evaluation Setup

Both the SFC-P and CFC-P models were implemented as an ILP in Scala [32] using the IBM Ilog CPLEX ILP solver [33], which solves the ILP using simplex and branch and bound algorithms. Note that the CFC-P model as specified in this article contains multiplications of binary decision variables with binary and continuous decision variables, making the model as presented quadratic instead of linear. We linearized these operations by replacing them with logically equivalent linear equations that use intermediary decision variables, ensuring both models are implemented as pure ILPs. As the SFC-P and CFC-P models allow service chain failures, a trivial feasible solution where every service fails can always be found. This ensures the algorithms will always result in a feasible solution.

We compare the two optimal ILP-based algorithms implementing the SFC-P and CFC-P models, and also compare their performance with time-limited versions of the ILPs. While during a normal execution, CPLEX will continue until a provably optimal solution has been found using simplex and branch and bound algorithms, the time-limited CPLEX invocations stop the execution of the optimizer after a given execution duration, and return the highest quality feasible result. This results in a suboptimal result, but guarantees a given execution duration, making it more feasible to use these algorithms within a dynamic management system.

We consider a connectivity service where a source and sink node are interconnected. This connection may be direct, but optionally, the packets may be intercepted and analyzed. This analysis can be achieved using a firewall, or alternatively using a DPI service. As a DPI service requires more computational resources, a sampled DPI service where a fraction of the requests are analyzed using DPI and the other packets are analyzed using a regular firewall can be supported as well.

A feature model for the connectivity CFC is shown in Figure 5, together with the associated services. Figure 6 shows the alternative CFC deployments that this feature model results in. Resource demand and service demand are represented by D^{net} and D^s respectively, and can vary for different applications. Table VI shows the resource and network impacts that features result in. Note that, while the `DPI` feature results in the inclusion of the DPI service, the load on this service depends on how the service is instantiated (i.e. as a full DPI service or as a sampled DPI service). The structure of this model makes it possible to create *open variation points* [22], that leave some variability decisions undecided when the CFC is specified, allowing the management system to decide at runtime how the service chains are deployed. A service chain requiring only sampled DPI can e.g. be implemented using either the `FullDPI` feature or the `SampledDPI` feature when only the `DPI` feature is selected. This can potentially result in a reduction of the number of instances when there

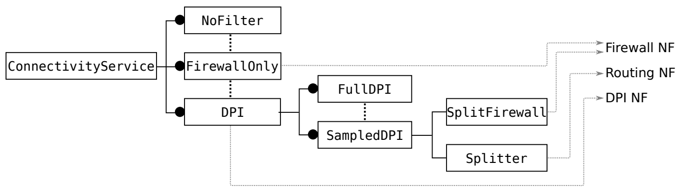


Fig. 5: Connectivity service feature model and associated NFs.

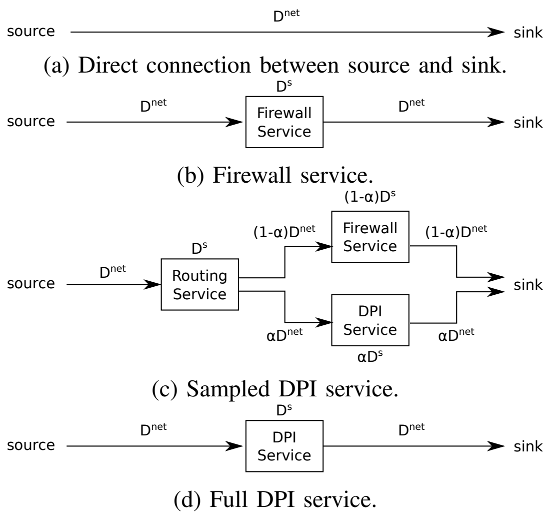


Fig. 6: An illustration of the various possible connectivity service CFC configurations. D^{net} represents the network demand while D^s represents the service demand. α represents the fraction of requests that must be sent to the DPI service.

is sufficient capacity on existing DPI NFs, as then no routing and firewall NF instances are needed.

We use multiple instances of the connectivity CFC within the evaluation scenario. Five different CFC types are defined. These types differ in the selection and exclusion of specific features, and are shown in Table VII. Selected features should be included in the CFC. If a valid feature model configuration exists where the selected features are not included, this model

TABLE VI: Resource and network impacts in the connectivity service scenario. D^s represents service demand (in rps), D^{net} represents network demand, and α represents the share of requests that require DPI.

Feature	Impacts	Demand
`FirewallOnly`	`FirewallOnly`	D^s
`SampledDPI`	`Splitter`	D^s
	`DPI`	$\alpha \times D^s$
	`SplitFirewall`	$(1-\alpha) \times D^{net}$
`FullDPI`	`DPI`	D^s
`NoFilter`	*source* → *sink*	D^{net}
`FirewallOnly`	*source* → `FirewallOnly`	D^{net}
	`FirewallOnly` → *sink*	D^{net}
`SampledDPI`	*source* → `Splitter`	D^{net}
	`Splitter` → `DPI`	$\alpha \times D^{net}$
	`Splitter` → `SplitFirewall`	$(1-\alpha) \times D^{net}$
	`DPI` → *sink*	$\alpha \times D^{net}$
	`SplitFirewall` → *sink*	$(1-\alpha) \times D^{net}$
`FullDPI`	*source* → `DPI`	D^{net}
	`DPI` → *sink*	D^{net}

TABLE VII: Evaluation CFCs.

Type	Selected	Excluded
Firewall	`FirewallOnly`	-
StrictFirewall	`FirewallOnly`	`NoFilter`
SampledDPI	`DPI`	`NoFilter`
FullDPI	`FullDPI`	`NoFilter`
StrictFullDPI	`FullDPI`, `DPI`	`NoFilter`, `SampledDPI`, `FirewallOnly`

configuration is however permitted, while a cost of failing the selected feature is incurred. Excluded features may never occur in a valid configuration of the CFC. These types result in two firewalled connectivity services, one where the firewall may be disabled during short periods of time due to service overload, and one where this may not occur, and three DPI services, one using sampled, one with full DPI with fallbacks, and one without fallbacks. The source and sink nodes are chosen randomly from a set containing all of the edge nodes of the network and a VM using 4 cores and 8GB of memory. This causes the majority of services to represent an interconnection of two sites, while some will represent the connection of a remote site with a cloud-hosted VM. For CFC and feature failure, a random management cost in the set $\{8, 16, 32, 64, 128\}$ respectively $\{4, 8, 16, 32, 64\}$ is chosen, reflecting that failing to provide a degraded service, resulting in the failure of critical features, is generally worse than failing to provide part of the service chain functionality, resulting in the failure of individual features, and that some services or features may be much more important than others (e.g. a free service compared to a paid service), resulting in much higher failure costs. These failure costs are an order of magnitude larger than server use costs (shown in Table VIII) to ensure the server use cost is minimized only as a secondary objective. Network demand is chosen uniformly between 10 and 1000Mbps, while service demand is chosen uniformly between 100 and 1000 rps. These demands are multiplied with a multiplier m, making it possible to scale the application load, increasing ($m > 1$) or decreasing ($m < 1$) the demand, throughout the evaluations. For sampled DPI, 10% of the requests are handled by the DPI NF.

The evaluation network is shown in Figure 7, and represents a small service provider, containing 27 edge nodes, 9 switches, 4 core routers, a hardware firewall, a small cloud datacenter, and three smaller edge datacenters which are located closer to the edge nodes. We also consider three variants of this network: one without edge clouds, one pure NFV variant (i.e. without PNFs) and a pure NFV network without edge clouds. The network hardware specifications used during the simulation are shown in Table VIII, while the specifications of the used VNFs are shown in Table IX.

The experiments were conducted on a HPC cluster running Scientific Linux 6.1. Every compute node contains dual Intel Xeon CPU E5-2670 octo-core processors, and 64GB of physical memory. For the CFC-P and SFC-P an entire node was used[2]. The time limited algorithm versions were limited to a single CPU core and 12GB of memory. All experiments were

[2]The execution time was limited to 72 hours on a complete node if no provably optimal solution was found, which occurred for 5 entries in total.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2016.2580668, IEEE Transactions on Network and Service Management

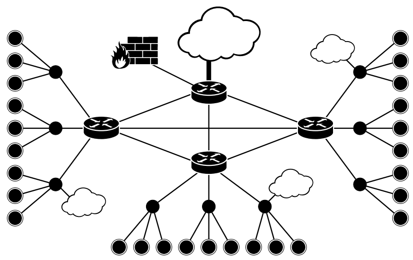


Fig. 7: The evaluation network contains a backbone network and multiple edge nodes interconnected using switches. A cloud and hardware firewall are accessible from the backbone network. Edge clouds are positioned close to the edge nodes.

TABLE VIII: Evaluation network hardware parameters.

Parameter	Value
Network link capacity	10 Gbps
Cloud uplink capacity	20 Gbps
Cloud CPU	1000 cores
Cloud memory	100000 GB
Cloud core use cost	0.1
Edge cloud CPU	100 cores
Edge cloud memory	10000 GB
Edge cloud core use cost	0.2
Hardware firewall capacity	50000 rps
Hardware router capacity	100000 rps

repeated 30 times.

VII. Evaluation Results

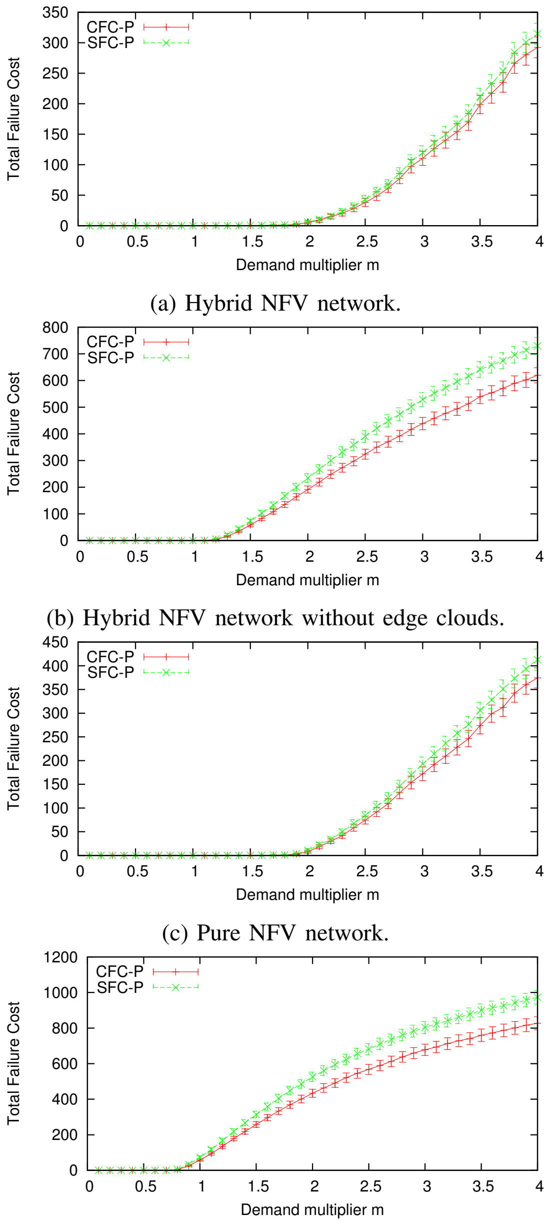
A. Total cost comparison

We compare the total cost making use of an evaluation scenario containing 5 instances of every CFC type shown in Table VII, and making use of the four network variants discussed in the previous section. Figure 8 shows how the total service failure costs compare for the various evaluation networks when the demand multiplier m is varied. In all results, CFC-P consistently results in a lower total cost than SFC-P. This is to be expected, as the CFC-P model extends SFC-P, meaning any solution for the latter is also a solution for the former. Therefore, an optimal solution for the CFC-P model will return a result with an identical or lower cost than the SFC-P solution. In all three scenarios, the total cost is dominated by the cost of failing to allocate service chains and features.

Figures 8a and 8b show two hybrid NFV networks, one where edge clouds are present, and one where they are not present. In both scenarios, network capacity becomes a bottleneck when m increases. When multiple clouds are present in a hybrid NFV network (Figure 8a) the benefits of CFC-P compared to SFC-P are limited, only resulting in marginal decreases in the cost. This is the result of the many cloud and PNF nodes spread throughout the network, resulting in many alternative ways to allocate the various service chains using SFC-P. As the cloud nodes are spread throughout the network, there will be limited network capacity, leaving little capacity for adding degraded network flows that the CFC-P could succeed in placing. Because of this, the CFC-P solution is unable to significantly reduce the total cost, as sufficient network capacity is needed to support the degraded fallback services. When no edge clouds are present however (Figure 8b), a clear difference between CFC-P and SFC-P costs can be observed. In this scenario, the SFC-P algorithm is severely restricted in how the various service chains are allocated, while CFC-P can improve the total cost by changing the feature configuration of the CFCs and partially allocating

TABLE IX: Evaluation VNF specification.

VNF	CPU	Memory	Provided resources
Routing VNF	1 core	100MB	1000 rps
Firewall VNF	1 core	100MB	1000 rps
DPI VNF	1 core	500MB	100 rps



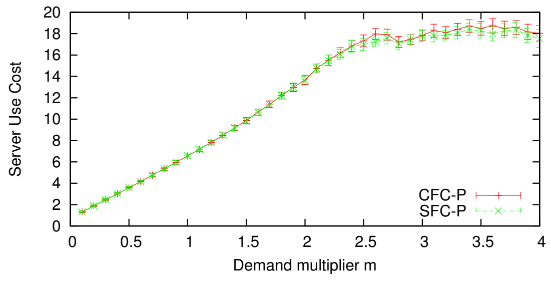
(a) Hybrid NFV network.

(b) Hybrid NFV network without edge clouds.

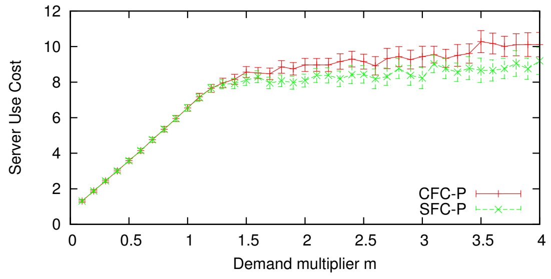
(c) Pure NFV network.

(d) Pure NFV network without edge clouds.

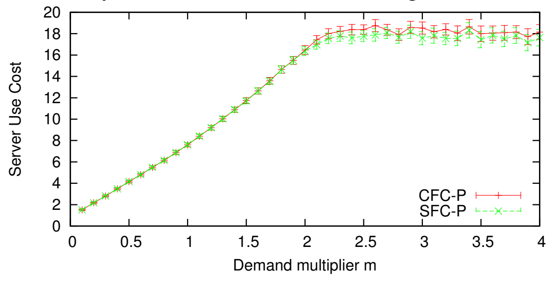
Fig. 8: Total failure cost comparison for increasing network and service demand. Values averaged over 30 iterations.



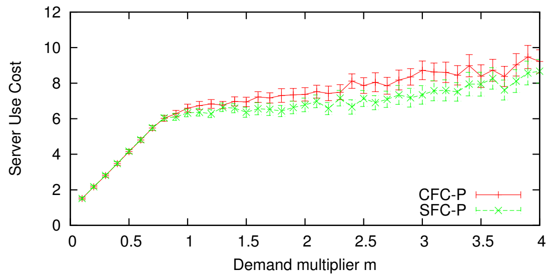
(a) Hybrid NFV network.



(b) Hybrid NFV network without edge clouds.

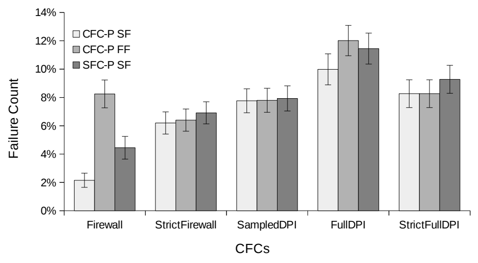


(c) Pure NFV network.

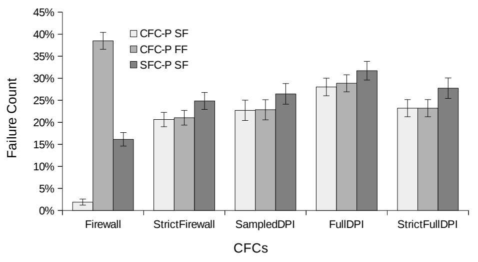


(d) Pure NFV network without edge clouds.

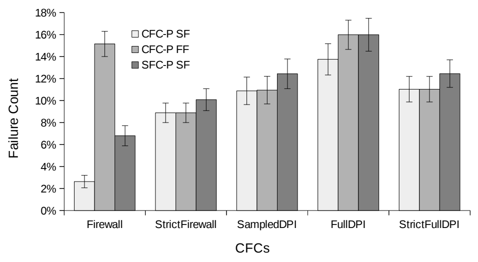
Fig. 9: Server use cost comparison in network-constrained evaluation scenarios. Values averaged over 30 iterations.



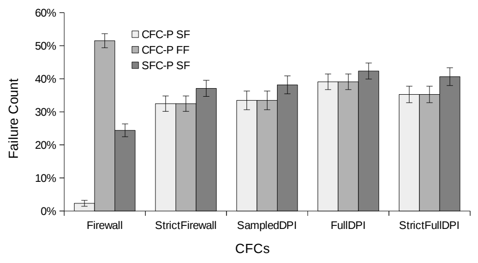
(a) Hybrid NFV network.



(b) Hybrid NFV network without edge clouds.



(c) Pure NFV network.



(d) Pure NFV network without edge clouds.

Fig. 10: A breakdown of the failures occurring during the evaluation scenario by CFC and failure type. For every CFC, the failure rate averaged over the entire evaluation scenario is shown. A distinction is made between service chain failures (SF), which lead to a complete service outage, and feature failures (FF), which result in service degradations. ($m \in [0 \ldots 4]$)

the service chain, causing it to fall back to a degraded service.

Comparing the results for a hybrid NFV network with those for a pure NFV network we find that, in pure NFV networks, the difference between CFC-P and SFC-P can be observed both when edge clouds are present, as shown in Figure 8c, and when there are no edge clouds, as shown in Figure 8d. As in the hybrid scenario, the difference between CFC-P and SFC-P is more pronounced when no edge clouds are present.

Figure 9 shows the evolution of the server utilization cost in the same scenario. We observe that CFC-P and SFC-P result in indistinguishable server use costs when no failures occur. Once the load becomes high enough to start causing failures, the server use cost of CFC-P becomes higher than that of SFC-P. This shows that CFC-P is better able to fully utilize the server capacity, resulting in the corresponding lower cost of failure discussed previously. This difference becomes more pronounced for network environments without edge clouds.

Figure 10 analyzes the service chain and feature failures, and shows the average failure rate of every CFC for the various network scenarios. For CFC-P, this failure rate is split into a service chain failure percentage and a feature failure percentage. When a service chain failure occurs, the features of the service fail as well. For the SFC-P, only the service chain failure percentage is shown, as it does not support

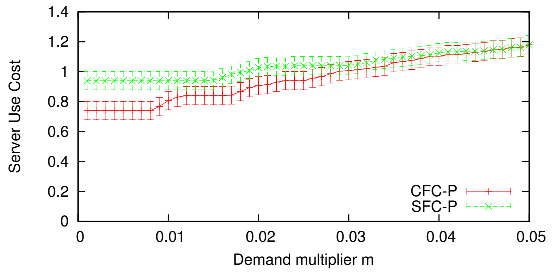


Fig. 11: Server use cost comparison of in an underloaded scenario containing only SampledDPI and StrictFullDPI CFCs in the pure NFV network.

partial service chain failures. We observe that, for the CFCs where service degradation is possible (Firewall and FullDPI), the number of feature failures exceeds the number of service chain failures, often resulting in more feature failures than there are failures of the entire service chain in SFC-P. This is particularly noticeable for the Firewall CFC, which rarely fails completely, but is frequently degraded to a lower quality version. This results from the CFC falling back to using the `NoFilter` feature, which is very easy to allocate, as it does not require an intermediary service and therefore is more flexible in how the service chain is routed. For services that can not fall back to a lower quality version, a lower failure rate is observed compared to SFC-P. This is caused by an increase in resource capacity due to the partial failures of the services that can fall back to a lower quality service.

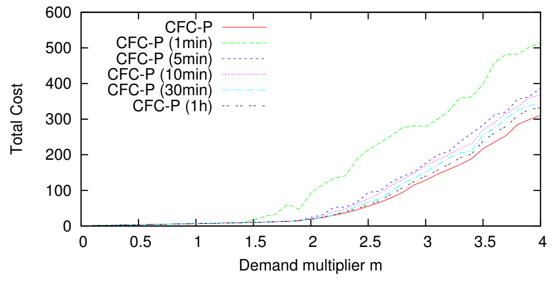
B. Server use costs

While most of the evaluation CFCs can be instantiated in different ways, most of these configurations incur a feature failure cost. Therefore, CFC-P will avoid using these CFC configurations, unless there is insufficient capacity. The SampledDPI CFC can however be implemented using either the `FullDPI` or `SampledDPI` features without incurring any cost. When an underutilized DPI NF is present within the network, workload for the SampledDPI CFC can be moved to it, reducing the load on Firewall and Routing NFs. This can in turn reduce resource consumption and the server use cost.

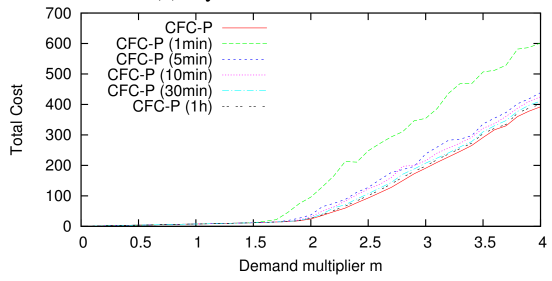
Figure 11 shows this effect for a scenario containing 10 SampledDPI CFCs and 10 StrictFullDPI CFCs in the pure NFV network. As the DPI NF within the evaluation scenario can only handle a limited number of requests per second, this behavior is only observable for low m values. When there is a low load on the system, server use cost reductions of up to 20% are observed when CFC-P is used instead of SFC-P.

C. Time-limited heuristic algorithm

We compare the performance of CFC-P when its execution is time-limited using multiple different time limits. Figure 12 shows the results of this comparison for the hybrid NFV network and the pure NFV networks. We observe that longer execution durations lead to higher quality results. Despite this, a close to optimal quality can be achieved after one minute of computation for lower demand multipliers. For higher demand,



(a) Hybrid NFV network.



(b) Pure NFV network.

Fig. 12: Total cost comparison of time-limited CFC-P in the hybrid and pure NFV networks.

longer computations may be required. For $m < 2$, the heuristic finds a solution which costs less than 5% more than the optimum. For higher multiplier values results within 20% to 30% are found in ten minutes. For the networks without edge clouds, which are not pictured, the algorithms finish noticeably quicker: after one minute a solution within $\pm 5\%$ of the optimal result is found, and after five minutes, the optimal result is found in all cases.

VIII. Conclusions

In this article, we introduced the concept of CFCs, a service chain modeling approach that extends SFCs by taking into account service chain variability. CFCs make it possible to model complex service chains in NFV networks that can be allocated in multiple alternative ways, e.g. using either physical or virtual NFs, or using different NFs with differing quality or functionality. This makes it possible to choose the best service implementation at runtime, reducing hosting costs, or to fall back to a degraded service when insufficient resources are available. We described how CFCs can be modeled and managed, and presented a formal CFC-P model that can be used to allocate CFCs on service provider networks.

We compared the CFC-P model with a SFC-P model which allocates SFCs, and therefore does not support service chain variability, using a connectivity service scenario executed in a simulated service provider network. In this scenario, CFC-P was shown to consistently result in a lower cost, indicating that taking variability into account during service chain placement can improve the quality of network services. CFC-P reduces costs in two ways: (1) by better handling service failures, making it possible to manage reduced quality versions of a service, resulting in cost reductions up to 15% depending on the scenario; and (2) by more efficiently using resources when low instance utilization occurs, showing server use cost reductions up to 20%. In future work, we will work on a prototype that will allow us to evaluate operational and systems aspects of the presented approach.

Acknowledgment

Hendrik Moens is funded by the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT). This work is partially funded by the FWO project G059615N. This work was carried out using the STEVIN Supercomputer Infrastructure at Ghent University, funded by Ghent University, the Flemish Supercomputer Center (VSC), the Hercules Foundation and the Flemish Government – department EWI.

References

[1] IETF, "Service Function Chaining (SFC) Architecture," RFC 7665, 2015. [Online]. Available: https://tools.ietf.org/html/rfc7665

[2] K. Pohl, G. Böckle, and F. Van Der Linden, *Software product line engineering: foundations, principles, and techniques*. Springer-Verlag New York, Inc., 2005.

[3] A. Basta, W. Kellerer, M. Hoffmann, H. J. Morper, and K. Hoffmann, "Applying NFV and SDN to LTE mobile core gateways, the functions placement problem," in *Proceedings of the 4th workshop on All things cellular: operations, applications, & challenges - AllThingsCellular '14*. New York, New York, USA: ACM Press, Aug. 2014, pp. 33–38.

[4] S. Mehraghdam, M. Keller, and H. Karl, "Specifying and placing chains of virtual network functions," in *3rd International Conference on Cloud Networking (CloudNet)*. IEEE, Oct. 2014, pp. 7–13.

[5] M. Bagaa, T. Taleb, and A. Ksentini, "Service-aware network function placement for efficient traffic handling in carrier cloud," in *2014 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, Apr. 2014, pp. 2402–2407.

[6] M. Bouet, J. Leguay, and V. Conan, "Cost-based placement of vDPI functions in NFV infrastructures," in *Proceedings of the 1st Conference on Network Softwarization (NetSoft)*. IEEE, Apr. 2015, pp. 1–9.

[7] S. Clayman, E. Maini, A. Galis, A. Manzalini, and M. Nicola, "The Dynamic Placement of Virtual Network Functions," in *Proceedings of the 14th Network Operations and Management Symposium (NOMS 2014)*. IEEE, may 2014, pp. 1–9.

[8] M. Yoshida, W. Shen, T. Kawabata, K. Minato, and W. Imajuku, "MORSA: A multi-objective resource scheduling algorithm for NFV infrastructure," in *The 16th Asia-Pacific Network Operations and Management Symposium*. IEEE, Sep. 2014, pp. 1–6.

[9] M. C. Luizelli, L. R. Bays, L. S. Buriol, M. P. Barcellos, and L. P. Gaspary, "Piecing together the nfv provisioning puzzle: Efficient placement and chaining of virtual network functions," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, May 2015, pp. 98–106.

[10] H. Moens and F. De Turck, "VNF-P : A Model for Efficient Placement of Virtualized Network Functions," in *Proceedings of the 10th International Conference on Network and Service Management (CNSM 2014)*. IEEE, nov 2014, pp. 418–423.

[11] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *Journal of Network and Systems Management*, vol. 23, no. 3, pp. 567–619, 2015.

[12] C. Low, "Decentralised Application Placement," *Future Generation Computer Systems*, vol. 21, no. 2, pp. 281–290, feb 2005.

[13] M. Rabbani, R. Pereira Esteves, M. Podlesny, G. Simon, L. Zambenedetti Granville, and R. Boutaba, "On tackling virtual data center embedding problem," in *Proceedings of the 13th IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*. IEEE, may 2013, pp. 177–184.

[14] F. Wuhib, R. Yanggratoke, and R. Stadler, "Allocating Compute and Network Resources Under Management Objectives in Large-Scale Clouds," *Journal of Network and Systems Management*, vol. 23, no. 1, pp. 111–136, jul 2013.

[15] M. Zhani, Q. Zhang, G. Simon, and R. Boutaba, "VDC Planner: Dynamic migration-aware Virtual Data Center embedding for clouds," in *Proceedings of the 13th IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*. IEEE, may 2013, pp. 18–25.

[16] L. Shi, B. Butler, D. Botvich, and B. Jennings, "Provisioning of requests for virtual machine sets with placement constraints in IaaS clouds," in *Proceedings of the 13th IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*. IEEE, may 2013, pp. 499–505.

[17] M. Alicherry and T.V. Lakshman, "Network Aware Resource Allocation in Distributed Clouds," in *IEEE INFOCOM*. IEEE, mar 2012, pp. 963–971.

[18] R. Esteves, L. Zambenedetti Granville, H. Bannazadeh, and R. Boutaba, "Paradigm-based adaptive provisioning in virtualized data centers," in *Proceedings of the 13th IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*. IEEE, may 2013, pp. 169–176.

[19] R. Guerzoni, R. Trivisonno, I. Vaishnavi, Z. Despotovic, A. Hecker, S. Beker, and D. Soldani, "A Novel Approach to Virtual Networks Embedding for SDN Management and Orchestration," in *Proceedings of the 14th Network Operations and Management Symposium (NOMS 2014)*. IEEE, may 2014, pp. 1–7.

[20] NFV ETSI ISG, "GS NFV-MAN 001. Network Functions Virtualisation Management and Orchestration V1.1.1," ETSI, Tech. Rep., 2014. [Online]. Available: http://www.etsi.org/index.php/technologies-clusters/technologies/nfv

[21] OASIS, "TOSCA simple profile for Network Functions Virtualization (NFV), committee specification draft 03," Tech. Rep., 2016. [Online]. Available: http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/csd03/tosca-nfv-v1.0-csd03.pdf

[22] R. Mietzner, A. Metzger, F. Leymann, and K. Pohl, "Variability modeling to support customization and deployment of multi-tenant-aware Software as a Service applications," in *Proceedings of the ICSE Workshop on Principles of Engineering Service Oriented Systems (PESOS 2009)*. IEEE, may 2009, pp. 18–25.

[23] M. Abu-Matar and H. Gomaa, "Feature Based Variability for Service Oriented Architectures," in *Proceedings of the 9th Working IEEE/IFIP Conference on Software Architecture (WICSA 2011)*. IEEE, jun 2011, pp. 302–309.

[24] ——, "Variability Modeling for Service Oriented Product Line Architectures," in *Proceedings of the 15th International Software Product Line Conference (SPLC 2011)*. ACM, aug 2011, pp. 110–119.

[25] S. T. Ruehl and U. Andelfinger, "Applying Software Product Lines to create Customizable Software-as-a-Service Applications," in *Proceedings of the 15th International Software Product Line Conference (SPLC 2011)*. ACM, aug 2011, pp. 16:1–16:4.

[26] G. H. Alférez and V. Pelechano, "Context-Aware Autonomous Web Services in Software Product Lines," in *Proceedings of the 15th International Software Product Line Conference (SPLC 2011)*. ACM, aug 2011, pp. 100–109.

[27] H. Moens, E. Truyen, S. Walraven, W. Joosen, B. Dhoedt, and F. De Turck, "Feature Placement Algorithms for High-Variability Applications in Cloud Environments," in *Proceedings of the 13th Network Operations and Management Symposium (NOMS 2012)*. IEEE, 2012, pp. 17–24.

[28] H. Moens, E. Truyen, S. Walraven, W. Joosen, B. Dhoedt, and F. De Turck, "Cost-Effective Feature Placement of Customizable Multi-Tenant Applications in the Cloud," *Journal of Network and Systems Management*, vol. 22, no. 4, pp. 517–558, oct 2014.

[29] H. Moens and F. De Turck, "Feature-Based Application Development and Management of Multi-Tenant Applications in Clouds," in *Proceedings of the 18th International Software Product Line Conference (SPLC 2014)*. ACM, sep 2014, pp. 72–81.

[30] NFV ETSI ISG, "GS NFV 002 v1.2.1. Network Functions Virtualization (NFV); Architectural Framework," Tech. Rep., 2014. [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_NFV002v010201p.pdf

[31] H. Moens, E. Truyen, S. Walraven, W. Joosen, B. Dhoedt, and F. De Turck, "Developing and Managing Customizable Software as a Service Using Feature Model Conversion," in *Proceedings of the 3rd IEEE/IFIP Workshop on Cloud Management (CloudMan 2012)*. IEEE, apr 2012, pp. 1295–1302.

[32] (2015) Scala 2.11.2. [Online]. Available: http://www.scala-lang.org/

[33] (2015) IBM ILOG CPLEX 12.6.1. [Online]. Available: http://www-01.ibm.com/software/integration/optimization/cplex-optimizer