2018 30th International Teletraffic Congress

Performance Evaluation of Service Functions Chain Placement Algorithms in Edge Cloud

Lam Dinh-Xuan[1], Michael Seufert[1], Florian Wamser[1], Phuoc Tran-Gia[1]
Constantinos Vassilakis[2], Anastasios Zafeiropoulos[2]
[1]*Institute of Computer Science, University of Würzburg, Würzburg, Germany*
[2]*Ubitech Ltd, Research Department, Athens, Greece*
[1]lam.dinh-xuan|seufert|florian.wamser|trangia@informatik.uni-wuerzburg.de
[2]constantinosvassilakis@gmail.com, azafeiropoulos@ubitech.eu

***Abstract*—The emergence of Network Function Virtualization (NFV) paradigm has become a potential solution dealing with the rapid growth of the global Internet traffic in the last decades. There, network appliances are transformed into Virtual Network Functions (VNF) running on standard server. This promises to significantly reduce overall cost and energy consumption. Additionally, hardware-based network function chain is replaced by a chain of the VNFs, called Service Function Chain (SFC). The expected benefit of SFC is the reduction in the complexity when deploying heterogeneous network services. However, the considerable drawback of SFC is the distribution of the VNFs over different hosts. An inefficient placement of VNFs can induce a high latency within the chain and wasted server resources.**

In this work, we propose four placement algorithms that aim to efficiently place the SFC in servers with regard to minimizing service response time and resource utilization. Herein, heuristic approaches are evaluated against optimal solutions for the placement problems, which are formulated by using Integer Linear Programming. We evaluate and compare these placement strategies in a simulator. Our result shows that the optimized solutions produce lowest service response time and least server utilization in all types of simulated SFCs. On the other hand, the heuristic algorithms are also able to come close to the optimum by simple placing rules.

***Index Terms*—Network Function Virtualization, Service Function Chain, Placement, Optimization, Edge Cloud**

I. Introduction

The tremendous growth of global Internet traffic has been forcing network operators to struggle with reducing capital and operational costs [1]. Additionally, they must cope with the increasing demand for flexibly provisioned services. Thereby, the network service must be provided by the network operators with a high Quality of Experience (QoE) in order to achieve high customer satisfaction and to avoid user churn [2], [3].

To handle these problems, a more innovative and agile network technology has been emerged, called Network Function Virtualization (NFV) [4]–[6]. The main idea of this paradigm is to decouple the network functions from their physical hardware. For example, the routing function of a router can be detached from its expensive dedicated hardware to become a plain software, which can run on any commodity server. Such kind of function is called Virtual Network Function (VNF). In the NFV architecture, multiple VNFs can be chained together. The resulting construct is called *Service Function Chain (SFC)* [7], [8]. In such a chain, typically, each VNF executes a certain function and all VNFs must be processed in a specific order. Thereby, the VNFs can be dynamically deployed in a server or distributed over different servers in datacenters to meet various requirements. For example, multiple VNFs can be instantiated for a particular function in order to increase resilience or for load balancing. Thus, the advantage of SFCs is promising to reduce the complexity of deploying heterogeneous network services.

However, deploying such a SFC in an NFV system has several challenges. On the one hand, VNFs distributing in multiple servers will increase the length of the chain, if the servers are placed in different datacenters. This will considerably rise the latency within the SFC itself and can reduce the QoE. On the other hand, since the VNFs can be placed in different servers, their resource utilization must be taken into account. In fact, complex optimization problems can be formulated for a given system, but their solution can be time consuming. Thus, heuristics are needed to quickly obtain solutions, which achieve a close-to-optimal performance.

To address these problems, we propose and evaluate two heuristics for distributing VNFs of service chains in datacenters of an edge cloud, named *centralization* and *orchestration* algorithms. These placement algorithms aim to minimize total latency or server utilization. They are evaluated against optimal solutions for the placement problems, which are formulated and solved by using Integer Linear Programming (ILP). For the performance evaluation, we extend the event-based EdgeNetworkCloudSim simulator [9] with the inclusion of the CPLEX Optimizer toolbox[1]. This toolbox uses Optimization Programming Language (OPL) to express the ILP mathematical model. Then, the OPL model is solved by using the CPLEX Optimizer.

In fact, SFC placement problem is widely studied on different directions. The objective of existing works has focused on cost reduction [10], [11], virtualization and trade-off between different objectives [12], [13] or optimizing energy consumption [14], [15] among others. Our study is different from these research works, where we consider the SFC placement problem in the context of edge cloud computing. Wherein, the

[1]https://www.ibm.com/analytics/data-science/prescriptive-analytics/cplex-optimizer

978-0-9883045-5-0/18/$31.00 ©2018 ITC
DOI 10.1109/ITC30.2018.00041

user is near to datacenter and his device is included as the end point of the whole chain.

In this study, we use EdgeNetworkCloudSim to simulate a fixed network topology in an edge cloud. Users randomly request service chains consisting of three VNFs that can be placed on different servers. The performance of all placement algorithms is evaluated with respect to QoE in terms of service response time, and with respect to resource consumption in terms of number of utilized servers. Our simulation result shows that the optimized solutions obtained by using ILP model achieve lowest service response time and least service utilization rate among the others. However, the heuristic algorithms are able to come close to the optimum by simple placing rules. Our insights may help network operators and the research community to quickly compute good SFC placements for NFV infrastructures in an edge cloud context.

The remainder of the paper is structured as following. After the introduction, background and related work are presented in Section II. Thereafter, Section III describes the extension of EdgeNetworkCloudSim, edge cloud topology, and performance metrics. The description of four placement algorithms is presented in Section IV. The outcome of our study is detailed in Section V. Finally, Section VI concludes this work.

II. Background and Related Work

In this section, we first introduce edge cloud and the definition of service function chain in Section II-A. Thereafter, we give an overview of related works in Section II-B.

A. Background

Edge computing is a method of enhancing cloud computing systems by off-loading applications, services, and hardware resources to the edge of the network [16]. Therefore, the edge cloud introduces a new intermediate layer at the edge of the network, which is physically placed in between cloud datacenter and user. This removes a major bottleneck and reduces high latencies in services due to the long distance to the user. Thus, in edge computing, time is the key parameter. In contrast to a conventional cloud datacenter, the latency here between the user and the SFC is relatively small, since the user device is located next to the datacenter. Consequently, the latency between the VNFs in the chain is important and has a high impact on the overall latency. A good SFC placement strategy produces a low internal delay in the SFC, which contributes much to the overall service response time. Thereby, study on SFC placement algorithms in the edge cloud is essential. In contrast, in cloud computing, the VNFs of a SFC are placed at datacenters. With a high latency between the user and the datacenter, the influence of an SFC optimization algorithm on the overall service response time is negligible with a fixed user compared to an edge cloud scenario.

In this paper, we assume that a user device is directly connected to an edge cloud with four datacenters. The user requests a service in the edge cloud, and receives response from servers located in these datacenters. We simulate three types of personal services, *Video Streaming (STR)*, *Web*, and *Database (DB)*. These services are characterized by their own virtual machine resource demands, and are requested and used by only one single user. Note that, these services do not resemble real cloud applications, but they were mainly specified in order to have different service chain characteristics. Each service is requested and processed as a chain of VNFs, thus, in the remainder of this paper, the terms SFC and service will be used interchangeably. In the simulation, we define a SFC consisting of three VNFs that must be executed in order to provide full service functionality. This definition is also consistent with the SFC described in [7]. Figure 1 shows an overview of communication between a user and a SFC.

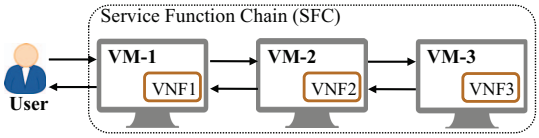

Fig. 1. Overview of Service Function Chain Communication

In Fig. 1, each VNF is installed in a Virtual Machine (VM). Herein, the type of a VM also represents the VNF requirements. The VM types are predefined based on instance types introduced by Amazon Elastic Compute Cloud[2]. Table I shows the definition of the VM types in EdgeNetworkCloudSim.

TABLE I
VM Type Definition

VM Type	CPU	RAM
T2Nano	1	1024 MB
T2Small	2	2048 MB
T2Large	4	4096 MB

In Fig. 1, the arrows represent the direction of data flow and the sequence of processing VNF tasks. Specifically, when the user requests a service, the request is processed at VNF1, followed by VNF2, and VNF3. Then, the response message will also be sent back in sequence from VNF3 to the user. Therefore, the placement of VMs primarily influences the latency between the VNFs in the chain, and thus, also the service response time. This makes it crucial to design placement strategies, which quickly find a good possible placement of SFCs in terms of QoE and resource utilization.

B. Related Work

In [17], Calheiros et al. present their original work on an event-based simulator for cloud computing infrastructure, called CloudSim. An extension of CloudSim with a GUI is CloudAnalyst [18]. In [19], Garg et al. introduce another extension of CloudSim that enables network communication, called NetworkCloudSim. In [9], Seufert et al. introduce EdgeNetworkCloudSim that is an extension of NetworkCloudSim. There, EdgeNetworkCloudSim moves from batch-like processing of computation jobs to more persistent and personalized cloud services that are implemented in an edge cloud. In the simulation, the authors allow to define several

[2]https://aws.amazon.com/ec2/instance-types/

characteristics of cloud services, which can be processed in a chain of VMs. The framework is able to simulate user requesting a service and can compute response times and resource utilization among others.

The architecture for the specification of SFCs is officially described in [7] by IETF. In fact, the concept of SFC receives a considerable interest from research community. In [8], John et al. introduce several research topics on network service chaining. In [10], Savi et al. study the impact of processing capability on the placement of service chains. They model a set of NFV nodes hosting service chains using an ILP model. The main objective of the model is to minimize the number of active NFV nodes with regard to the cost of processing task. The authors conclude that with the increasing number of service chains, the context switching costs strongly influence the implementation cost of NFV.

In [12], Luizelli et at. consider the placement of SFC where the placement problem is formulated by using an ILP model. The main objective of the model is to minimize the number of VNF instances mapped to infrastructure. The authors conclude that the ILP model leads to a reduction of up to $25\,\%$ in the end-to-end delay compared to a baseline. However, their baseline solution is another ILP model, where the objective function is changed to minimize the chain length.

The authors in [13] formalize the chaining of VNFs using a context-free language. Then, they formulate the placement problem of chained VNFs as a Mixed Integer Quadratically Constrained Program with regard to three different objectives. Their results show a trade-off between optimizing the remaining data rate, latency, and number of used nodes. In [14], the authors propose the Merge-RD algorithm to place service functions with respect to minimizing energy consumption. They use GreenCloud to simulate a datacenter topology and to evaluate the performance of the proposed algorithm. However, with $O(n^4)$, the complexity of the algorithm is quite high and time consuming. Other approaches in energy-efficient and bandwidth-efficient SFC placement are presented in [15] and [20], respectively. In [11], Bari et al. propose a heuristic algorithm and an ILP model for optimizing SFC placement. The algorithms aim to reduce capital and operational cost of VNF deployment in the operator network. They conclude that the heuristic algorithm outperforms the optimal solution in the aspect of execution time.

The method of using an ILP model to formulate the placement problem of VNFs is widely studied. In [21], Bouet et al. propose a study that aims to minimize the overall cost of vDPI deployment in an NFV infrastructure. The authors conclude that the network structure and costs strongly influence the execution time of the vDPI function. However, the authors do not consider the chaining form of vDPI in their ILP model. Similar studies on solving the placement problem of VNFs formulated by using ILP are presented in [13], [22], [23].

The existing studies use similar methods as our study to some extents in the aspect of optimizing SFC placement. Most of the works are based on mixed integer programming with different objective functions and constraints. Although, their approaches can deliver optimal solutions for individual problems, the performance of their proposals still need to be assessed in a real scenario. Moreover, their result is mostly obtained by solving the ILP model using CPLEX.

Our study is different from the mentioned research works since we compare four different approaches for SFC placement. This helps the network operator to have a comparative view of the advantages and drawbacks of each solution. Moreover, we consider an edge cloud context where the user is close to the datacenter. In our heuristic algorithms and ILP model, the user device is also included as the end point of the whole chain. Furthermore, to the best of our knowledge, this study is the first that uses EdgeNetworkCloudSim to simulate and evaluate the influence of different SFC placement strategies on service response time or server utilization. Wherein, user requests and SFC placement are done consecutively on the same platform that increases the practical of our approach.

III. SFC Simulation in EdgeNetworkCloudSim

In this section, we first present the EdgeNetworkCloudSim extension. Subsequently, we introduce the edge cloud topology, simulation configurations, and service chain characteristics used for the simulation. Finally, we present several metrics to evaluate the performance of the algorithms.

A. EdgeNetworkCloudSim Extension

Placement algorithms for virtual machines can be easily implemented in EdgeNetworkCloudSim. However, the framework did not provide means to compute an optimal placement with the help of ILP. In this work, EdgeNetworkCloudSim is extended to be able to implement OPL models and solve them with a CPLEX Optimizer. The following new classes are added to the simulator to achieve the mentioned goal.

First, the **AvailableResource** class is implemented whenever a new service is requested. It monitors resource utilization (e.g., CPU, RAM) of all servers in datacenters. Additionally, the resource demand of the VMs in SFCs is also provided. This information is used for the **OPLData** class.

The **OPLModel** class contains the pre-defined OPL model with two objectives, which are minimizing service time and resource utilization. The mathematical expression of the model can be found in Section IV. The **OPLData** class contains the data of the OPL model. The data consists of static and dynamic information. The static information like topology and link resources between nodes is initially provided. The dynamic information consists of available resources of the servers (e.g., CPU, RAM), resource demand of the VMs, and location of the user. Note that the user in EdgeNetworkCloudSim is simulated as another VM, called *UserVM*. This *UserVM* is located in a server of a dedicated datacenter, called *UserDC*, which can not be used to place virtual machines of SFCs. The dynamic information is gathered by the **AvailableResource** class and regularly updated whenever a new SFC is requested.

Lastly, the **PlacementSolver** class is triggered when a new service has been requested and the *UserVM* has been already specified by the simulator. This class exploits the CPLEX

Optimizer toolbox to solve the OPL model and returns one optimal solution each time. The solution is the specific location of each VM of the service chain. EdgeNetworkCloudSim can now make use of the solution and place these VMs accordingly to start the service. If no solution can be found due to insufficient available resources in the system, EdgeNetworkCloudSim will discard the incoming SFC request. Since we do not modify the placement during service run time, a new optimal placement will be computed when a SFC has been terminated and allocated resources are released.

B. Edge Cloud Topology

Figure 2 shows the topology of the simulation. This topology is designed based on a real testbed of the EU H2020 INPUT project [24]. It consists of four datacenters (*DC*) and four user datacenters (*UserDC*). One of the *DCs* has two servers, and the others have only one. These servers have different resource capacity. In the simulation, the VMs of SFCs are distributed over these five servers of the edge cloud depending on their available resources. The user device is simulated as a *UserVM* that is installed within a server in the *UserDC*. Each *UserDC* is connected to a *DC*, thereby, representing all users, which are close to that edge datacenter. Each *UserDC* is considered to have unlimited resources to host *UserVMs*. The interconnection of *DCs* and *UserDCs* in edge cloud is operated by different link capacities via two types of switch. The server in each datacenter is directly connected to its EdgeSwitch by a link with $5\,\mathrm{ms}$ delay. However, in *DC-1*, two servers are set to be interconnected with zero delay. The EdgeSwitch of a *DC* is connected to its AggregateSwitch through a $5\,\mathrm{ms}$ delay connection. While, in case of the *UserDC*, this connection has $10\,\mathrm{ms}$ delay. All AggregateSwitches are interconnected via a link with $50\,\mathrm{ms}$ delay. In the simulation, this topology is mapped with a BRITE file [25] for modeling link bandwidth and associated latency. Since we only consider link delay in the topology of the simulation, we configure the link bandwidth of the topology with a large number to ensure there is not any additional latency in the network due to bandwidth bottlenecks.

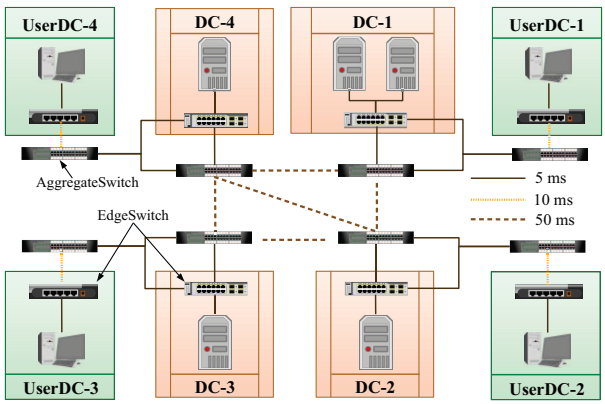

Fig. 2. Overview of Network Topology in Simulation

C. Service Chain Characteristic

As briefly described in Section II-A, we simulate three types of personalized service. Each service chain has a single *UserVM* with a fixed location in a *UserDC*, which sends requests only to its particular service chain. Each service chain requires three VMs with various types. Table II summarizes the required VMs type of each service.

TABLE II
SUMMARY OF SFCS WITH MULTIPLE VM TYPES

Service	VM-1	VM-2	VM-3
Video Streaming	T2Small	T2Nano	T2Large
Web	T2Large	T2Small	T2Nano
Database	T2Nano	T2Nano	T2Small

The table shows that the Streaming and Web services require a similar total size of VMs. However, their characteristics are different. The Streaming service is simulated to deliver video with different lengths. Specifically, the video length is distributed exponentially around a mean of $30\,\mathrm{s}$. Each delivered video chunk has $2\,\mathrm{s}$ in length and $1100\,\mathrm{KB}$ in size. Consequently, when a Streaming service is requested, the number of responses is corresponding to the number of video chunks. On the other hand, Web and Database services have only one response per request, but their response data size is distributed around a mean of $2000\,\mathrm{KB}$ and $50\,\mathrm{KB}$, respectively. Besides, to simulate user-like behavior in the simulation, we use an exponential distribution for service requests (i.e., the time at which the service is instantiated) with the mean value of $5\,\mathrm{min}$.The service life times and user request inter-arrival times also use exponential distribution and are different among simulated SFCs. Specifically, the mean service life times of the *Web*, *Database*, and *Streaming* services are $15\,\mathrm{min}$, $80\,\mathrm{min}$, and $20\,\mathrm{min}$, respectively. Whereas, the corresponding mean request inter-arrival times are $1\,\mathrm{min}$, $10\,\mathrm{min}$, and $5\,\mathrm{min}$, respectively.

D. Performance Metrics

In this subsection, we define several metrics that are used to evaluate the performance of SFC placement algorithms.

1) Service Response Time and Hop Count: Service response time is the amount of time between the user request a SFC and the reception of its response. In the simulation, the service response time consists of the sum of all link delays and the processing times at all VNFs. Herein, the processing time at each VNF is set by default of $50\,\mathrm{ms}$ in the simulator for all services. Therefore, the total link delay is the main factor that impacts the service response time of a SFC. Additionally, the total *hop count* from the user to the SFC, which is the number of intermediate nodes including switches between the UserVM and the datacenter can be analyzed. It shows insights into the dispersion of the SFC over different servers in the topology.

2) Resource Utilization: This is an important metric, since reducing power consumption saves energy. In that way, additionally greenhouse gas emission is reduced, decreasing carbon footprint. To this end, the idea is to place all VMs on the smallest set of servers capable to deal with all running tasks.

Empty servers can then be shut down to save energy. Based on this, we consider the *number of servers*, which are utilized to provide a given number of services.

IV. SFC Placement Algorithms

In this section, we present four SFC placement strategies, which are *Centralization (CEN)*, *Orchestration (ORC)*, *Service Time Optimization (STO)* and *Resource Optimization (RO)*. On the one hand, CEN and ORC are heuristic approaches. These algorithm are designed and included within EdgeNetworkCloudSim to search quickly for a proper placement of SFCs. On the other hand, STO and RO are optimal solutions, where the placement problems of SFC are formulated using Integer Linear Programming (ILP). In the simulation, the *UserVM* is randomly chosen with equal probability among four *UserDC* locations and is applied for all algorithms.

A. Centralization Algorithm

The centralization (CEN) placement algorithm tries to place all VMs of a SFC as close as possible to the user, meaning to have the lowest delay between the VMs and the user. This algorithm is useful in classical cloud computing where independent VMs of a user should be placed close to him. However, it is unclear how the CEN performs in case of service chains with communicating VMs. Algorithm 1 shows the simplified pseudo-code of the centralization approach.

Algorithm 1 Centralization

```
1: Input: List of VMs, requested service and UserVm
2: Initialization;
3: for each vm in chain do
4:     DC = findClosestDcToUser(uservm);
5:     if DC == -1 then
6:         abandonService();
7:     end if
8:     createVmInDc(vm, DC);
9: end for
```

Data provided for the algorithm are an ordered list of VMs, *UserVM* location, and type of service. When receiving requests to create VMs in a data center (*DC*), the simulator starts to find a closest *DC* to the user for each VM. If there is still an available *DC*, then the VM is placed in sequence and the service can be processed. All the VMs can be placed in one *DC* or distributed over different data centers. If at least one VM could not be placed, there are not enough resources (i.e., CPU or RAM) in the system to deploy the entire SFC. In this case, the requested SFC will be discarded or blocked.

In the CEN approach, all VMs of the service chain will be placed around and close to the *UserVM*, but the algorithm does not try to reduce the length of the chain. A large VM might be placed far away from the *UserVM* due to the lack of resources at the closer *DC*, but a smaller VM later in the chain may fit. As a consequence, the length of the chain is increased and data oscillate between data centers. This might significantly influence the efficiency of the algorithm with respect to service response time.

B. Orchestration Algorithm

The orchestration (ORC) algorithm differs from the CEN, where only the first VM in the chain is attempted to be placed as close as possible to the user. For the subsequent VMs, the aim is to place them close to the previous VM. Based on this, ORC tries to shorten the length of the chain that reduces the latency. Algorithm 2 shows the simplified pseudo-code of the orchestration algorithm.

Algorithm 2 Orchestration

```
1: Input: List of VMs, requested service and UserVm
2: Initialization;
3: DC = findClosestDcToUser(uservm);
4: if DC == -1 then
5:     abandonService();
6: else
7:     createVmInDc(firstVm, DC);
8:     for next vm in chain do
9:         DC = findClosestDcToPreviousVm();
10:        if DC == -1 then
11:            abandonService();
12:        end if
13:        createVmInDc(vm, DC);
14:    end for
15: end if
```

Similar to CEN, this algorithm is provided with the ordered list of VMs, *UserVM* location, and type of service. At first, it tries to find a closest *DC* to the *UserVM* for the first VM in chain. Afterwards, the algorithm tries to place the next VM in chain as close as possible to the previous one in sequence. Additionally, the next VM is only sent if the previous one has been successfully placed. Thus, ideally all VMs are placed in the same *DC* if it has enough resources. If no *DC* is found in any case, the requested service is abandoned.

This algorithm overcomes the limitation of CEN, as it avoids placing later VMs close to the user rather than close to the previous VMs if possible. This helps to reduce the delay within the SFC. Nevertheless, prioritizing the closest *dc* for the first VM is not always the best option. Especially, when this closest *dc* has only resources for some VMs and the other VMs must be distributed to another farther *dc*. This leads to an increased hop count within the chain and increased latency. In this case, placing all VMs in the farther *dc* might be better, since the delay within the chain would be zero. However, a heuristic for this idea is not evaluated in this work. In the next subsection, we present an optimized approach where the placement problem of SFC is formulated by using ILP model.

C. Service Response Time and Resource Optimization

To formulate the problem and align with the definition of the infrastructure and SFC described before, we consider that each server is a member of a datacenter. We assume that all servers within a datacenter are fully connected with links of practically infinite bandwidth and zero delay. The first node of a network inside a datacenter is considered to be the network gateway. A *UserVM* is declared as statically allocated in a

server within a dedicated *UserDC*. The *UserVM* only acts as the source of requests to a particular service chain. Table III shows the notations that are used for the formulation of the optimization problem.

TABLE III
SUMMARY OF PARAMETERS USED IN THE ILP MODEL

Parameters	Description
T	Set of application components
C	Set of channels between application components, $C \subseteq T \times T$
H	Set of hosts
L	Set of links between hosts, $L \subseteq H \times H$
S	Set of user application components statically allocated at hosts, $S \subset T$
H^S	Set of hosts where static user application components are placed, $H^S \subset H$, $f : S \to H^S \mid \forall h' \in H^S, \exists t' \in S : h' = f(t')$ (f is subjective)
R	Set of unique resources offered by hosts
R'	Set of unique resources offered by links
M	Set of monitored metrics at hosts
M'	Set of monitored metrics at links
a_t^r	Amount of resource r demand by application component t
i_h^r	Capacity of resource r at host h
β_h^r	Amount of resource r available at host h
m_h^k	Measured value of metric k at host h
c_{sd}^r	Amount of resource r demand required by channel (s, d)
b_{uv}^r	Amount of resource r available at link (u, v)
μ_{uv}^k	Measured value of metric k at link (u, v)

Based on the notations and the considerations mentioned before, the optimization problem is formulated as follows.

Given:

$$R = \{CPU, Memory\}$$
$$R' = \{Bandwidth\}$$
$$M = \emptyset$$

Minimize:

- ***Objective 1:*** Minimize service response time (STO)

$$Objective_1 = \sum_{(s,d)\in C} \pi_{uv,sd}\mu_{uv}^{Delay}, \quad \forall (u, v) \in L. \quad (1)$$

- ***Objective 2:*** Minimize resource utilization (RO)

$$Objective_2 = \sum_{h\in H} (min\{\sum_{t\in T} \sigma_{ht}, 1\} \frac{100\beta_h^{CPU}}{i_h^{CPU}}). \quad (2)$$

Objective 2 aims to minimize the product of the number of servers used in a placement and the percentage of available CPU. This optimization procedure will attempt to collocate VMs in a server but will have the preference to an already utilized server. Thus, the servers that have zero utilization will not be activated until the others have been fully utilized. By doing this, unused servers can be put in the idle state to save energy. However, at the initial placement all servers will have the same probability to be selected. The objective function does not favor any of them (e.g., a server with high capacity).

Subject to:

$$\pi_{uv,sd} \in \{0, 1\}, (s, d) \in C, (u, v) \in L. \quad (3)$$

In Equation (3), $\pi_{uv,sd}$ is a decision variable and equals to 1 if task channel (s, d) is routed from link (u, v), 0 otherwise.

$$\sigma_{ht} \in \{0, 1\}, \quad h \in H, t \in T. \quad (4)$$

In constraint (4), σ_{ht} is a decision variable and equals to 1 if task t is assigned to host h, 0 otherwise.

$$\sum_{h\in H} \sigma_{ht} = 1, \quad \forall t \in T, \quad (5)$$

$$\sigma_{ht} = 1, \quad h \in H^S, t \in S, \quad (6)$$

$$\sum_{t\in T\setminus S} \sigma_{ht} = 0, \quad \forall h \in H^S. \quad (7)$$

Constraint (5) ensures that a task (or application component) is assigned only to one host. The static placement of the user task is defined in Eq. (6), it is given as an input to the problem and not decided. Whereas, constraint (7) specifies that user applications are only placed in hosts assigned for them.

$$\sum_{t\in T} \sigma_{ht}\alpha_t^r \leq \beta_h^r, \quad \forall r \in R, \forall h \in H. \quad (8)$$

Equation (8) stipulates that the considered host h must have enough resources to allocate the application component t.

$$\sum_{(u,h)\in L} \pi_{uh,sd} + \sigma_{hs} = \sum_{(h,v)\in L} \pi_{hv,sd} + \sigma_{hd}. \quad (9)$$

Constraint (9) captures and expresses in one equation,

- the unsplittable flow constraint: A channel uses a single outgoing link from source and a single incoming link at destination and does not split,

$$\sum_{(u,h)\in L} \pi_{uh,sd} = 1 \quad if\sigma_{us} = 1,$$
$$\sum_{(h,v)\in L} \pi_{hv,sd} = 1 \quad if\sigma_{vd} = 1,$$

- the collocation of tasks: A communication path is not required in the case that both s and d are assigned to the same host (and no capacity checking),

$$\sigma_{hs} = \sigma_{hd},$$
$$\pi_{uu,sd} = 0,$$

- the flow conservation constraint: No traffic is stored in a node unless this node is the source or the destination or collocated source and destination,

$$\sum_{(u,h)\in L} \pi_{uh,sd} = \sum_{(h,v)\in L} \pi_{hv,sd},$$
$$\forall h \in H : \sigma_{hs} = 0, \sigma_{hd} = 0.$$

$$\sum_{(u,h)\in L} \sigma_{hs}\pi_{uh,sd} = 0. \quad (10)$$

Constraint (10) makes sure that there is no loop in the path before reaching destination.

$$\pi_{uv,sd} = \pi_{vu,ds}, \quad (s, d), (d, s) \in C, (u, v), (v, u) \in L. \quad (11)$$

As determined in Eq. (11), a bidirectional communication between two tasks is routed through the same bidirectional overlay path. In addition to this, upstream and downstream of a flow is not routed separately.

$$\sum_{(s,d)\in C} \pi_{uv,sd}c_{sd} \leq b_{uv}, \quad \forall (u, v) \in L. \quad (12)$$

Constraint (12) guarantees that the link (u, v) must have enough resource required by channel (s, d).

V. Result

In this section, we present the simulation results that compare the performance of four SFC placement algorithms. We simulated three types of personalized services with individual algorithms in the extended EdgeNetworkCloudSim. The details of simulation implementation in EdgeNetworkCloudSim can be found in the original work [9]. Based on the topology and metrics presented above, we implement 20 replications for each algorithm to increase the statistical significance. In the next subsections, we evaluate the performance of the algorithms according to two criteria, *service response time* and *resource utilization*. The algorithms under evaluation are *Centralization (CEN)*, *Orchestration (ORC)*, *Service Response Time Optimization (STO)*, and *Resource Optimization (RO)*. Wherein, *STO* and *RO* are $Objective_1$ and $Objective_2$ formulated in Eq. (1) and Eq. (2), respectively.

A. SFC Placement Algorithms vs. Service Response Time

Figure 3 shows the average service response time of the three SFCs with different placement algorithms. The x-axis shows the services, the y-axis shows the service response time in milliseconds. The bar group with different colors of each service shows the mean service response time with a $95\,\%$ confidence interval of different placement algorithms.

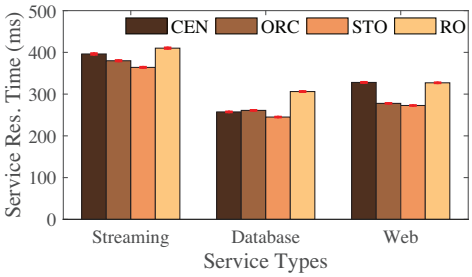

Fig. 3. Average Service Response Time of Different SFCs

Figure 3 shows that the Streaming service has a higher service response time than the Database and the Web service. It is due to the fact that the Streaming service responds multiple video chunks per request, depending on the video length as described in Section III-C. In fact, the average total response time of the Streaming service is about $6000\,\mathrm{ms}$. However, for the sake of comparability with other services, we only show in Fig. 3 the average service response time of one video chunk. The overhead of sending multiple video chunks increases the average response time of one chunk as shown in the figure. Conversely, the Database service has the lowest average response time of about $250\,\mathrm{ms}$. Since it requires a lower total size of all VMs, the placement algorithms have a higher chance to place the VMs in a desired server which decreases the overall service response time.

Regarding the service response time produced by different placement algorithms, STO gains the lowest service response time of all services, followed by ORC, CEN, and RO. For instance, when STO is used as placement algorithm for the Web service, it takes $272.80\,\mathrm{ms}$ for a user to request the service until receiving its response. Whereas, by using ORC, CEN, or RO algorithms, it takes $277.80\,\mathrm{ms}$, $328.00\,\mathrm{ms}$, and $327.10\,\mathrm{ms}$, respectively. This does not come as a surprise, as STO was designed to compute a placement, minimizing the service response time by using the ILP model and considering the whole system state. Note that, since our topology has only four data centers, the processing time of CPLEX Optimizer is negligible. However, with a larger network topology, the solution space created by the CPLEX Optimizer is also large. As a consequence, the solving time for an optimal placement can negatively impact the overall service response time, since the optimal solution is a *NP-hard* problem.

Thus, heuristic approaches have to be investigated, such as ORC which reaches the second lowest service response time for Streaming and Web services. In this algorithm, all VMs in the chain are placed to have shortest distance between them. The first VM in the chain is placed as close as possible to the user. As a result, ORC constantly tries to minimize the length of the chain. In fact, ORC has to scan all servers with multiple loops to find the best placement for VMs. The chosen servers must have enough resources for all VMs as well. This operation is executed each time for a new coming service. Thus, with an increasing number of requests from the user, it also influences the overall service response time.

In fact, the ideal placement for the lowest service response time is the situation, where all VMs in the chain are placed in one server. In this case, the delay between the three VMs in the chain is zero, which substantially decreases the overall service response time. Figure 4 shows the probability of the occurrence of this situation. The x-axis indicates the three service types, the y-axis shows the probability of placing the whole chain in one server. The bars with different colors represent the mean probability for the different algorithms with a $95\,\%$ confidence interval.

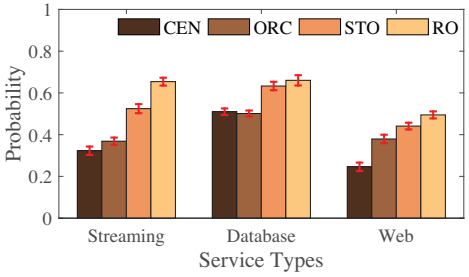

Fig. 4. Probability of Placing the Complete Chain in one Server

It can be seen that RO exhibits a higher results compared to the other algorithms, since it is specially designed for resource optimization. This behavior is described in more detail in the next subsection. Regarding the other algorithms, Figure 4 shows that STO has a considerable higher value than ORC and CEN algorithms. In the Streaming service, there is $52.44\,\%$ chance that STO places the whole chain of the service in one server, while this number in case of ORC and CEN is $36.86\,\%$

and 32.29 %, respectively. This tendency is also encountered in the Database and the Web services. This is reasonable, since STO calculates the SFC placement based on minimized total delay. Thereby, it is one option to place all VMs in one server. For instance, the selected server for the whole chain may not be the closest one to the user. Since the delay between the VMs in the chain is zero, the total delay is still smaller than the case where the VMs are distributed over different servers. This is the major difference between the optimized solution and the heuristics. Indeed, ORC and CEN choose the placement of a SFC based on iteratively scanning every server in data centers. They try to select the server as close as possible to the user. As a consequence, the closest servers are rapidly running out of resources. Afterwards, the VMs of the next SFC must be distributed over different servers. This is the reason why their probability of all VMs in one server is lower.

Figure 5 shows the mean hop count calculated from different algorithms. The x-axis shows the algorithms and the y-axis displays the mean hop count with a 95 % confidence interval.

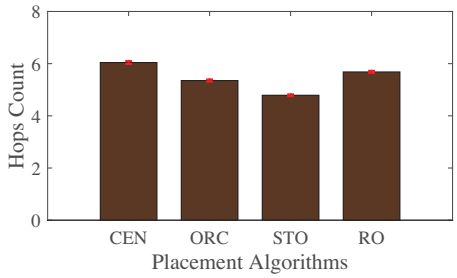

Fig. 5. Average Hop Count of Different Placement Algorithms

The figure indicates that, CEN has the highest average hop count compared to other algorithms. The average number of intermediate nodes between the user and SFC calculated by CEN is 6.04, while in case of STO and ORC is 4.7 and 5.3, respectively. In contrast to ORC, the CEN algorithm always selects the closest server to the user to place VMs, without consideration of the length of the chain. Consequently, although the chosen servers are near to the user, the data flow might be transferred in a long chain that significantly increases the service response time.

To conclude this subsection, we have shown that the STO placement strategy carried out by using ILP model has the highest performance, since it achieves the lowest service response time of all types. However, the performance of STO might be influenced by the processing time of CPLEX Optimizer in a larger topology. In this situation, the solving time for the optimization problem would be high and might considerably increase the overall service response time. The heuristic algorithm ORC shows an acceptable performance, since it attempts to minimize the length of the chain. Although RO has low performance in service response time, it is particularly designed for resource optimization as presented in the next subsection.

B. SFC Placement Algorithms vs. Server Utilization

As presented in Section IV-C, the second objective of our ILP model is to specify the placement of a SFC, where resource utilization is minimized (i.e., RO algorithm). Figure 4 in the previous subsection indicates that RO has a noticeable higher probability that it places all VMs of a SFC in one server compared to the others. To this end, the objective function tries to minimize the number of servers used for SFCs regarding their available resources. This means, RO will place as much SFCs as possible in one server and has the preference to an already utilized server as well. Thereby, the unused servers can be put in the idle mode or shut down to save energy.

To evaluate the performance of this placement strategy, we calculate the number of utilized servers along with the number of concurrently instantiated SFCs. Herein, a server is considered as utilized when at least one CPU is allocated to a VM of a SFC. Figure 6 shows the correlation between the number of utilized servers and the number of concurrent services. The x-axis shows the number of concurrent services, the y-axis indicates the average number of corresponding used servers, meaning the server utilization (ServUtil) rate. The different colored lines display the mean ServUtil rate produced by different placement algorithms. The error bars on each line indicate the 95 % confidence interval of the mean values.

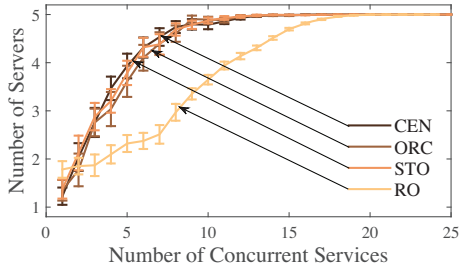

Fig. 6. Number of Utilized Servers vs. Number of Concurrent Services

Figure 6 shows that STO, ORC, and CEN placement algorithms produce similar ServUtil rate, when all servers are handled by more than ten concurrently instantiated SFCs. This is reasonable, since these algorithms attempt to minimize the service response time by selecting the closest servers to the user. Since users are located at different *UserDCs* as shown in the topology, their nearest servers are quickly utilized. Nevertheless, ORC, CEN, and STO are outnumbered by far by the RO placement algorithm, which has a much lower ServUtil rate as represented by the separate yellow line. It can be seen that ten concurrent services only use 3.6 servers on average. All servers are constantly utilized when there are 21 services instantiated at the same time. This maximum number of concurrent services doubles the other placement algorithms. This is due to the fact that the RO algorithm always prioritizes the collocation of VMs in one server before considering the others. Furthermore, when a SFC has finished execution and releases resources of the hosted server, but this server is still used by other services, it has a higher priority to be chosen

for the next incoming service than the unused ones. Based on this, the number of utilized servers is minimized and the other servers can be put to standby state. This can significantly reduce the power consumption and save energy. To conclude, it could be seen that the presented heuristic algorithms show a decent performance in terms of service response time, but still need improvements in terms of resource utilization.

VI. Conclusion

In the NFV paradigm, the usage of SFCs is promising to reduce the complexity of heterogeneous service deployment. Nevertheless, the distribution of VNFs over different hosts increases the overall latency and server utilization. In this work, we evaluate four algorithms to efficiently place SFCs in the context of an edge network. These algorithms aim to decrease the service response time or resource utilization. To evaluate the performance of these placement algorithms, we use the event-based EdgeNetworkCloudSim simulator.

Regarding service response time, the result show that STO performs better than the other algorithms in all types of service. This demonstrates that, the use of ILP model is able to compute an optimal solution. Especially, the probability of placing all VMs of a chain in one server is higher than CEN and ORC algorithms, which results in reduced service response time. However, the processing time of the optimizer is a considerable drawback as the placement problem is *NP-hard*. Despite of producing higher service response time than STO, ORC algorithm always tries to shorten the length of the chain. This algorithm can be an alternative for STO in a large network topology where the processing time of STO might be high. The CEN algorithm produces highest service response time, since it only places VMs as close as possible to the user without the consideration of the chain itself and the communication of the VMs within.

The second objective of the ILP model is to minimize server utilization. Herein, the placement of SFC is optimized to utilize the least number of server. Out result shows that, with the optimized placement, 10 concurrent SFCs only require a half of all server resources. In contrast, the heuristics and STO utilize all datacenters to handle the same number of concurrent SFCs. This insight shows that while the evaluated heuristics perform well in terms of service response times, they need to be improved to have the ability to reduce power consumption and carbon footprint of datacenters.

Future work may extend this study to evaluate these placement algorithms in different network topologies with more detailed investigation of bandwidth or energy consumption.

References

[1] Cisco Systems, "Cisco visual networking index: Forecast and methodology, 2016-2021," White Paper, 2016.

[2] M. Fiedler, T. Hoßfeld, and P. Tran-Gia, "A generic quantitative relationship between quality of experience and quality of service," *IEEE Network Special Issue on Improving QoE for Network Services*, 2010.

[3] ITU-T Rec. G1030, "Estimating end-to-end performance in ip networks for data applications," *ITU-T Recommendation*, Nov. 2005.

[4] M. Chios, D. Clarke, P. Willis, A. Reid, J. Feger, M. Bugenhagen, W. Khan, M. Fargano, C. Cui, and H. Deng, "Network functions virtualisation: an introduction, benefits, enablers, challenges and call for action," White Paper available at http://portal.etsi.org, 2012.

[5] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, 2015.

[6] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee, "Network function virtualization: Challenges and opportunities for innovations," *IEEE Communications Magazine*, vol. 53, no. 2, 2015.

[7] J. Halpern and C. Pignataro, "Service function chaining (sfc) architecture," Internet Requests for Comments, RFC, Tech. Rep. 7665, Oct 2015.

[8] W. John, K. Pentikousis, G. Agapiou, E. Jacob, M. Kind, A. Manzalini, F. Risso, D. Staessens, R. Steinert, and C. Meirosu, "Research directions in network service chaining," in *SDN for Future Networks and Services (SDN4FNS)*. Trento, Italy: IEEE, Nov 2013.

[9] M. Seufert, B. K. Kwam, F. Wamser, and P. Tran-Gia, "Edgenetworkcloudsim: Placement of service chains in edge clouds using networkcloudsim," in *IEEE Conference on Network Softwarization (NetSoft 2017)*. Bologna, Italy: IEEE, Jul 2017, pp. 1–6.

[10] M. Savi, M. Tornatore, and G. Verticale, "Impact of processing costs on service chain placement in network functions virtualization," in *IEEE Conference on Network Function Virtualization and Software Defined Network (NFV-SDN)*. San Francisco, CA, USA: IEEE, Jan 2015.

[11] M. F. Bari, S. R. Chowdhury, R. Ahmed, and R. Boutaba, "On orchestrating virtual network functions," in *11th International Conference on Network and Service Management*. Barcelona, Spain: IEEE, Nov 2015.

[12] 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 *IFIP/IEEE International Symposium on Integrated Network Management (IM)*. Ottawa, Canada: IEEE, May 2015, p. 98106.

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

[14] K. Yang, H. Zhang, and P. Hong, "Energy-aware service function placement for service function chaining in data centers," in *Global Communications Conference (GLOBECOM), 2016 IEEE*. Washington, DC, USA: IEEE, Dec 2016, p. 16.

[15] N. Huin, A. Tomassilli, F. Giroire, and B. Jaumard, "Energy-efficient service function chain provisioning," *Journal of Optical Communications and Networking*, vol. 10, no. 3, pp. 114–124, 2018.

[16] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, 2016.

[17] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011.

[18] B. Wickremasinghe, R. N. Calheiros, and R. Buyya, "Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications," in *24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*. Perth, WA, Australia: IEEE, Apr 2010, pp. 446–452.

[19] S. K. Garg and R. Buyya, "Networkcloudsim: Modelling parallel applications in cloud simulations," in *Fourth IEEE International Conference on Utility and Cloud Computing*. Victoria, Australia: IEEE, Dec 2011.

[20] C.-H. Hsieh, J.-W. Chang, C. Chen, and S.-H. Lu, "Network-aware service function chaining placement in a data center," in *18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. Kanazawa, Japan: IEEE, Oct 2016, p. 16.

[21] M. Bouet, J. Leguay, T. Combe, and V. Conan, "Cost-based placement of vdpi functions in nfv infrastructures," *International Journal of Network Management*, vol. 25, no. 6, 2015.

[22] M. Bouet, J. Leguay, and V. Conan, "Cost-based placement of virtualized deep packet inspection functions in sdn," in *MILCOM IEEE Military Communications Conference*. San Diego, USA: IEEE, Nov 2013.

[23] H. Moens and F. De Turck, "Vnf-p: A model for efficient placement of virtualized network functions," in *10th International Conference on Network and Service Management (CNSM)*. Rio de Janeiro, Brazil: IEEE, Jan 2014, pp. 418–423.

[24] R. Bruschi, P. Lago, and C. Lombardo, "In-network programmability for next-generation personal cloud service support (input)," *Procedia Computer Science*, vol. 97, pp. 114–117, 2016.

[25] A. Medina, A. Lakhina, I. Matta, and J. Byers, "Brite: An approach to universal topology generation," in *Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. Cincinnati, OH, USA: IEEE, Aug 2001, pp. 346–353.