EdgeNetworkCloudSim: Placement of Service Chains in Edge Clouds Using NetworkCloudSim

Michael Seufert, Brice Kamneng Kwam, Florian Wamser, Phuoc Tran-Gia

University of Würzburg, Institute of Computer Science, Chair of Communication Networks, Würzburg, Germany

{seufert | florian.wamser | trangia}@informatik.uni-wuerzburg.de, brice.kwam@stud-mail.uni-wuerzburg.de

Abstract—Edge cloud computing is a trending paradigm, which extends cloud computing by additionally utilizing computing resources at the network edge, e.g., at mobile base stations. Especially personalized services can be instantiated or migrated close to end users, which improves the latency and supports user mobility. However, the placement of the service chains is crucial for the performance of the services and the energy consumption of the edge cloud platform, and appropriate algorithms have to be designed. To support the simulative performance evaluation of such algorithms, EdgeNetworkCloudSim was developed. It is an extension of NetworkCloudSim, and allows to simulate and evaluate the orchestration and consolidation of service chains in an edge network cloud.

Index Terms—Edge computing, edge cloud network, simulation, performance evaluation, service chain, placement, orchestration, consolidation.

Fig. 1: INPUT service chains

I. INTRODUCTION

Cloud computing relies on sharing computing resources on demand, which allows to elastically scale the resource utilization according to the workload. Most of the cloud workloads are applications (software as a service), which have high computing and network requirements. As the usage of cloud computing is trending, also the network loads due to data center traffic are increasing [1]. However, clouds, which are highly centralized data centers in the core network, are not optimized for personal services consumed by (mobile) end users, e.g., in terms of low latency. Therefore, the new paradigm of edge cloud computing aims to extend cloud computing by additionally utilizing computing resources at the network edge, e.g., at mobile base stations. Virtualized cloud services can be instantiated or migrated at the network edge close to end users to offer personalized services, support user mobility, and achieve low latencies.

The EU H2020 project INPUT (In-Network Programmability for next-generation personal cloUd service supporT [1]) aims to provide a programmable platform for fully virtualized, personalized cloud services at the network edge, which optimizes both power and performance, and vertically integrates with cloud computing. The platform is instantiated on distributed network infrastructure (INPUT nodes) and manages the utilization of the physical resources and the deployment of services. When a new personal cloud service is requested by an end user, the whole service chain, composed of the
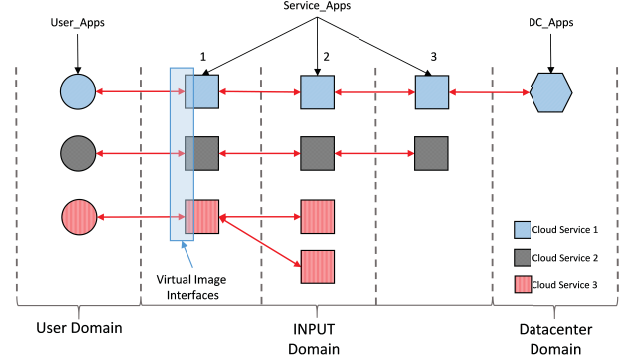
specific *service apps* and the related back-end network, must be deployed in the edge network, which is illustrated in Figure 1. The single service apps composing the service chain are implemented as virtual machines (VMs), which can be instantiated and migrated to any INPUT node. Moreover, each service chain contains a *user app* on an end device of the user, which communicates with the service apps from the user's personal network, and optional *datacenter apps*, which can run in traditional clouds in the core networks.

The placement of the service apps, i.e., the allocation of VMs to physical servers, of the needed service chains highly influences the performance of the services and the energy consumption of an edge cloud platform. Therefore, appropriate algorithms for the orchestration and consolidation of service chains have to be designed and evaluated. As the complexity of a sufficiently large edge cloud platform soon exceeds the applicability of analytic evaluation methods or hardware testbeds, simulative performance evaluation is a suitable alternative. This paper presents EdgeNetworkCloudSim, which was developed in the INPUT project based on NetworkCloudSim [2]. It allows to simulate and evaluate the placement, orchestration, and consolidation of service chains in an edge network cloud.

Section II presents related simulation frameworks. Section III describes the extensions of NetworkCloudSim to support service chains and edge networks. Section IV presents the parameters and metrics of the simulation framework, and describes an exemplary performance evaluation. Finally, Section VI concludes.

II. Related Work

For cloud computing, a few simulation frameworks have already been developed. In the following, an overview of open source frameworks is given.

GreenCloud [3] is a powerful simulation tool to observe, measure, and interact with cloud resources. GreenCloud was developed for energy-aware data centers and is an extension of the well-known NS2 Network Simulator [4]. It is primarily focused on communications within the cloud, e.g., all communication processes are simulated at the packet level.

iCanCloud [5] can be used to predict the trade-offs between cost and performance of a given set of applications executed in a specific hardware. It simulates instance types provided by Amazon EC2 [6] and customized VMs. This framework has a flexible cloud hypervisor module and provides a user-friendly GUI to ease the generation and customization of large distributed models. New components can be added to the repository of iCanCloud to extend its functionality.

CloudSim [7] provides a generalized and extensible simulation framework that enables modeling, simulation, and experimentation of emerging cloud computing infrastructures and application services. It has some pre-defined policies for allocation of host resources to virtual machines, but also supports user-defined policies. During the simulation, components can be dynamically inserted, stopped, and resumed. This framework supports virtualized server hosts, energy-aware computational resources, and message-passing applications, but lacks a GUI. CloudAnalyst [8] is an extension of the CloudSim framework, which can be used to model and evaluate large-scale cloud applications in terms of geographic distribution of both computing servers and user workload, such as networking applications. It provides geolocation for data centers and user distribution on a map of the world. The simulation is made with the purpose of studying the behavior of such applications under various deployment configurations. It has a very easy to use GUI and the simulation can be configured with a high degree of flexibility. The experiments can be easily repeated since the configuration can be saved in a file and restarted by loading that file. Moreover its output is graphical, which makes its analysis very easy.

NetworkCloudSim [2] is an extension of CloudSim, and implements the network layer in CloudSim to also simulate network traffic. The drawback is that no GUI is available. However, as NetworkCloudSim brings all the features of CloudSim, and additionally allows to simulate the underlying network, it was selected as the base of the edge network cloud simulation framework.

III. Extension of NetworkCloudSim

Before getting into the actual work, it is important to understand how NetworkCloudSim works. Therefore, this section first introduces the most important components of NetworkCloudSim. More details can be obtained from the original publications [2], [7].

A **NetworkDatacenter** encapsulates a set of computing hosts. **Switch** represents a network entity which can be configured as a router or as a switch. Currently, to allow to create various topologies, three types of switch are available: root, aggregate, and edge switch. The edge switch is directly connected to hosts. An aggregate switch connects different switches. The root switch connects the datacenter to the Internet. The network can be configured either through a BRITE [9] file or through explicit generation of links between the entities. Two types of packets can be sent. **HostPackets** are transmitted in the virtual network of a single host, and **NetworkPacktes** are sent from one server to another.

NetworkHost models a physical computer or storage server. It hosts and manages **NetworkVMs**, which represent virtual machines (VM). They are allocated to hosts according to a **NetworkVmAllocationPolicy**. **AppCloudlet** represents an application with multiple tasks (NetworkCloudlets). A **NetworkCloudlet** is a task with various stages of execution, which runs in a VM. There are four stages, namely, send, receive, execute, and finished. They allow the NetworkCloudlet to send or receive packets to/from other NetworkCloudlets, and process data itself, which can be configured and implemented by an event-based logic according to custom requirements. To run an application, the **NetworkDatacenterBroker** schedules the NetworkCloudlets on different VMs. **NetworkCloudletScheduler** implements a policy, which determines the sharing of processing power among NetworkCloudlets in a VM.

A. Requirements for Edge Computing Platform

In order to simulate an edge computing platform in the INPUT project, the simulator must fulfill some additional requirements, which are described in the following.

Services have to be supported, which consist of several service apps (VMs) that exchange and process data in a specified order. This means, services are implemented as service chains, and in the remainder of this paper, these terms will be used interchangeably. Policies have to be implemented, which allocate or migrate service chains according to service-level agreements or energy efficiency. Users have to be considered, which start and interact with the services (inter-arrival time distributions of service request or service interaction) and possibly change their locations (user mobility). Services can be used by users multiple times before the user terminates the service (service time distribution). Statistics have to be collected and exported regarding the service chain allocation, host and network utilization, and energy consumption. They can be used to evaluate the performance of different orchestration and consolidation policies and investigate trade-offs.

B. Implementation

The first implemented extensions introduce service chains, one user, and new statistics to NetworkCloudSim. Many classes of NetworkCloudSim needed only minor modifications, so they will not be discussed here. The remainder of this section will only explain the major modifications, and the features and capabilities they bring into the framework. Note that multiple users, probabilistic user interactions, and mobility of users were not implemented yet, but can be easily

added in the next extension phase. The current source code of EdgeNetworkCloudSim is available on GitHub[2] and can be used under the Apache License, Version 2.0.

The **EdgeService** class represents a service chain that can be started by a user. The service chain defines the NetworkCloudlets and the corresponding task stages. Thereby, NetworkCloudlets are bound to one VM each and consume the allocated resources of the VM exclusively, which is assured by using a modified space-shared NetwockCloudletScheduler of NetworkCloudSim. The NetworkCloudlets of the chain have typically the following six stages: receive, execute, send, receive, execute, send. First, the request is received from the user or the previous NetworkCloudlet. Then, some data can be processed locally. A new request is sent to the next NetworkCloudlet (if any). After all NetworkCloudlets of the chain have forwarded the requests, the responses traverse the chain backwards. Thus, a NetworkCloudlet receives the response, it can again compute something locally, and then send a response to the previous NetworkCloudlet or the user. Thus, the user only has to start a service chain and send a request to the first NetworkCloudlet. Then, the chain will process the request as described above and return the result to the user. The EdgeService also takes care of the VM management for the given service and starts the required VMs (one per NetworkCloudlet) in the appropriate data centers according to the predefined allocation policy. Therefore, different VM allocation policies were implemented, which consider proximity to the user and available CPU, RAM, storage, and bandwidth.

The **EdgeDatacenterBroker** class represents a broker, which mediates the actions of a user. In NetworkCloudSim, the broker was responsible for the VM management (creation, destruction, migration) as well as the scheduling of NetworkCloudlets. In EdgeNetworkCloudSim, this is now the responsibility of the service chain. However, the broker is now responsible for the service management. This means, it creates and starts the user VM and the services of a user and sends requests to the services on behalf of the user.

To normalize the available VMs, different **EdgeVms** were predefined and can be used to specify the size of service apps easily. Several **VmTypes** based on the virtual machines available on Amazon Elastic Compute Cloud (Amazon EC2) were implemented as listed in Table I. Nevertheless, other custom VMs types can be added. Each VM runs a single specific NetworkCloudlet. For each of these NetworkCloudlets, the number of instructions, which have to be executed, and the amount of data, which is sent by each of them, can be specified or computed randomly.

After the deployment of the VMs on different hosts, the packets of requests and responses have to be routed to the destination VM (i.e., VM of next/previous NetworkCloudlet in the service chain, or user VM). Therefore, the Floyd-Warshall algorithm [10] was implemented to calculate the delays between all data centers and the next hop matrix. At

[2]https://github.com/lsinfo3/EdgeNetworkCloudSim, accessed: 2017-05-03

TABLE I: VM Type Definition

VM Type	CPUs	RAM
T2Nano	1	512 MB
T2Small	2	2 GB
T2Large	4	6 GB
M4XLarge	8	16 GB

the beginning of the simulation, after loading the BRITE file, a global (next hop) routing table of the complete topology is computed and stored. This computation is done only once, which allows a linear reconstruction of the paths on demand, minimizing the memory consumption.

IV. SIMULATION ENVIRONMENT, PARAMETERS, AND METRICS

The main purpose of this section is to explain how this framework is to be used. First, the simulation environment is described. Then, the specific parameters, which can be configured, are presented. Finally, the results are outlined, which can be obtained from a simulation run.

A. Simulation Environment

As this framework is an extension of NetworkCloudSim [2], which was implemented in Java, a Java Development Kit (JDK) and optionally an integrated development environment (IDE) are needed to configure the simulation. Java is not specific to any processor or operating system, thus, EdgeNetworkCloudSim can be run on mostly all operating systems (Linux/Mac/Windows) with a Java Runtime Environment (JRE). Simulation runs can be started on a console or within an IDE.

B. Parameters

In order to use this framework, the user first has to set up the investigated system and its entities, and define the evaluation scenario. First of all, the edge resources have to be defined. Therefore, the edge data centers (DC) are created. Each DC has a name, a number of hosts, which have predefined resources (i.e., CPU, RAM, storage), and a VM allocation policy. The internal network of the edge resources has to be defined as well. For this purpose, the type and number of switches that are going to be used as well as the links between them have to be specified. The links and their characteristics (i.e., capacity and delay) are defined in a BRITE file, and then the end points of the defined links have to be mapped to the simulation entities (hosts and switches).

The next step is to define the user, who is represented by the EdgeDatacenterBroker, and the services he intends to start in the edge cloud. For simulating the user, a dedicated user DC has to be created to ensure that only the user VM but no service VMs are allocated in that DC. Then, the services have to be defined and added to the service list of the broker. The service chain of each service has to be manually set up by specifying the VM type and their order in the service chain, creating the corresponding NetworkCloudlets, defining the task stages of each NetworkCloudlet. This includes the definition of the

number of instructions, which have to be processed in the execution stages, and the amount of data, which are sent in the send stages of each NetworkCloudlet. These numbers can be set statically or follow probability distributions.

As the implementation of the framework is ongoing, the framework will support multiple users and user mobility in the next extension phase. Therefore, several dedicated user DCs will be instantiated, which can host many user VMs. These user VMs can optionally be moved among the user DCs according to mobility patterns. Moreover, the current static service instantiation and interaction will be extended to support also probabilistic user interactions. This means, services can be started by users according to specific service inter-arrival time distributions and these services usage time follows specific service time distributions. Also within these service usage phase, the requests, which are sent by the user to the service, can follow certain request inter-arrival time distributions.

C. Metrics

The goal of a simulation is to evaluate the performance of orchestration or consolidation policies in different scenarios. After the scenario has been defined, the simulation runs are performed, and several metrics can be obtained. This section explains what kind of metrics are produced and what they can be used for.

Several information about the simulation environment itself are collected to check the orderly execution of the simulation runs. This includes the start and end time of the simulation run, whether the entities (hosts and switches) were initialized, and the network was successfully activated. Also the infrastructure of the edge cloud is monitored. Therefore, the hosts and links regularly log their utilization. This allows to compute the energy consumption of the current allocation by adding energy models for the simulated entities.

The framework also monitors the service placement and request processing. Therefore, it logs the checks that were performed to allocate a VM according to the VM allocation policy and if and why the allocation failed for the checked hosts (e.g., due to insufficient CPU or RAM on the target host). Finally, it stores the data center in which the VMs of a service were created, or whether VMs, and thus, the whole service chains, could not be instantiated. If a service chain was successfully placed, EdgeNetworkCloudSim monitors the start and end of each VM, as well as each stage of the NetworkCloudlets. Thereby, it can be monitored how long each request was processed by a single service app or the whole service chain. Finally, the EdgeDatacenterBroker logs the times and types of the user interactions, i.e., the start of a service and the corresponding requests and responses.

As the framework does not have a GUI, the logs are output to the console or to a text or CSV file, depending on the needs and the way the data will be evaluated. A built-in custom logger gives the user the possibility to redirect the output to a file in a custom format.

V. EXEMPLARY PERFORMANCE EVALUATION

In order to show how the framework shall be used, this section presents a simple example. Figure 2a shows the scheme of the edge network topology for this example. This topology contains five data centers (DCs), four of the DCs are for running the services and one DC for the user himself. To simulate the user, he has to be located in a dedicated DC, which hosts the user VM for sending requests to the services in the edge cloud network. For simplicity, each DC contains one host with 4 CPUs, 8 GB RAM, and 1 TB storage. The interconnection of the edge cloud network has to be implemented by choosing the appropriate switch and link types. The hosts in each DC are connected to an EdgeSwitch with a 10 Gbps connection with 0 ms delay to model high speed connections within DCs. The DCs are connected to AggregateSwitches with a 50 Gbps cable and 10 ms delay, and the AggregateSwitches are interconnected with 100 Gbps links and 50 ms delay. The resulting implemented topology can be found in Figure 2b. It ensures that on each link of the scheme of the network topology, the simulated packets traverse one switch in the implemented topology.
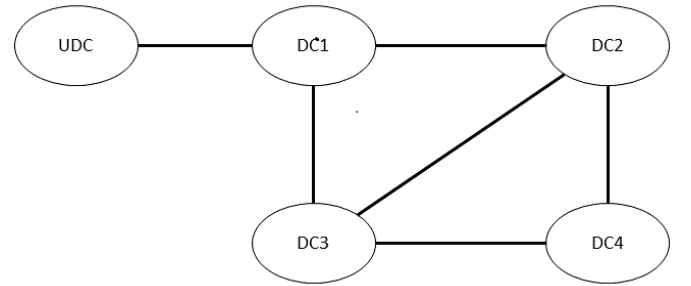
(a) Scheme of network topology
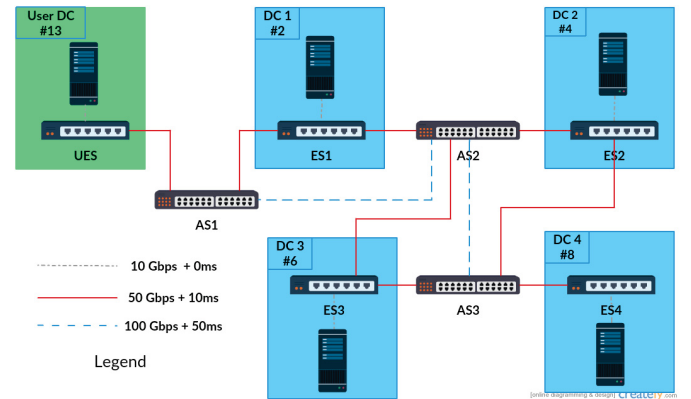
(b) Implemented network topology

Fig. 2: Network topology of exemplary evaluation scenario

Listing 1 shows the BRITE file used to configure this network. It contains 13 nodes and 15 edges (some lines of the file have been left out for simplicity). The list of nodes contains the node ID in the first column, which is the only field used by the simulation framework at the moment. The list of edges has an edge ID in the first column, the source

node ID in the second column, and the destination node ID in the third column. Moreover, the delay (in ms) and capacity (in kBps) are specified in the fifth and sixth column respectively. The other fields of the BRITE file are not used by EdgeNetworkCloudSim yet, but can be later used to specify the x and y position of nodes, the indegree and outdegree, the autonomous systems, or the node and link types [9]. The entities then have to be defined and mapped to the nodes configured in Listing 1.

Listing 1: Excerpt from BRITE network configuration file

```
Topology: ( 13 Nodes, 15 Edges )
...

Nodes: ( 13 )
0 1 1 1 1  1  RT_NONE
1 1 2 2 2  1  RT_NONE
2 2 3 3 3  1  RT_NONE
...

Edges: ( 15 )
0 0 1 3.16 0.0   1250000.0    1  1  E_RT U
1 1 2 5.09 10.0  6250000.0    1  1  E_RT U
2 3 4 1.41 0.0   1250000.0    1  1  E_RT U
3 2 4 6.00 10.0  6250000.0    1  1  E_RT U
4 2 5 5.09 50.0  12500000.0   1  1  E_RT U
...
```

In this exemplary performance evaluation, two service chains will be placed consisting of VMs defined in Table I. The service chain of Service 1 consists of two sequential T2Nano VMs followed by a T2Small VM, Service 2 uses a T2Small, a T2Nano, and a T2Large VM in this specified order. The user will request Service 1 (A), Service 2 (B), and a second instance of Service 1 (C). The policy to place the VMs in the edge network cloud is the `VmAllocationPolicyCpu`, which will try to allocate the VMs of a service chain one after another. All data centers are checked if they can host the VM according to its requirements. Then, the placement policy will prefer the data center with the lowest delay and the host with the smallest number of CPUs allocated. Since the user and his corresponding broker are located in the user DC (#13), the closest data center is DC #2, in which all services try to start their VMs first. Service A can be completely allocated to DC #2 as the resources of the host are sufficient for all three VMs of the service. The second service chain (B) fails to allocate its first VMs in DC #2 because no more CPUs are available. Thus, it places the T2Small and the T2Nano VMs on DC #4, which is the next close DC (together with DC #6). The T2Large VM needs 4 CPUs, which are not available on DC #4, but on DC #6. Finally, the third service chain succeeds to start its first T2Nano on DC #4, and the other two VMs have to be started in DC #8. An overview of the placement is given in Table II.

After the start of the services, the user can now send requests to the services. In order to show the impact of the network

TABLE II: Placement of VMs in data centers

Entity	VM Type	VM ID	DC ID
User	T2Nano	0	13
Service A	T2Nano	1	2
	T2Nano	2	2
	T2Small	3	2
Service B	T2Small	4	4
	T2Nano	5	4
	T2Large	6	6
Service C	T2Nano	7	4
	T2Nano	8	8
	T2Small	9	8

configuration in the framework, the service time of the baseline configuration and three modified configurations are compared. Note that only one simulation run was conducted per configuration because this exemplary performance evaluation is completely deterministic. This means, in each simulation run, the user sent one request to each service, and the services processed a deterministic number of instructions in each VM. Also the amount of data transmitted between the user and the service chain, and between the VMs within a service chain is fixed to 1 GB. This deterministic evaluation scenario allows to explicitly calculate the service times of each service depending on its placement, and thus, to verify the simulation framework. In a real performance evaluation, multiple mobile users have to be considered, as well as probabilistic service requests and request sizes.

Figure 3a shows that the baseline service times of all three services (black bars) are similar but slightly increasing from service A to C. This is expected as more links have to be traversed by packets of services, which are more distant to the user (service B and C). The figure also shows a comparison between the baseline and a configuration, in which the user and services have to send twice as much data (2 GB) as in the baseline configuration (1 GB, yellow bars). It can be seen the service time of each service increases, because the data transmission in the network takes longer.

Also when the link delay is doubled, the packets need more time on each link. This affects all service times negatively, which is shown in Figure 3b. Again the baseline is depicted in black, while the results of the modified configuration are plotted as yellow bars. It can be seen that service A is least affected as only four transmissions (from user DC via AggregateSwitch 1 to DC #2 and back) are needed. In contrast, packets of service B and C traverse much more links, which causes the higher increase of the service time.

Finally, Figure 3c compares the baseline (black) to a configuration with double link bandwidths (yellow). The bars show that the service times slightly improve. Services, which send packets over more links, benefit more from the double bandwidth. Note that for all service times presented in Figure 3, the obtained results from the simulation exactly matched the computed service times of this deterministic performance evaluation scenario.

To sum up, this exemplary performance evaluation showed how to configure EdgeNetworkCloudSim for a specific evalu-
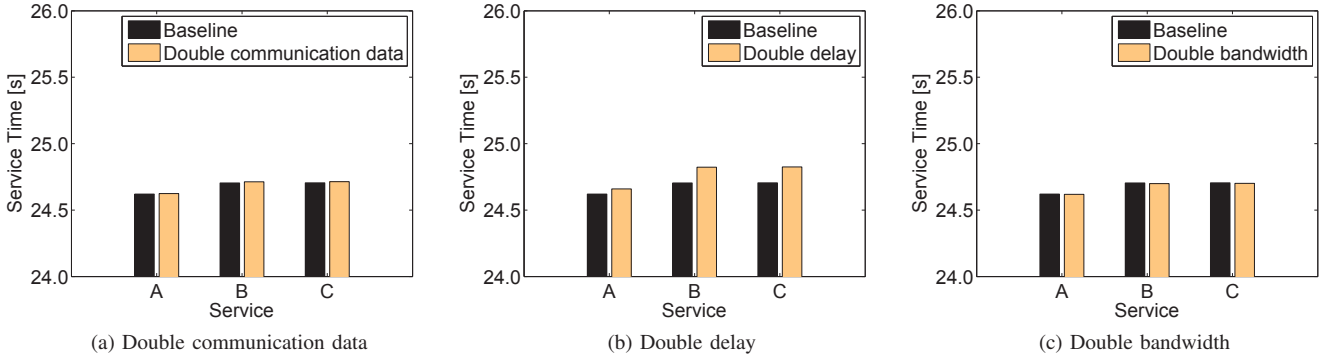
Fig. 3: Comparison of service times for baseline configuration and modified configuration

ation scenario. The obtained evaluation results confirmed that placement algorithms for edge clouds have a huge impact on the overall performance of an edge cloud. Already for this simple example with a single user and static service requests, the placement of a particular service chain determined how different network conditions differently affected the service times. In future work, more complex simulative evaluations have to be conducted, which investigate the performance of edge service placement algorithms in more detail, for example, in terms of resource and energy utilization of hosts, service quality, and placement algorithm computation time. Therefore, EdgeNetworkCloudSim proved to be a valuable tool, which can be easily customized to the needed evaluation scenarios.

VI. CONCLUSION

This paper presented EdgeNetworkCloudSim, an extension of NetworkCloudSim [2], which allows to simulate the orchestration and consolidation of service chains in an edge cloud environment. The extension introduces users, service chains, and service request processing, which is implemented as a combination of local processing and forwarding of requests and responses along the service chain. The framework allows to specify the edge cloud resources, the network topology, the user interactions, and the VM allocation policies. The results of a simulation run allow to evaluate the resource utilization and service processing according to energy efficiency and service-level agreements.

An exemplary performance evaluation has been presented to show how the simulation framework can be configured. Moreover, the impact of different network configurations on the service times of the placed services was investigated. It could be seen that, depending on the particular placement of a service chain, the service time was impacted differently by the different network configurations. Thus, placement algorithms for service chains have a huge impact on the overall performance of an edge cloud, and need to be evaluated and optimized.

As the implementation of EdgeNetworkCloudSim is ongoing, the current implementation only supports one user and static service requests. In the next extension phase, which will be published in a few months, the support for multiple users,

probabilistic user interactions, and mobility of users will be added. Then, the framework will be a valuable tool, which is used in the INPUT project to compare the performance of placement strategies for service chains on edge clouds. The goal is to assess the different orchestration and consolidation approaches (e.g., heuristics, linear programs) in terms of run time to compute the placements, and goodness of placements, i.e., trade-offs between service quality and energy efficiency.

REFERENCES

[1] Cisco, "Cisco Global Cloud Index: Forecast and Methodology, 2015–2020," Cisco, Tech. Rep., 2016.

[2] S. K. Garg and R. Buyya, "NetworkCloudSim: Modelling Parallel Applications in Cloud Simulations." in *4th IEEE/ACM International Conference on Utility and Cloud Computing (UCC)*, Melbourne, Australia, 2011.

[3] D. Kliazovich, P. Bouvry, and S. U. Khan, "GreenCloud: A Packet-level Simulator of Energy-aware Cloud Computing Data Centers." *Journal of Supercomputing*, vol. 62, no. 3, pp. 1263–1283, 2012.

[4] "The Network Simulator Ns2," http://www.isi.edu/nsnam/ns/, accessed: 2017-03-20.

[5] A. Nuñez, J. L. Vázquez-Poletti, A. C. Caminero, G. G. Castañé, J. Carretero, and I. M. Llorente, "iCanCloud: A Flexible and Scalable Cloud Infrastructure Simulator." *Journal of Grid Computing*, vol. 10, no. 1, pp. 185–209, 2012.

[6] "Amazon EC2 Intances," https://aws.amazon.com/de/ec2/instance-types/, accessed: 2017-03-20.

[7] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. D. 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.

[8] 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, Australia, 2010.

[9] A. Medina, A. Lakhina, I. Matta, and J. W. Byers, "BRITE: An Approach to Universal Topology Generation," in *9th International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Cincinnati, OH, USA, 2001.

[10] R. W. Floyd, "Algorithm 97: Shortest Path," *Communications of the ACM*, vol. 5, no. 6, p. 345, 1962.