MUTE: MUlti-Tier Edge networks

A. Silvestro [*] [†] N. Mohan [*] [‡] J. Kangasharju [‡] F. Schneider [†] X. Fu [⊥]

NEC Laboratories Europe, Germany [†] University of Helsinki, Finland [‡] University of Goettingen, Germany [⊥]

Abstract

Several Internet-of-Things (IoT) based use-cases require an increased amount of computation resources with extremely low latency. Cloud-based services may fail to provide such requirement given the high latency required to access their facilities. Therefore, an increasing number of resources are being deployed at the network "edge" as *Edge Clouds*.

However, as the modeling of edge network is still in its early stages, the existing solutions for service placement and resource utilization are found to be quite inefficient. In this paper, we model a *multi-tier* edge network and propose a service placement algorithm, *Mute*. In our evaluation, performed on real network topologies, we show that *Mute* achieves 66% reduction in network cost and 50% reduction in service placement when compared to state-of-the-art solutions.

Keywords Edge clouds, in-network services, service function chains, edge service placement

1. Introduction

In recent years, several application use-cases, requiring high data availability and quick computation, such as Internet-of-Things (IoT), vehicular networks, etc. have proliferated to a great extent. Such applications require computational resources that can handle highly variable data with stringent completion time requirements. The traditional centralized cloud model is unable to support these use cases due to possibly high network delays encountered while offloading data to the location of cloud data centers. Researchers have proposed decoupling the traditional cloud model to several smaller computation resources installed closer to data generators [1]. Due to their proximity to the network "edge", these collections of resources are termed as Edge cloud [2].

[*] Joint first authors

CrossCloud'18, April 23, 2018, Porto, Portugal.

http://dx.doi.org/10.1145/3195870.3195871

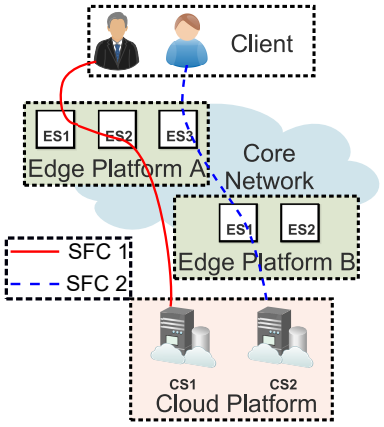


Figure 1: Service Function Chains (SFC) over Edge Cloud

In the past, several edge cloud models have been proposed [3–6] to decouple network delay from computation time in concrete deployments. Telecom operators have also adopted such models, for instance, models such as Mobile Edge Computing (MEC) enable edge servers and cellular base stations to be operated simultaneously [7]. Recently, significant improvements in MEC have enabled Mobile Network Operators (MNO) to integrate 5G telecommunication in the cloud platform itself [2]. However, at this stage, only proprietary edge services can be deployed by MNOs MEC instances, and the platforms are not open to third-party providers. Other research proposals have considered open-to-all edge cloud instances to be composed of possible community driven compute resources which drastically increase the density and variability of the edge. For instance, Mohan et al. [8] present a model where the edge cloud is composed of a combination of voluntary compute resources such as mobile phones, workstations, etc. and managed micro-cloud instances such as mini-datacenters.

Existing service/task placement and resource selection algorithms attempt to map multiple services on a set of homogeneous cloud resources with a consistent network delay from the clients [9]. Figure 1 shows an end-to-end Service Function Chaining (SFC) deployment on the edge. However, the significant heterogeneity of the edge resources, regarding processing capability and network distribution, necessitates re-designing such placement algorithms, to make them match better with edge computing environments.

In this paper, we provide the following contributions.

1) We define a use case scenario for the Edge where Edge Platform Providers open their infrastructure to third-party Service Providers as shown in Figure 2. Building on the success of other open systems, with the Internet and Web being prime examples, we conjecture that a similarly open approach will enable edge computing to flourish. Existing solutions, such as auctioning strategies [10] can be used to allow platform providers to run their systems for profit.

2) We define a model of *multi-tier edge network* in which edge resources are logically clustered into distinct tiers, based on their characteristics such as processing capabilities, network delay from the client, etc. (shown in Figure 2). This characterization enables Edge Service Providers, to efficiently manage their governed edge resources, and to find more solutions optimized for multi-tier scenarios.

3) We design *Mute*, a placement algorithm which leverages multi-tier edge architecture to find an edge server which best supports the needs of a requested service.

4) We perform an extensive set of simulations using real network topologies [11]. We show that the *Mute* algorithm achieves 66% reduction in network cost, when compared to state-of-the-art non-edge aware placement algorithms. Additionally, *Mute* leverages the multi-tier structure to achieve the service placement up to 50% faster, when compared to non-tier aware placement algorithms.

The rest of the paper is organized as follows. We discuss the architecture and stakeholders in an edge network in Section 2 and present Multi-Tier Edge architecture and *Mute* algorithm in Section 3. Section 4 evaluates *Mute* with state-of-the-art algorithms on realistic topologies. Section 5 presents related work and use-cases of proposed edge architecture. We conclude our paper in Section 6.

2. Architecture & Stakeholders

In Figure 1, we show the architecture and the stakeholders of an edge cloud. We envision a model where several edge platforms co-exist on the network. An end-to-end connection is established between the client and the cloud platform. Multiple services can be deployed on the edge servers which can enrich client's connection with the cloud (e.g., video transcoder, web proxy, etc.). The resulting *Service Function Chain (SFC)* is routed through deployed services which can either be placed on servers co-located in the same facility or in different platforms and location. Based on the ownership of resources, the edge cloud model has three primary stakeholders: Clients, *Edge Service Providers (ESP)* and *Cloud Service Providers* (shown in Figure 2).

Clients establish the connection with a *Cloud Server (CS)* in a *Cloud Platform (CP)*. The clients and/or cloud can request to include virtualized services deployed at the edge via SFC. The client can have either an *active* or a *passive* role in the SFC traffic steering depending if the SFC is transparently enforced [12], for instance by the network operator, or is explicitly defined by the client or the server [13, 14].

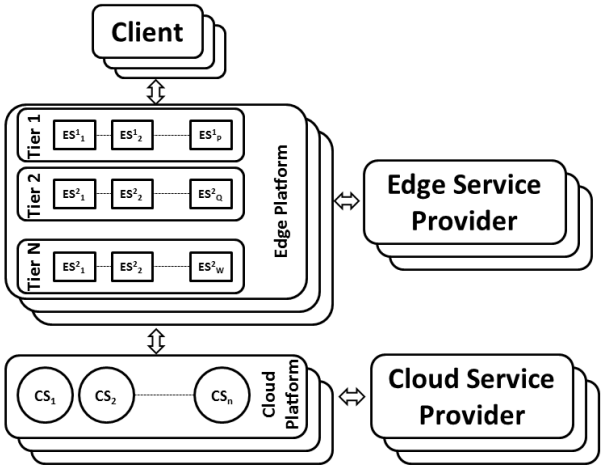


Figure 2: Multi-tier Edge Architecture & Stakeholders

Cloud Service Providers (CSPs) consolidates several CS deployed on a CP. Individual CSs are grouped together in data centers (DCs) facilities, which are distributed at various locations in the network [15]. Upon arrival of a connection request from a client, the CSP reroutes the request to one of the available CSs in the CP.

Edge Service Providers (ESPs) act as an intermediate entity between clients and CSPs. An ESP hosts several computation and storage capable *Edge Servers (ES)* federated into *Edge Data Centers (Edge DCs)*. Unlike the cloud DCs, the Edge DCs are spread over a broader geographic region and have a significantly lower latency to connect to a client in proximate location.

3. Edge Platform Modelling and Deployment

The edge cloud model discussed in Section 2 considers the interaction between the involved stakeholders. In this section, we discuss the techniques by which the ESPs can model its governed Edge Platform and select the best ES to deploy requested service functions.

3.1 Multi-Tier Edge

Past state-of-the-art research has proposed grouping ESs in an EP wherein all servers have similar characteristics [5]. The ESP can then employ a selection algorithm which "picks" the best ES capable of supporting the processing and number of users[1] required by the requested service.

We propose *multi-tier edge* platform, which further categorizes resources within a platform in *tiers* depending on their network delay from the edge. Figure 2 illustrates platform's architecture. The properties of the ESs in resulting architecture varies from tier-to-tier.

[1] We denote the number of users that can be supported by a server as its *bandwidth* throughout the paper.

I) Network Delay: the network delay to the edge is the primary attribute for classifying ES in tiers. The tiers which are closer to the network edge are composed of ESs with lower network delay to clients than higher tiers. However, the network delay of ES to end-client is location-dependent; e.g. Tier 1 ESs in New York and California will have the least network delay to clients located in their proximate locations but will have a very high delay for the other's location.

II) Bandwidth and Processing Power: edge servers in lower tiers have limited processing capability and can only support a restricted number of users simultaneously. However, their proximity to the network edge makes them desirable to deploy virtualized services. Tiers closer to the cloud are composed of ESs with higher processing capability.

III) Number of Servers: lower tiers are characterized by a greater number of small edge servers, that show limited resource capability. On the other hand, tiers which are closer to the cloud, have a small number of more powerful servers.

3.2 Network Structure & Model Definition

Here we formulate the problem of placing services on an Edge Platform (EP). Figure 3 shows the network architecture of a multi-tier Edge Platform composed of three ES with different bandwidth and processing capability. Each server has a network connection to the client and the cloud platform of a certain weight. The EP is composed of multiple Edge Servers (ES) of different attributes, i.e. $EST = ES_1, \ldots, ES_r$ where EST denotes set of all ES physical machines. As discussed in Section 3.1, the Edge Service Provider (ESP) categorize ES into tiers $t = t_1, \ldots, t_n$ wherein each ES in $EST \in t$. The resulting ES are grouped in n tiers as:

$$ES^n = \{ES^n_q, ES^n_{q+1}, ES^n_{q+2} \ldots, ES^n_r\}$$

where

$$EST = ES^1 \cup ES^2 \cup \ldots \cup ES^n$$

The Edge tier ES^1 lies closest to clients $C = C_1, C_2, ..., C_m$ and the last tier ES^n is closer to the end server/cloud. Each ES physical machine has a maximum resource utilization denoted by $ES^{proc} = ES^p_1, ES^p_2, \ldots, ES^p_r$ and maximum bandwidth as $ES^{bw} = ES^{bw}_1, ES^{bw}_2, \ldots, ES^{bw}_r$. As shown in Figure 1, ES in lower tiers have smaller processing and bandwidth capability but also have a significantly lower network cost to clients.

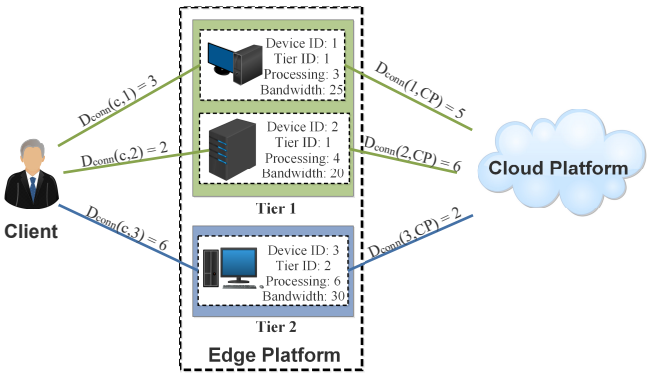


Figure 3: Multi-Tier Edge Network Architecture

3.3 Placing Services on the Edge

Let $S = S_1 \ldots, S_m$ denote the set of $'m'$ services to be placed on EST. The service providers want to enrich the experience of their clients and overall end-to-end connections by deploying services on ES closest to the client. We define services to have different processing requirements (in terms of the number of CPU-cycles) required for their execution denoted as $S^{proc} = S^p_1, S^p_2, \ldots, S^p_m$. Similarly, according to the Service Level Agreement (SLA), each service must support a bandwidth quota denoted as $S^{bw} = S^{bw}_1, S^{bw}_2, \ldots, S^{bw}_m$.

Furthermore, every Edge server ES has an associated cost for deploying service S_j on physical machine (PM) ES_i per unit time denoted as c_{ij}. The cost is dependent on ES's capability (regarding processing and networking) for running the service and the tier-level it belongs.

$$EST^c = \{ES^c_1, ES^c_2, \ldots, ES^c_r\} \quad ES \text{ deployment cost}$$

The resulting pricing model assigns more cost to low tier ESs due to their limited processing capabilities and lower network delays.

We denote variable x to indicate whether a service S_i is deployed on the edge server EST_j.

$$x_{ij} = \begin{cases} 1, & \text{if } S_i \text{ deployed on } ES_j. \\ 0, & \text{otherwise.} \end{cases}$$

Considering the placement requirements, the deployment algorithm can optimize the following attributes, i) the networking delay between client and the service, and ii) operational cost of deploying a service on ES with certain processing capability.

3.3.1 Minimizing Operational Cost

As discussed in Section 3.1, a low tier ES is likely to have a higher cost of deployment compared to a higher tier server. The operational cost for deploying service S on the Edge Platform EST can be formulated as

$$C(\lambda) = \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{S^{proc}_i}{ES^{proc}_j} ES^c_j \, x_{ij} \tag{1}$$

subject to

$$0 < \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{S^{proc}_i}{ES^{proc}_j} < 1 \tag{2}$$

$$\sum_{j}^{n} x_{ij} = 1 \qquad \forall ii \in S = S_1, \ldots, S_m \tag{3}$$

The deployment algorithm minimizes Equation 1 to optimize total cost of processing a virtualized service on an ES. The constraint in Equation 2 ensures that the ES has enough processing capability to host the requested. Equation 3 guarantees deployment of all requested services.

3.3.2 Minimizing Network Delay

Considering that the requested service is to be deployed on an ES member of an EP (shown in Figure 3), the resulting end-to-end connection between client and cloud will be composed of the nodes client, ES and cloud. We denote network link between client to Edge Server (ES_i) hosting the service and ES to the cloud as denoted as $d_{conn}(c, ES_i)$ and $d_{conn}(ES_i, cloud)$ respectively. The end-to-end network cost can be denoted as

$$N(S_j) = \sum_{i}^{m} [d_{conn}(c, ES_i) + d_{conn}(ES_{i-1}, ES_i) + d_{conn}(ES_i, cloud)]\, x_{ij}$$

subject to

$$\sum_{i}^{m} S_i^{bw} x_{ij} \leq ES_j^{bw} \qquad \forall j \in ES = ES_1, \ldots, ES_n \tag{4}$$

$$\sum_{j}^{n} x_{ij} = 1 \qquad \forall i \in S = S_1, \ldots, S_m \tag{5}$$

The network optimizing deployment algorithm minimizes network cost presented in Equation 3.3.2. The constraint in Equation 5 ensures that the selected ES is able to support the bandwidth by the service.

The algorithm can be further modified to minimize only the network cost between client and the Edge Server. i.e.

$$N(S_j^{edge}) = \sum_{i}^{m} (d_{conn}(c, ES_i))\, x_{ij} \tag{6}$$

3.4 Tier-based Optimization

The processing and network optimizing algorithms iterate over the entire search space (read Edge Platform) to find the ES satisfying the requirements. As discussed in Section 1, an EP can be composed of hundreds of ESs. The algorithms discussed above imposes significantly large compute time to find the optimal solution.

In Algorithm 1 we present the pseudo-code of *Mute*. It exploits the multi-tier edge to find a server with optimal network cost to edge and near-optimal processing cost.

The algorithm exploits the network cost trend of the tiers, i.e., the Edge servers in lower tiers have a lower network cost to clients than higher tiers. The ESP associates EST_{tier}^{bw} and EST_{tier}^{proc} with each tier, which denotes the maximum supported bandwidth and maximum processing capability of the tier respectively. The algorithm approximates the location of the ideal ES by utilizing tier parameters and prioritizes lower tiers for placement. It further iteratively searches for ES only in the tier whose processing and bandwidth best satisfy the requirements imposed by the service.

Algorithm 1 Mute

Inputs:
  Total number of tiers
  $\texttt{tiers} \in \{\texttt{tier}_1, \ldots, \texttt{tier}_n\}$ Available Edge servers in tiers $\texttt{EST} \in \{\texttt{EST}_{tier1} \cup \ldots \cup \texttt{EST}_{tiern}\}$
  $\texttt{EST}^{bw} \in \{\texttt{EST}_{tier1}^{bw}, \ldots, \texttt{EST}_{tiern}^{bw}\}$
  $EST^{proc} \in \{\texttt{EST}_{tier1}^{proc}, \ldots, \texttt{EST}_{tiern}^{proc}\}$
Initialize:
  $\texttt{selectedServer} \leftarrow \texttt{NONE}$
//This function returns Edge Server ES for deploying Service S
for each $\texttt{t} \in \texttt{tiers}$ **do**
  if $\texttt{EST}_t^{bw} \geq S^{bw}$ and $\texttt{EST}_t^{proc} \geq S^{proc}$ **then**
    lowestNetworkCost $\leftarrow \infty$
    for each ES $\in$ EST$_t$ **do**
      if ES$^{proc} \geq S^{bw}$ and EST$_t^{proc} \geq S^{proc}$ **then**
        Optimize network cost as Equation 6
        if networkCost $\leq$ lowestNetworkCost **then**
          lowestNetworkCost $\leftarrow$ networkCost
          $\texttt{selectedServer} \leftarrow$ ES
        end if
      end if
    end for
    if $\texttt{selectedServer} \neq \texttt{NONE}$ **then**
      break
    end if
  end if
end for

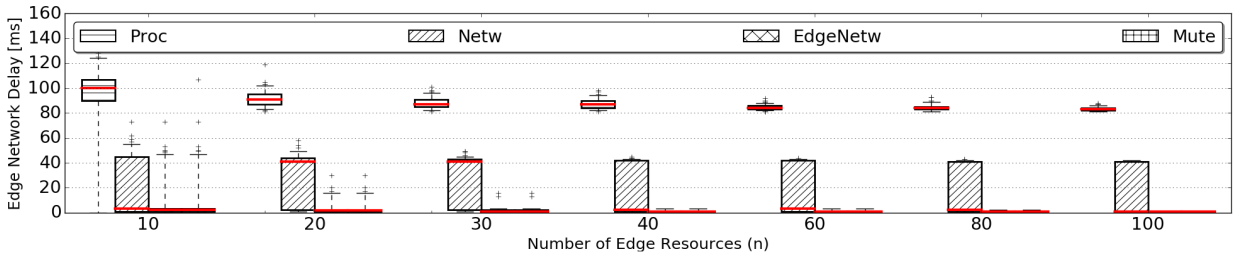
4. Evaluation

In this section, we analyze the performance achieved by *Mute* and compare it with the *Netw*, *Proc* and *EdgeNetw*. *Netw* and *Proc* are state-of-the-art placement algorithms, applied on edge networks whereas *EdgeNetw* is an iterative-variant of *Mute*.

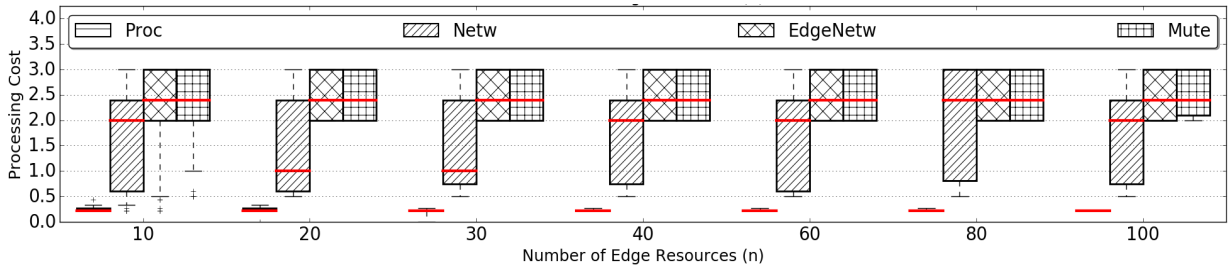
I) Network Optimizing Server Selection (*Netw*) iteratively searches for ES with least network cost of deployment (as modeled in Equation 3.3.2) in an Edge Platform.

II) Processing Optimizing Server Section (*Proc*) selects ES with least processing cost of service deployment (as modeled in Equation 1).

III) Edge-Network Optimizing Server Selection (*EdgeNetw*), similarly to *Mute*, selects the server with least network cost to the client (as modeled in Equation 6). However, unlike *Mute*, *EdgeNetw* is not aware of the multi-tier structure of the edge network and iteratively searches for the optimal server in the search space.



(a) Network Cost Comparison



(b) Processing Cost Comparison

Figure 4: Service Deployment Cost Comparison between *Proc*, *Netw*, *EdgeNetw* and *Mute* algorithms

4.1 Experiment Setup

We implemented *Mute* and the selection algorithms discussed above in a custom Python-based simulator. The simulator considers Edge network graphs based on Rocket Fuel topologies [11] which also provides per-link delays between graph nodes. Overall, we generated 61 network graphs with $\approx 25 - 115$ nodes. Edge networks are generated assigning Edge Servers (e.g., from 10 to 100) on the generated network graphs, which we assume as underlying network topology. The network cost between any two ESs is defined as the sum of the link's delay on the shortest path between such nodes in the underlying network. We provide processing and bandwidth capabilities to each ES in the edge network and cluster them in tiers, as discussed in Section 3.1. For each network graph, we perform 100 placements, resulting in 100 different edge networks. Therefore, we generate ≈ 6000 edge networks throughout the experiments. For the sake of simplicity, we only consider placing a single service on the network in current evaluation and leave the analysis of multiple services as future work.

4.2 Results

Figure 4a and 4b present the box plot results of network and processing cost respectively and capture data distribution throughout all our experiments. The top and the bottom of the boxes represent the first and third quartile respectively, and the red waist represents the median.

Proc, as expected, performs the best in terms of processing cost but has the worst performance for network cost. It does not represent a practical solution as it always selects an ES on tiers that show the least processing cost. ESs within such tiers have significantly higher network delay from the edge and thus show the worst networking cost.

Netw performs significantly better than *Proc* and selects an ES with much lower networking cost. However, it performs worse than *EdgeNetw* and *Mute*. *Netw* selects an ES with the least end-to-end delay which encompasses the delay from the client to the ES and delay from the ES to the cloud. The algorithm does not optimize edge placement as it is unable to make a distinction between servers with similar end-to-end network cost but significantly different path delays between client to the ES. Considering the median of the experiment results, *EdgeNetw* and *Mute* (which show similar results) achieve 66% reduction in network cost on average when compared to *Netw*. However, as discussed in Section 3.1, ESs in lower tiers are not processing capable which reflects in the processing cost achieved by both the algorithms. Both *EdgeNetw* and *Mute* show an increase of 20% in associated processing cost, on average, when compared to *Netw*.

Additionally, we analyze the average time required by each algorithm to complete the service placement, the results of which are shown in Figure 5. As evident from the figure, *Mute* completes its placement in $\approx 50\%$ lesser time (37.5% average reduction) when compared to the other algorithms. *Mute*, unlike the other algorithms, searches over a significantly reduced problem space as it utilizes EST^{bw} and EST^{proc} to estimate the tier which hosts the optimal ES. Therefore, the proposed *Mute* algorithm can discover the ES which can support the requirements imposed by Service Provider while ensuring least network delay to the client. Furthermore, it efficiently utilizes the multi-tier ar-

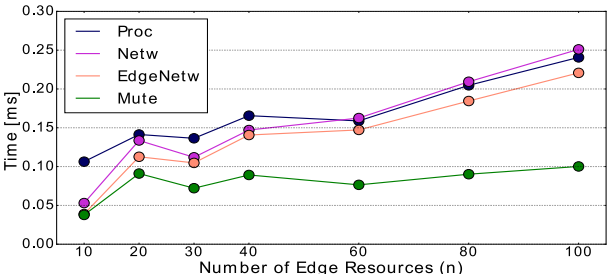


Figure 5: Time Complexity Graph

chitecture of an Edge Platform to achieve a deployment time significantly lower than the state-of-the-art.

5. Related Work & Use-Cases

An increasing number of application use-cases such as Internet-of-Things (IoT), industrial automation, Augmented Reality (AR), smart cities, autonomous transportation systems, etc., are warranting the need for edge compute clouds [16–18]. Several other research areas, such as Industry 4.0 utilize edge clouds to provide efficient solutions to several open questions. For example, researchers have proposed automated collaborative robots which require time-critical processing with extremely low latency (in order of milliseconds) to create a safety zone for their operators [3, 19]. Augmented Reality glasses can assist operators in a continuously varying production environment by performing markerless object recognition and accurate tracking in a factory. However, such use-cases can only be fulfilled if the required data is cached and computed at closely located servers.

State-of-the-art solutions in the fields of virtualization, Software Defined Networking (SDN) and Network Function Virtualization (NFV) represents key technologies to deploy virtualized services at the very edge of the network in a flexible way and on cheap commodity hardware [2]. In this paper, we envisioned the case in which clients have control over which services will be included in their network path using state-of-the-art Service Function Chaining (SFC) techniques such as [13, 14]. In an open market of choosing services, the client can discover specific edge servers which are hosting the required service by utilizing Domain Name System (DNS) based techniques [20, 21]. The clients can significantly enhance their connectivity with the end-server by using services with very low network delay.

6. Conclusion

In this paper we proposed *Mute*, a multi-tier edge cloud architecture which enables edge cloud providers to efficiently deploy services at the edge. *Mute* categorizes edge servers into groups based on their network delay from the client. Due to its unique architecture abstraction, *Mute* can efficiently deploy service function chains on edge servers across multiple edge platforms. Through our extensive simulation-based evaluation on RocketFuel topologies, we show that *Mute* achieves a significant reduction in edge network delay and completion time when compared to state-of-the-art.

In our future work, we plan to investigate the impact of different edge resources clustering strategies on the service placement.

Acknowledgement

This research work has been partly funded by the joint EU FP7 Marie Curie Actions CleanSky Project (G.A.: 607584).

References

[1] Decentralized edge clouds. ieee internet computing 2013.

[2] ETSI MEC-IEG004. Mobile-Edge Computing (MEC): Service Scenarios.

[3] Mohan et al. Managing Data in Computational Edge Clouds. In SIGCOMM Workshop, MECOMM'17 .

[4] Hong et al. Mobile Fog: A Programming Model for Large-scale Applications on the Internet of Things. In *SIGCOMM Workshop on MCC*.

[5] Bonomi et al. Fog Computing and its Role in the Internet of Things. SIGCOMM Workshop Mobile Cloud Computing'12.

[6] Satyanarayanan et al. The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing*, 2009.

[7] Hu et al. Mobile edge computing – A key technology towards 5G. *ETSI white paper*, 2015.

[8] Mohan et al. Edge-Fog cloud: A distributed cloud for Internet of Things computations. CIoT 2016.

[9] Xu et al. Multi-objective virtual machine placement in virtualized data center environments. In *IEEE GreenCom*, 2010.

[10] Prasad et al. Raera: A robust auctioning approach for edge resource allocation. sigcomm workshop, mecomm'17.

[11] Mahajan et al. Inferring link weights using end-to-end measurements. In *ACM SIGCOMM IMW 2002*.

[12] Bifulco et al. Ready-to-deploy service function chaining for mobile networks. In *IEEE NetSoft*. IEEE, 2016.

[13] Naylor et al. Multi-Context TLS (mcTLS): Enabling Secure In-Network Functionality in TLS. *ACM SIGCOMM*, 2015.

[14] Zave et al. Dynamic service chaining with Dysco. SIGCOMM'17.

[15] Bari et al. Data center network virtualization: A survey. *IEEE Communications Surveys Tutorials*, 2013.

[16] Zanella et al. Internet of things for smart cities. *IoT journal*.

[17] CISCO. Fog Computing and IoT (Whitepaper). 2015.

[18] Hong et al. Mobile fog: a programming model for large-scale applications on the internet of things. In *MCC workshop, ACM SIGCOMM'13*.

[19] Robert Bosch GmbH Bosch APAS description. `https://www.bosch-apas.com/produkte-und-services/apas-assistant-mobile/`.

[20] Silvestro et al. MISE: MIddleboxes SElection for Multi-domain SFCs. CAN Workshop, ACM CoNEXT'17.

[21] Silvestro et al. Is today's DNS the right solution for middleboxes selection?. In CrossCloud 17 workshop, ACM EuroSys.