Anveshak: Placing Edge Servers In The Wild

Nitinder Mohan
University of Helsinki, Finland
nitinder.mohan@helsinki.fi

Aleksandr Zavodovski
University of Helsinki, Finland
aleksandr.zavodovski@helsinki.fi

Pengyuan Zhou
University of Helsinki, Finland
pengyuan.zhou@helsinki.fi

Jussi Kangasharju
University of Helsinki, Finland
jussi.kangasharju@helsinki.fi

ABSTRACT

Edge computing provides an attractive platform for bringing data and processing closer to users in networked environments. Several edge proposals aim to place the edge servers at a couple hop distance from the client to ensure lowest possible compute and network delay. An attractive edge server placement is to co-locate it with existing (cellular) base stations to avoid additional infrastructure establishment costs. However, determining the exact locations for edge servers is an important question that must be resolved for optimal placement. In this paper, we present *Anveshak*[1] , a framework that solves the problem of placing edge servers in a geographical topology and provides the optimal solution for edge providers. Our proposed solution considers both end-user application requirements as well as deployment and operating costs incurred by edge platform providers. The placement optimization metric of *Anveshak* considers the request pattern of users and existing user-established edge servers. In our evaluation based on real datasets, we show that *Anveshak* achieves 67% increase in user satisfaction while maintaining high server utilization.

CCS CONCEPTS

• **Networks** → **Network architectures**; *Wireless access points, base stations and infrastructure*; *Topology analysis and generation*; • **Applied computing** → *Service-oriented architectures*;

KEYWORDS

Edge clouds, Fog clouds, server placement, deployment framework

ACM Reference Format:
Nitinder Mohan, Aleksandr Zavodovski, Pengyuan Zhou, and Jussi Kangasharju. 2018. Anveshak: Placing Edge Servers In The Wild. In *MECOMM '18: Workshop on Mobile Edge Communications, August 20, 2018, Budapest, Hungary*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3229556.3229560

1 INTRODUCTION

Novel applications, such as the Internet of Things (IoT) and augmented and virtual reality, have exponentially increased the amount of data generated and transported over the network. To mitigate the response time and handle large-scale data analysis closer to the users and data generators, the researchers have proposed *edge clouds*. As the name suggests, edge cloud is a consolidation of compute servers deployed very close to end user with limited compute, storage and network capability [1, 12, 22]. The central objective of edge clouds is to ensure low network delays for latency-critical applications such as autonomous driving, drones, augmented reality, etc. [10]. Such a requirement can be fulfilled by exploiting the *physical proximity* between the edge server and the client.

Existing studies focus on optimal utilization of the edge server by end-user requests, assuming that the server has been placed already [2, 18]. Little to no attention has been paid to model the edge server deployment problem along with its placement constraints. There are similarities between the edge server placement problem and replica server deployment problem in CDNs, for which several solutions exist in the literature [14, 15, 17]. Akin to CDN cache servers placement problem, edge server placement must also ensure consistent connectivity to end users while minimizing the cost of such a deployment. However, we argue that despite similarities in their objectives, the two placement problems are essentially quite different. Unlike replica servers, an edge server is more likely to cater to several compute requests of local relevance which does not require high volume data transfer over the network. In such a case, the availability and network latency associated with an edge server have greater priority over link usage and network bandwidth.

Several options for deploying edge servers have been proposed in the literature. Mobile Edge Clouds (MEC's), defined by European Telecommunications Standards Institute (ETSI), aim to co-locate edge servers with cellular base stations set up by telecom providers operating in the area [1]. On the other hand, researchers have also proposed to utilize non-conventional compute resources, such as WiFi access points, smart speakers, network switches, etc., to support computation capability at the network edge [9]. Unlike MEC, these resources are owned and managed by end-users. Even though the proposed models differ in deployment requirements, management, capacities, etc.; we envision that the models are independent of the protocols, software stacks and user applications that will drive the edge cloud platform as a whole.

In this paper, we present *Anveshak*, a deployment framework which enables edge service providers to select optimal sites for edge server placement. Our contributions are as follows. (1) *Anveshak*

[1] *Anveshak* means finder in Hindi

MECOMM '18, August 20, 2018, Budapest, Hungary
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5906-1/18/08…$15.00
https://doi.org/10.1145/3229556.3229560

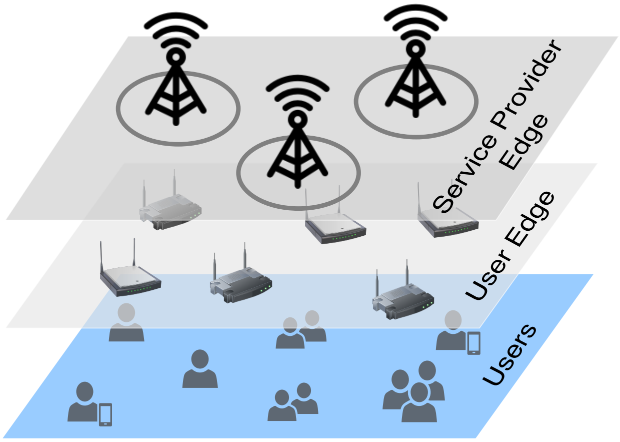


Figure 1: Abstraction of possible edge servers and users in a geographical area

considers the availability and density of unmanaged edge resources in the area to steer the deployment location of a managed server. The novelty lies in predicting future deployments of user-owned edge servers and incorporating it in current edge server deployment problem. (2) We identify the areas of higher preference for deployment by observing the mobility pattern of the users in the area. We consider previous requests issued by the users to prioritize locations with a higher probability of edge service requests, thereby optimizing user satisfaction. (3) We extensively simulate *Anveshak* on real-world datasets collected over the city of Milan, Italy. Our evaluation shows that *Anveshak* increases the user request satisfaction by 67% while maintaining an average server utilization of 83%. To the best of our knowledge, there exist no previously known studies which consider server provisioning in a scenario where multiple edge cloud models coexist and operate in the same physical space.

The rest of the paper is organized as follows. Section 2 discusses the physical edge cloud abstraction composing of multiple edge cloud models in same space. Section 3 provides an in-depth description of model, framework design and optimization problem of *Anveshak*. We implement *Anveshak* and evaluate its performance on real datasets in Section 4. Section 5 reviews the related work. We conclude our paper in Section 6.

2 PHYSICAL EDGE CLOUD NETWORK

Researchers have proposed several edge cloud architectures to support the use-cases present in real world [2]. Mobile Edge Computing (MEC) is a telecommunication-vendor centric edge cloud model wherein the deployment, operation, and maintenance of edge servers is handled by an ISP operating in the area [10]. The model has garnered interest from standardization bodies [6]. On the other hand, researchers have proposed a user-centric view where a user can deploy computationally-capable network devices local to their surroundings. The proliferation of smart speakers, home automation hubs, intelligent wireless access points provides evidence to the adoption of such edge architectures [16]. Unlike the MEC resources, the user-centric edge resources are self-managing in nature and are less likely to have consistent network and computational availability.

Both above models consider different deployment options from in-network placement at aggregation level to opportunistic consolidation composed of small compute hubs. However, we consider a holistic view of a physical space where several edge servers belonging to different cloud models and technologies coexist. As each model brings in its advantages and drawbacks, the coexistence and cooperation between available edge servers will be critical to efficient computation and context availability in future. Figure 1 shows the physical abstraction of edge servers and users coexisting in a geographical area. The model is a two-tier hierarchy of edge servers in a physical space alongside with users, the details of which are explained below.

Users: The subscribers of edge cloud in a region act as the source for all compute requests. Previous research in user mobility has shown that user request distribution in any area is temporally and behaviorally influenced [11]. For example, user request density is more populated in city centers than suburban areas. Such request patterns profoundly affect the utilization of edge server deployments in any region. An efficient server deployment algorithm must consider the origin and pattern of user requests in a geographical region to allocate server resources for optimal utilization and availability.

User-managed Edge: This layer is composed of edge servers which are managed by individual entities for local usage and are likely to be deployed in households, small workplaces, etc. These servers utilize WiFi (short-range) networks to interact with end-user. The user-managed edge servers are responsible for handling computational requests from a small set of clients and are thus limited in computation power. However, they provide a very local context to user-generated request. The availability of such servers is highly dependent on user residency and mobility itself. For example, densely-populated residential areas and tourist attraction spots have a higher availability of WiFi access points than in industrial/office areas [19].

Service Provider-managed Edge: The top-most layer of the edge server abstraction model is composed of service-provider managed edge servers. Such servers are co-located with cellular base stations set up in the region due to their strategic locations and constant ISP management. An edge server physically co-located at the base station significantly reduces the operation and maintenance costs involved for specifically setting up a location to house a server. Unlike user-managed edge, the edge servers managed by a third-party service provider have a higher computational capability and wider-area coverage. These edge servers utilize the network fabric and capability offered by the cellular base station to connect with users and amongst themselves. ISPs can also remotely manage and maintain these edge servers by utilizing their existing set up infrastructure.

3 *ANVESHAK*: MODEL AND DESIGN

The problem of deploying edge servers in a physical space boils down to ensuring low latency, proximity and high availability to

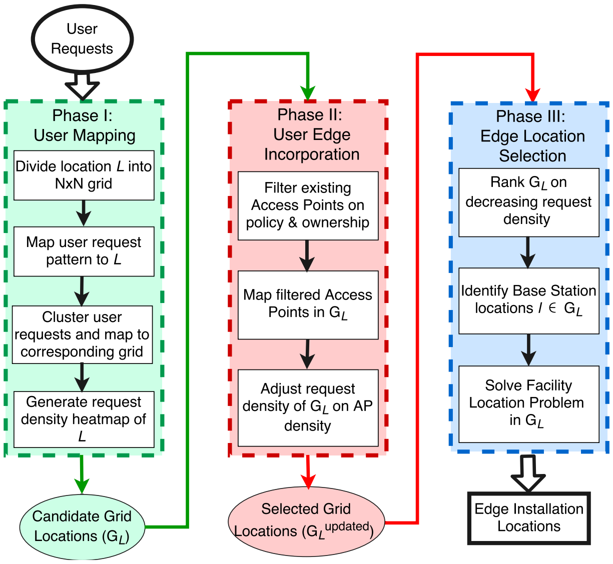
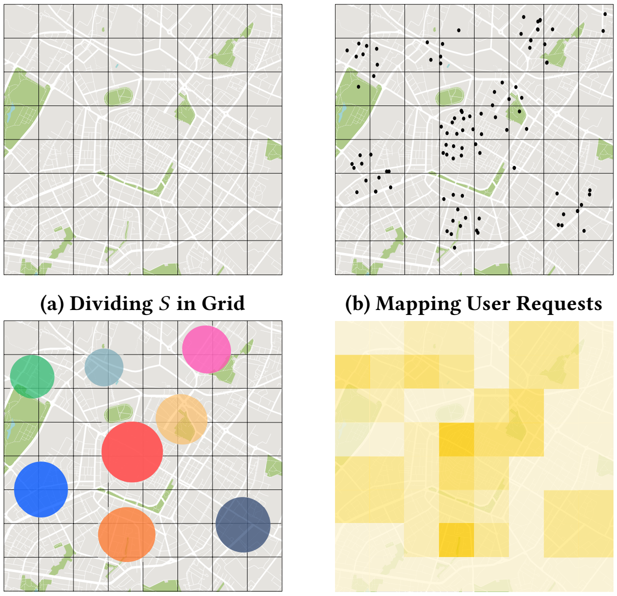

User Requests
Phase I: User Mapping
Divide location *L* into NxN grid
Map user request pattern to *L*
Cluster user requests and map to corresponding grid
Generate request density heatmap of *L*
Candidate Grid Locations (G_L)
Phase II: User Edge Incorporation
Filter existing Access Points on policy & ownership
Map filtered Access Points in G_L
Adjust request density of G_L on AP density
Selected Grid Locations ($G_L^{updated}$)
Phase III: Edge Location Selection
Rank G_L on decreasing request density
Identify Base Station locations $l \in G_L$
Solve Facility Location Problem in G_L
Edge Installation Locations

Figure 2: The overall workflow of *Anveshak*.


(a) Dividing S in Grid

(b) Mapping User Requests

(c) DBScan Request Clusters

(d) User Request Heatmap

Figure 3: *Anveshak* Phase 1.

clients. Further, installing a server at any location incurs a combination of *CAPEX* (purchase and deployment) and *OPEX* (maintenance, security) costs to the service provider. To maximize the profits, it is in best interests of the provider to select edge sites *intelligently* such that the deployed server has maximum impact and utilization. *Anveshak* enables service providers to find optimal edge sites in a large metropolitan area. It selects a prioritized list of cellular base stations within an area which can be augmented with an edge server. Through its insightful utilization of pre-existing user request patterns and edge servers in the area, *Anveshak* ensures that the selected edge site has maximum reachability and high user request satisfaction.

Figure 2 shows the workflow of *Anveshak*. We categorize the framework's functioning in three phases, *user mapping*, *user edge incorporation* and *edge location selection*.

Phase 1: User Mapping to Physical Space

The design of *Anveshak* is based on the assumption that the edge service provider works in conjunction with the ISP to ensure optimal installation of edge servers on ISP-managed base stations. Therefore, the service provider will have access to the user request database from ISPs operating in the region. These request databases can include Call Detail Records (CDR), message requests, internet usages etc. which can be augmented to form user request pattern. *Anveshak* utilizes the dataset of communication requests by the clients of the ISP in its first phase. The objective of the framework in this phase is to identify areas of high communication requests in the geographical region as these areas have a higher probability of receiving edge compute requests.

Anveshak begins the phase by dividing the space S into evenly spaced square grids[2] (shown in Figure 3a). Further, *Anveshak* maps the user communication request originating from a location in S, as shown in Figure 3b. The user requests are normalized and averaged over a time duration of one to several months such that temporal outliers in the dataset (user gatherings, fairs, concerts etc.) are ironed out. Once the framework has all user requests mapped to point set P in S, it clusters them based on inter-request distances and density[3]. The clustering algorithm identifies regions with dense and frequent user requests in S by selecting a minimum number of request points within $MinPts$ radii of an existing base station in that area. Further, the algorithm also specifies ϵ which defines the minimun required distance between two points to classify them as part of a single cluster. Figure 3c presents the user requests clustered together in S. The choice of ϵ and $MinPts$ is key to efficient clustering in *Anveshak* and can be easily adjusted by the service provider to best suit deployment requirements.

Following request clustering, *Anveshak* maps arbitrary cluster shapes to the corresponding grids in S (shown in Figure 3d). The density of a cluster is normalized to generate grid-based heatmap of the region. In doing so, *Anveshak* can handle overlapping clusters, small dense clusters, and clusters of various shapes more efficiently than related approaches. Furthermore, this enables the framework to overcome the inefficiencies of the clustering algorithm used.

The density heatmap and its location coordinates is fed into the next phase of *Anveshak*.

Phase 2: User Edge Incorporation

As discussed in Section 2, compute-capable network devices such as smart speakers, home automation, smart WiFi routers, etc. have become quite popular and are expected to develop into *$4.2 billion*

[2] The grid size and total number can be tweaked to allow at least one-to-one mapping between grid and base station.

[3] Even though numerous clustering algorithms have been proposed and used in related works [3], we adopt *Density-Based Clustering Essentials (DBScan)* [5] in *Anveshak*.

market by 2022 [16]. Such smart devices can resolve relatively small computations locally to the clients and will be preferred by users over a service provider-managed server in the same location. The availability of these devices significantly impacts the utilization of deployed edge server in an area as the number of user requests which will be offloaded to a managed edge server will notably reduce. In its *second phase*, *Anveshak* integrates the current and future availability of user-managed edge servers by the end users. It does so by building on the assumption that areas with high density of WiFi access points (APs) are more likely to have a future deployment of user-managed edge servers. The inclusion of this phase is key to the novelty of *Anveshak* over related works.

Anveshak merges all user requests from grid $G_i \in S$ such that user requests in the same cluster C are distributed over several grid groups. It further exploits already existing datasets of WiFi APs in space S (such as Wigle [19]) and maps them on the same grid G_i. Based on the density of existing deployment, *Anveshak* revises the user request heatmap of S where grids with denser WiFi availability receives negative request density adjustment. The resulting map prioritizes locations with a lower number of local edge deployments as the probability of clients to request a provider-managed edge server is higher. The grid locations G_L is fed into the next phase of *Anveshak*.

Phase 3: Edge Location Selection

In its final phase, *Anveshak* increasingly orders grids from Phase 2 on ratio of user request density. The set of users U within a grid G_i can be served by x possible edge locations (existing base stations) denoted by $L^{G_i} = l_1, \ldots, l_x$[4]. *Anveshak* ensures one-hop connectivity between users and deployed server by selecting a location l_k which is best reachable for majority of users in G_i. Let $R^{max}_{(u,S_l)}$ be the maximum tolerated network distance between U and S_l where $l \in L^{G_i}$.

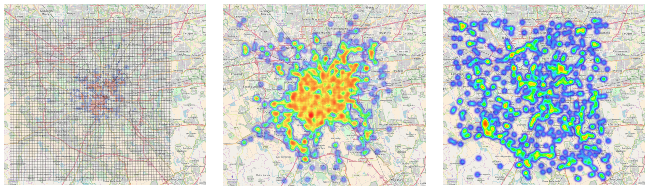
$$R^{max}_{(u_l,S_l)} = max[u - S_l] \quad \forall u \in U \tag{1}$$

Based on the requirements and number of servers to be placed in S, R^{max} of S_l will specify the cluster boundary for satisfying u and is influenced by the connectivity range of the existing base station. Further, let α denote the maximum cost incurred to access the server S_l by users in the cluster. Thus, the network cost ($n_{(S,u)}$) of a cluster can denoted as

$$n_{(S,u)} = \alpha * R_{(S,u)} \tag{2}$$

In order to estimate the network costs between users and server location within a grid, the model utilizes a coordinate based network latency approximation technique [13]. *Anveshak* attempts to minimize the latency to one-hop between majority of users and deployed server S_l within grid G_i based on Equation 1. Further, the users in same grid which do not fall under direct connectivity of S_l are reachable within 2-3 hops by utilizing the internal network between base stations.

Let x_l denote a binary decision variable which is 1 if we locate S_l in candidate location $l \in L^{G_i}$. Therefore, the optimal server location for an arbitrary user u can be defined as,



(a) User requests in 100x100 grid (b) User-managed WiFi access points (c) Base station density

Figure 4: Distribution of normalized user communication requests, WiFi access points and Telecom Italia's cellular base stations over Milan, Italy

$$S_u = min \sum_{l \in L^{G_i}} \{S_l | S_l \in S, n_{(S_l,u)} < n_{max}\} x_l \tag{3}$$

The equation 3 is a variant of facility location problem (FLP) [7] with network capacity constraints. The resulting optimization is a well-known NP-hard problem, the approximate solution of which can only be obtained by adding specific placement constraints. However, since *Anveshak* divides S in small grids with limited number of edge site locations, even the worst-case iterative solution for optimizing Equation 3 takes reasonable time.

4 EVALUATION METHODOLOGY

We now evaluate the efficiency of *Anveshak* in placing edge servers over Milan, Italy by utilizing several open datasets. We first implement *Anveshak's* workflow (shown in Figure 2) as two separate pluggable modules. The Phase 1 of the framework is implemented as *clustering module* in R. The module produces clusters of user requests based on request patterns, WiFi access points, and base stations datasets provided to it. We design the module to be independent of the choice of clustering algorithm used and can be freely selected by the service provider (default is DBScan). Phase 2 and 3 of *Anveshak* are implemented in Python and return base station coordinates to the service provider considering the constraints imposed.

We compare *Anveshak* with two alternative placement approaches which have been discussed in the literature [14, 17]. The approaches are described as follows:

(1) *Greedy:* This method allocates average user request densities to the base stations in the area of interest. It then utilizes a greedy selection algorithm to select *top-k* base stations which serve most number of users in the area.
(2) *Random:* As its name suggests, this approach randomly chooses k base stations on the map and assigns edge servers to them.

Unlike *Anveshak*, both of the approaches mentioned above neither consider whether selected base stations serve the same set of users due to connectivity overlap nor the availability of other edge servers in the area.

4.1 Dataset

In order to gauge the impact of selection algorithm on real networks, we utilize several open datasets over city of Milan, Italy. For user

[4] In case there exists only single base station within a grid, it becomes the only candidate for server deployment for that grid

connectivity requests, we use the dataset published by Telecom Italia from November 1st to December 31st 2013. The anonymized dataset divides the map of Milan into 100x100 grids of 250m width. The dataset contains user's internet connection to base station as user request tied to its grid ID along with the time when it was made. In our evaluation, *Anveshak* utilizes the average total user requests in November, 2013 to generate clusters of user requests. The heatmap of unclustered user internet requests for November is shown in Figure 4a .

We map all WiFi access points in the same area of that of Telecom Italia dataset by utilizing open crowd-sourced dataset from Wigle [19]. The dataset contains SSID, location coordinates, signal strength, channel number etc. for all access points. Out of the entire dataset, we filter out the hotspot access points to reduce variations in access point location density. Figure 4b shows the density heatmap of WiFi access points in Milan. We utilize an open dataset of all cellular base stations in the world and use the ones in Milan using the coordinates provided in the dataset. We further filter and use close to 800+ Telecom Italia base stations in Milan in our evaluation. The heatmap of Telecom Italia base stations in Milan is shown in Figure 4c.

As assumed in design of *Anveshak*, we can observe from Figure 4 that both user requests and WiFi access points are concentrated in populated areas of the city (such as city center) whereas the cellular base stations are evenly distributed throughout the map.

4.2 Results

We now evaluate the placement efficiency of the discussed approaches. We task the placement algorithms to select 50 out of total 812 base stations in Milan as edge server deployment sites. The average coverage radius of base station in the dataset is little higher than 1000m; we utilize a coordinate based latency approximation [13] to estimate user requests which can be satisfied within this area. These requests may originate from neighboring grids of the selected base station[5]. Further, *Anveshak* utilizes users internet traffic requests for November 2013 for initial clustering and edge site selection. We then evaluate the efficiency of the selection for user requests in December 2013.

We focus our evaluation and comparison on two metrics: (1) the percentage of user requests satisfied by selected edge site, and (2) the total utilization of the deployed edge server. All our results are averaged over ten runs.

User Request Satisfaction: Figure 5 compares the percentage of user requests which were satisfied by the base stations selected by each approach for every third day in December. As observed from the figure, edge servers deployed via *Anveshak* can serve ≈ 67% more users than *Greedy* in an area. We attribute this behavior to greedy selection of sites based on user requests inherent to *Greedy* functioning. Even though the site selection by *Greedy* prefers highest serving base stations, it often fails to consider locations which are far away from densely populated areas yet having significant request origination. On careful analysis, we found that unlike *Anveshak* which satisfies all clusters on the map, *Greedy* favors base stations within most dense user cluster.

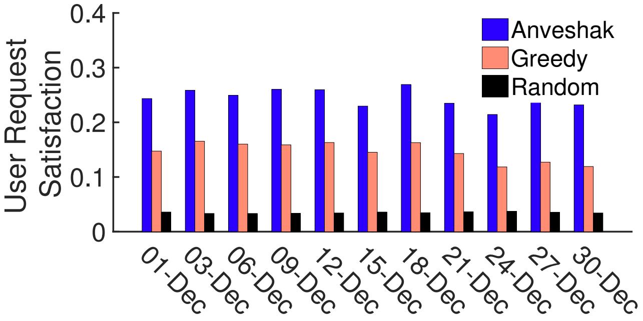


Figure 5: Normalized User Request Satisfaction

.

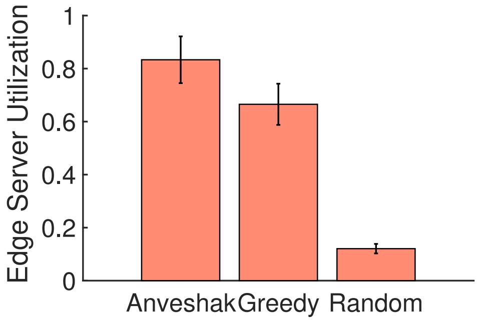


Figure 6: Average Server Utilization

.

From the results, we also see that *Anveshak* satisfies ≈ 25% of total user requests on average by selecting 8% of total base stations. In our further experiments, we found that *Anveshak* achieves more than 90% user satisfaction by installing just 124 edge servers (on average). Whereas *Greedy* and *Random* require 218 and 300+ servers respectively. We do not show the detailed results due to space limitations.

Server Utilization: We deploy edge servers on all selected locations where a server can handle up to 500 user requests every 10 minutes. Further, we augment 10% WiFi APs in the coverage area as compute-capable and a single AP handle 50 requests/10mins within the grid thereby operating at 10% compute power of that of a managed edge server. As discussed in Section 2 user-managed edge server first handles all user requests upon exceeding which it is sent to base station edge server. If the base stations receive more requests than its capacity in 10 minutes, it offloads additional requests to the remote cloud. Figure 6 shows overall server utilization in December 2013.

Anveshak achieves 83% server utilization on average whereas *Greedy* and *Random* achieve 66% and 12% utilization only. We attribute the reason for such high utilization by *Anveshak* to its selection of edge servers with less availability of user-managed edge

[5] As mentioned in Section 3, *Anveshak* utilizes *DBScan* for clustering user requests as it is one of the most commonly used clustering algorithm used in the literature. We currently leave the effects of different clustering algorithms on performance of *Anveshak* in our future work.

servers. The sites selected by *Greedy* have a high concentration of WiFi APs which leads to lesser requests sent to the managed server.

5 RELATED WORK

In the past few years, several works have focused on offloading user requests to a network embedded server [8]. Several studies have focused on selection algorithms for identifying the strategic location on in-network servers in the context of Content Delivery Networks (CDN) [17]. The CDN cache placement problem is modeled in two problem subclasses in literature, (i) *cache placement* and (ii) *server placement*. Both placement problems optimize the objective function designed to choose K replicas from N possible sites within the network. The cache placement problem concentrates on minimizing user latency whereas the server placement minimizes network traffic load [14].

Considering the similarity between edge and CDN servers, researchers have proposed edge cloud architectures closely mimicking that of CDNs [2]. Instead of specific CDN nodes set up in the network, the edge servers are deployed within the ISP network to ensure that they lie on client's path to the cloud. However, the specific node placement within the ISP network is left up to the discretion of the service provider. In [4], the authors place edge server on aggregator network nodes and focus on VM placement and migration between deployed edge servers. Yang et al. [21] consider a MEC architecture where the server nodes have already been placed within the ISP network and only need to be activated per-user request capacity and the required delay.

In other edge cloud models, researchers consider placing servers in strategic locations within a geographical area. Tentacle [23] attempts to find optimal edge sites for cache placement which may or may not be co-located with an existing base station. The authors optimized selection problem by adding constraints on user request latency from selected edge server site. However, due to the design of their objective function, the authors prioritize edge sites which do not have an associated base station which leads to a high first-time deployment cost for edge service provider. Furthermore, the work only accounts for network delays between users and base station but do not consider the high-speed network connections between neighboring base stations. In [20], authors design a placement algorithm for capacitated cloudlets in a flat wide-area metropolitan network (WMAN). In their abstraction model, an edge server can only colocate with a WiFi access point whose network delay to other access points is the least. The author's model required per-link delay knowledge for entire edge network. Lastly, Bouet et al. [3] divides a geographical area into distinct user clusters and assign an edge server to each cluster based on request density. However, the estimation of exact edge site within the cluster where the server must be placed (whether co-located with a base station or as independent) is left un-tackled by the authors.

6 CONCLUSION

Edge clouds have emerged as a leading research interest which aims to deploy compute servers couple of network hops away from the end user. Several edge cloud models have been proposed by researchers which focus on edge server placement in the network based on application use-case being tackled. In this paper, we propose *Anveshak*, a deployment framework designed to assist edge service providers in efficiently identifying base stations in a geographical region for edge server deployment. *Anveshak* considers the user request patterns and user-managed edge servers in the area to holistically select which base stations must have a colocated edge server. We evaluate *Anveshak* by utilizing real user request and cellular datasets over city of Milan, Italy released by Telekom Italia. Our results show that edge servers deployed by *Anveshak* achieves upto 67% increase in user request satisfaction while keeping average edge server utilization at 83%.

ACKNOWLEDGMENT

The work for this paper was jointly funded by the joint EU FP7 Marie Curie Actions the Cleansky project (grant Agreement No. 607584) and the Academy of Finland in the Adaptive and Intelligent Data (AIDA) project (grant no. 317086).

REFERENCES

[1] A. Ahmed and E. Ahmed. 2016. A survey on mobile edge computing. In *Conference on Intelligent Systems and Control*. IEEE, 1–8.

[2] F. Bonomi et al. 2012. Fog Computing and Its Role in the Internet of Things. In *Proceedings of MCC*.

[3] Mathieu Bouet and Vania Conan. 2017. Geo-partitioning of MEC Resources. In *Proceedings of MECOMM*. ACM.

[4] Alberto Ceselli, Marco Premoli, and Stefano Secci. 2017. Mobile Edge Cloud Network Design Optimization. *IEEE/ACM Trans. Netw.* (2017).

[5] M. Ester et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *Kdd*.

[6] M. Patel et al. [n. d.]. Mobile-edge computing introductory technical white paper. ETSI, Sophia Antipolis, France. ([n. d.]).

[7] K. Jain et al. 2002. A new greedy approach for facility location problems. In *Proceedings of ACM STOC*. ACM, 731–740.

[8] R. Kemp et al. 2010. Cuckoo: a computation offloading framework for smartphones. In *International Conference on Mobile Computing, Applications, and Services*. Springer, 59–79.

[9] P. G. Lopez et al. 2015. Edge-centric Computing : Vision and Challenges. *ACM Computer Communications Review* (2015).

[10] R. Mahmud et al. 2018. Fog computing: A taxonomy, survey and future directions. In *Internet of Everything*. Springer, 103–130.

[11] Rod Meikle and Joseph Camp. 2012. A Global Measurement Study of Context-based Propagation and User Mobility. In *HotPlanet*.

[12] N. Mohan and J. Kangasharju. 2016. Edge-Fog Cloud: A Distributed Cloud for Internet of Things Computations. In *2nd Int. Conf. on Cloudification of the Internet of Things (CIoT'16)*. IEEE.

[13] T. S. E. Ng and Hui Zhang. 2002. Predicting Internet network distance with coordinates-based approaches. In *IEEE Infocom*.

[14] L. Qiu, V. N. Padmanabhan, and G. M. Voelker. 2001. On the placement of Web server replicas. In *Proceedings IEEE INFOCOM* .

[15] P. Radoslavov, R. Govindan, and D. Estrin. 2002. Topology-informed Internet replica placement. *Computer Communications* (2002).

[16] Arizton Market report. [2017]. Smart Voice Assistant Speaker Market - Global Outlook and Forecast 2017-2022. https://www.arizton.com/market-reports/smart-voice-assistant-speaker-market. ([n. d.]).

[17] J. Sahoo, M. A. Salahuddin, R. Glitho, H. Elbiaze, and W. Ajib. 2017. A Survey on Replica Server Placement Algorithms for Content Delivery Networks. *IEEE Communications Surveys Tutorials* (2017).

[18] P. Silva, et al. 2016. Efficient Heuristics for Placing Large-Scale Distributed Applications on Multiple Clouds. In *16th IEEE/ACM CCGrid*.

[19] wigle. [n. d.]. Wireless basestation dataset. https://wigle.net/. ([n. d.]).

[20] Z. Xu et al. 2016. Efficient Algorithms for Capacitated Cloudlet Placements. *IEEE Transactions on Parallel and Distributed Systems* (2016).

[21] B. Yang et al. 2018. Cost-Efficient NFV-Enabled Mobile Edge-Cloud for Low Latency Mobile Applications. *IEEE Transactions on Network and Service Management* (2018).

[22] B. Yang, W. K. Chai, Z. Xu, K. V. Katsaros, and G. Pavlou. 2018. Cost-Efficient NFV-Enabled Mobile Edge-Cloud for Low Latency Mobile Applications. *IEEE Transactions on Network and Service Management* (2018).

[23] H. Yin et al. 2017. Edge Provisioning with Flexible Server Placement. *IEEE Trans. Parallel Distrib. Syst.* (2017).