On the Impact of Clustering for IoT Analytics and Message Broker Placement across Cloud and Edge

Daniel Happ
happ@tkn.tu-berlin.de
Technische Universität Berlin, Germany

Suzan Bayhan
s.bayhan@utwente.nl
University of Twente, The Netherlands

ABSTRACT

With edge computing emerging as a promising solution to cope with the challenges of Internet of Things (IoT) systems, there is an increasing need to automate the deployment of large-scale applications along with the publish/subscribe brokers they communicate over. Such a placement must adjust to the resource requirements of both applications and brokers in the heterogeneous environment of edge, fog, and cloud. In contrast to prior work focusing only on the placement of applications, this paper addresses the problem of jointly placing IoT applications and the pub/sub brokers on a set of network nodes, considering an application provider who aims at minimizing total end-to-end delays of all its subscribers. More specifically, we devise two heuristics for joint deployment of brokers and applications and analyze their performance in comparison to the current cloud-based IoT solutions wherein both the IoT applications and the brokers are located solely in the cloud. As an application provider should consider not only the location of the application users but also how they are distributed across different network components, we use von Mises distributions to model the degree of clustering of the users of an IoT application. Our simulations show that superior performance of our heuristics in comparison to cloud-based IoT operation is most pronounced under a high degree of clustering. When users of an IoT application are in close network proximity of the IoT sensors, cloud-based IoT unnecessarily introduces latency to move the data from the edge to the cloud and vice versa while processing could be performed at the edge or the fog layers.

1 INTRODUCTION

Internet-of-Things (IoT) systems have evolved from single-sensor systems serving a single user to complex systems collecting a massive volume of data from multiple sensors. The value of this network enfolds when deriving a deeper understanding of our surroundings by processing the collected multimodal data. Use cases benefiting IoT data analytics are broad, e.g, forest fire detection [13], smart city video analytics [1], smart manufacturing [17], or step counting in the health domain [15]. With this change and increasing complexity of IoT systems, there is a need to reconsider the current practice of cloud-based IoT deployments in which sensors communicate exclusively with a cloud backend and all further processing is offloaded to the cloud [17].

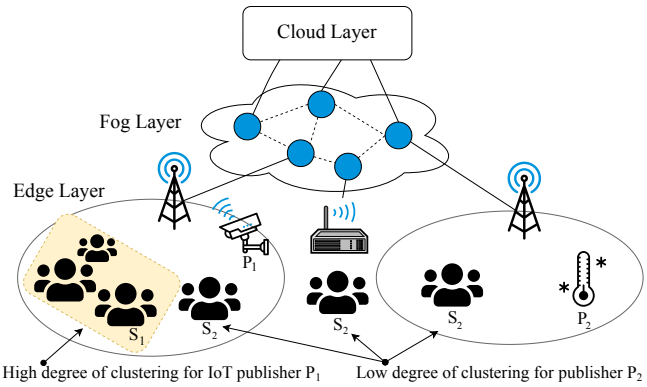


Figure 1: IoT publishers might have their subscribers in their close proximity with a high degree of clustering as for Publisher 1 or the subscribers might be uniformly distributed as for Publisher 2.

Today, sensor data is often sent to a cloud-based service tier, which distributes the data to interested end-users and value-added services. However, cloud-based IoT solutions suffer from several shortcomings. First, many applications such as Augmented Reality (AR) are sensitive to the latency introduced by relaying data and processing over the cloud [12, 21]. Additionally, the upstream bandwidth required for sending raw data might prohibit this approach where no broadband connection is available and alternative technologies, e.g. cellular data links, have to be used. Other shortcomings of the cloud-based approach include unknown or ambiguous privacy policies [17]. Last but not the least, it is undesirable from the perspective of the network operators to transfer the high volume of IoT data to the cloud while they are already facing the challenge to serve an increasing amount of network traffic, referred to mostly as *data tsunami*.

To mitigate the above-listed shortcomings, the emerging fog-centered architecture [2, 19] offers IoT data processing at least partially before reaching the cloud, e.g., within the core network of an ISP. Conceptually, this approach presents an opportunity for (additional) data peering closer to the data generators, e.g., IoT sensors, reducing latency and network traffic, increasing reliability and privacy. However, due to three key reasons, it is not straightforward how an application provider should exploit this fog architecture to achieve its performance goals. First, an IoT application might be composed of smaller processing tasks with certain dependency

EdgeSys '20, April 27, 2020, Heraklion, Greece

ACM ISBN 978-1-4503-7132-2/20/04…$15.00
https://doi.org/10.1145/3378679.3394538

and possibly with different resource requirements. Hence, the deployment must account for this dependency in addition to the rate and amount of data transferred from one task to the other. Second, there might be many users of this IoT application spread over different parts of the network. Finally, network nodes might exhibit differences in terms of their resource availability, e.g., CPU or bandwidth, and monetary cost. In addition to these three aspects, communication approach between the IoT sensors and the users of the sensor data has to be considered.

With larger-scale IoT systems, the efficiency of the communication model between the IoT sensors and their subscribers also becomes crucial. The publish/subscribe interaction pattern has emerged as the de-facto standard for IoT messaging due to several benefits enabled via decoupling the IoT publishers from their subscribers by the help of *brokers* [5, 9]. To realize benefits of this interaction mode, an IoT application provider has to decide on the number and placement of the core analytic operators as well as the pub/sub brokers to exchange messages over. Placing brokers, however, without considering the IoT analytics operators might lead to suboptimal operation. For instance, fog and edge layers might be constrained in their resources which can support a limited number of users/services whereas the cloud tier is resourceful. Also, there might be conflicting goals for the owner and the users of an IoT platform that are not easily reconciled.

In contrast to prior work such as [2, 10], we analyze the joint placement of operators and message brokers across cloud, fog, and edge. Different than our recent work [8], we focus on the distribution of the IoT application users and use von Mises distributions to model the degree of clustering which is a metric to reflect how the users of an IoT application are distributed in the near or far network proximity of the IoT sensor. Our analysis via simulations shows that the current cloud-based IoT application results in significantly longer application latency in comparison to our heuristics under a high degree of clustering of the IoT application users.

2 SYSTEM MODEL

In this work, we consider the placement of IoT analytics applications and pub/sub brokers on a given set of connected nodes from the perspective of an application provider. Therefore, assume an application owner offers multiple IoT applications where each application consists of a set of interdependent operators. An IoT application takes as input the data published by an IoT sensor measuring real world phenomena and processes this data to serve the application users. Because of the benefits offered by the decoupling between producers, operators and consumers, we assume a publish/subscribe-based IoT system with matchmaking brokers.

We consider a network of a fixed number of cloud-based virtual machines at the *cloud layer*, fog virtual machines operated by Internet service provider (ISP) at the *fog layer*, and on-premise sensor gateways at the *edge layer* as in Fig.1. We assume that the application owner can deploy operators and brokers on those nodes in an on-demand pay-as-you-go fashion. To decide on the best placement, the application owner needs to know the following properties of each node k: (1) the available maximum capacity $R_{k,q}$ of a specific resource $q \in Q$ (e.g., CPU or memory); (2) the available uplink ($\beta_k^{\uparrow}$) and downlink ($\beta_k^{\downarrow}$) capacity of the node to the access network; (3)

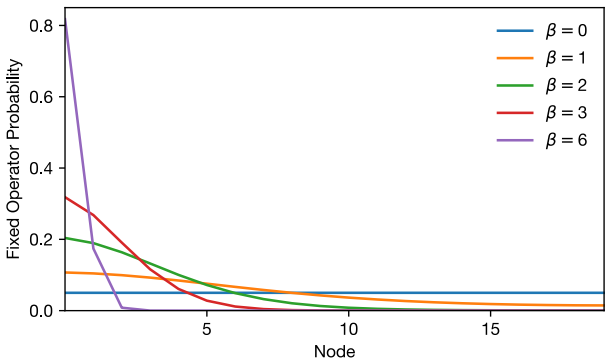


Figure 2: Probability distribution considering von Mises based clustering model for a fixed operator being located on the node with the specific number.

a cost factor f_k indicating a cost ratio with regard to a reference machine representing the cost of deploying an IoT *operator* or a broker at this node.

We define an *operator* as the smallest deployable software component [6] corresponding to an arbitrary computation mapping a set of input streams to a set of output streams. Using this abstraction, we can represent publishing sensor devices, processing tasks, subscribing applications or services of all applications under the same application owner as an *operator* o_i. Moreover, we define an application as a directed acyclic graph of operators.

To represent the relationship between the operators of an application, we define a binary dependency matrix $D_{i,j}$ as follows: $D_{i,j} = 1$ if o_i depends on o_j and zero otherwise. As IoT sensors have only output data stream but no input streams, they do not depend on other operators, i.e., $D_{i,*} = 0$ for such o_i. Similarly, IoT subscribers have no other operators depending on them and no output, i.e., $D_{*,i} = 0$ for such o_i. Each operator o_i is characterized by an associated cost c_i for calculating one output message, an average output rate r_i (per seconds), and the resource demand $U_{i,q}$ for several resources $q \in Q$ such as CPU and memory. They could also include further requirements such as a certain type of hardware that is required for the particular operator, e.g., a dedicated graphic processor for machine learning or video encoding in hardware. Finally, we assume that each operator publishes to exactly one broker, from where the data is disseminated to one or more subscribers.

2.1 Clustering of fixed operators

When a service provider plans its deployment, it has to consider the geospatial distribution of its users. Consider the two IoT sensors denoted by P_1 and P_2 in Fig.1 as an example. All subscribers S_1 of publisher P_1 are in the same network area. We refer to such a scenario as a highly clustered case in terms of proximity of the IoT publisher and its subscribers. On the other hand, P_2 has subscribers S_2 distributed over three network areas. We refer to such cases as a scenario with low clustering.

More formally, to better quantify the clustering of fixed operators (publishers and subscribers), we propose a modified von Mises

distribution [14] as follows. Our clustering distribution is characterized by the parameter $\beta \geq 0$. The parameter κ of the von Mises distribution is given as follows: $\kappa = 2^{\beta} - 1$. As von Mises distributions are usually defined on the interval $[-\pi, \pi]$, we only consider twice positive side and scale the x-axis to the interval $[0, n]$, where n is the number of nodes that an IoT sensor or an IoT user might be located at. Then, we can state the probability density function as follows:

$$f(x, \kappa, n) = \frac{e^{\kappa \cos(\frac{x\pi}{n})}}{\pi I_0(\kappa)} \tag{1}$$

where $x \in \mathbb{N}_0$, $0 \leq x < n$, and $I_0(\kappa)$ is the modified Bessel function of order 0. Our modified von Mises distribution has several properties that make it very suitable as a metric for the clustering of fixed operators on nodes. First, if β and therefore κ is zero, the distribution becomes a uniform distribution. So, we choose places for fixed operators randomly across all nodes. Second, with increasing κ, the distribution becomes concentrated around node zero and the distribution approaches the positive half of a normal distribution around zero and variance $\frac{1}{\kappa}$.

Since von Mises distributions are continuous and we need to choose a discrete node for each fixed operator, we sample on the continuous interval of $[0, n]$ and round the obtained value down to the next integer. Fig. 2 shows the probability that a fixed operator is placed on a certain node in a network with 20 nodes under different β values. Using our distribution model, we generate scenarios with a certain clustering factor and analyze the impact of clustering via simulations in Sec.4.

3 IOT ANALYTICS PLACEMENT

Problem Statement: Given a network of nodes, the application owner has to decide on which nodes to deploy its application operators and brokers. Since nodes have finite capacity, e.g., computation power or bandwidth, the application owner has to ensure that the placement is feasible by considering the capacity of each network node and link, e.g., $R_{k,q}$, $\beta_k^{\uparrow}$, and $\beta_k^{\downarrow}$ of node n_k. Using the location of its customers, the application owner can formulate an optimization problem to minimize the total cost of serving the IoT application users in terms of total end-to-end delays of all subscribers. Formally, we state the objective as follows:

$$min \sum_{o_i \in O} (1 - \theta_i) d_i, \tag{2}$$

where binary auxiliary variable θ_i is 1 if the operator has an output while d_i represents the completion time of operator o_i and depends on the following three components. First, all other operators that this operator o_i depends on must finish their task and publish the output data that this operator will use as its input. Second, the published data will experience transmission delay on the paths to this operator. Since the operator has to receive all the input before it can start its execution, the slowest operator o_j's delay (sum of its completion and published output transmission time) determines this second component. Finally, once all the published data is received, the operator o_i has a specific execution time on its host. Let us first introduce several variables needed to calculate d_i: $x_{i,k}$, $z_{i,k,i',k'}$, $C_{k,k'}$, r_i, f_k and c_i.

The binary decision variable $x_{i,k}$ yields value 1 if o_i is placed on node n_k while $z_{i,k,i',k'}$ yields 1 if o_i is on n_k and $o_{i'}$ publishes to a broker on $n_{k'}$. $C_{k,k'}$ denotes the cost of sending one byte from n_k to $n_{k'}$ and r_i is the output rate of o_i. Finally, f_k represents the cost factor of n_k in comparison to some reference machine. Using the aforementioned three delay components, we define d_i recursively as follows:

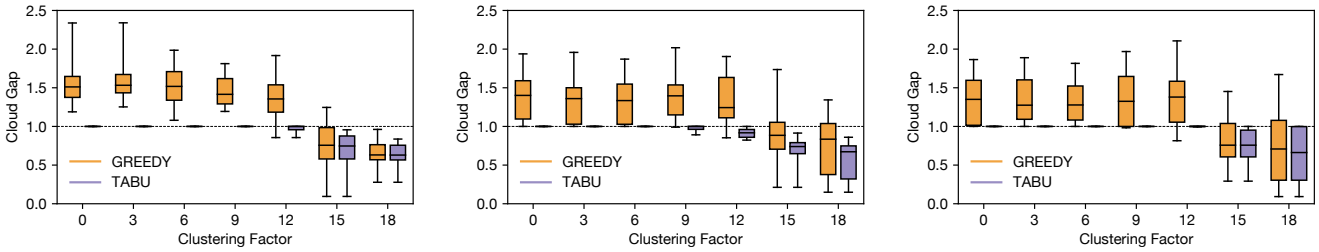
$$d_i = \max_{D_{i,j}=1} \left(d_j + \sum_{n_k \in N} \sum_{n_{k'} \in N} z_{j,k,j,k'} C_{k,k'} + z_{i,k,j,k'} C_{k',k} \right) + \sum_{n_k \in N} r_i x_{i,k} f_k c_i. \tag{3}$$

As solving this joint problem optimally does not scale for the size of realistic topologies, we design two heuristics, namely GREEDY and TABU, which we introduce next.

Greedy heuristic (GREEDY): First, GREEDY places fixed operators (i.e., publishers and subscribers) and fixed brokers (if any) to their corresponding nodes. Next, GREEDY assigns a stratum number to each operator where the stratum represents the number of hops from a data source. GREEDY sorts operators according to their stratum number and places operators with the lowest stratum number first on the closest feasible node to their subscriber(s). In this process, GREEDY ignores operators that are not yet placed as well as the position of brokers. It searches the solution space in a depth-first fashion: if an operator cannot be placed on any node given the current assignment of already-placed operators, the most-recently placed operator is moved to the second best node. This step is repeated recursively until either a valid solution is found or every possible combination is tried, which means that there is no feasible assignment.

Once an operator is placed, GREEDY looks for a location for this operator's broker. GREEDY places the broker optimally considering the already-placed operators and ignoring those that are yet to be placed. After each operator is assigned an output broker, if there is a constraint on the maximum number of brokers, GREEDY checks if the placement violates this constraint. If it does, GREEDY merges two brokers that are the closest in terms of latency to each other until the constraint on the total number of brokers is satisfied.

Tabu-like search (TABU): Our second proposal is based on tabu search [16]. Briefly, TABU starts with a feasible solution already available and tries possible actions to improve the existing solution iteratively by searching for a better solution in the neighborhood of the current solution. As the initial solution, we take first the solution found by GREEDY and run TABU. Next, we run TABU taking the solution by CLOUD as its initial solution. Then, TABU takes the best of these two solutions. In each iteration, TABU might take one of the following actions in an attempt to improve its existing solution: (1) change operator position, (2) change operator/broker assignment, and (3) change broker position. We choose the actions one after the other. For choice (1) and (2), TABU chooses an operator randomly. For (1), the neighborhood is given by every possible movement of the operator. For (2), the neighborhood is every possible reassignment to an existing broker. Similarly, in case of (3), TABU chooses a broker randomly and considers every possible movement of the broker to find an improved solution. The



(a) Random application graph **(b) Fanout application graph** **(c) Sequence application graph**

Figure 3: Cloud gap with increasing clustering factor (β) for each application type.

search uses a short term memory that stores the last u solutions as a tabu list. The algorithm in general chooses the best solution of the neighborhood as the next candidate, except if the solution was already tried recently, which is a tabu. In that case, the next best solution that was not tried recently is chosen, which makes the algorithm consider sub-optimal solutions to avoid getting trapped in a local optimum. The candidate is chosen as the starting point for further iterations. TABU stops when it has tried a fixed number v consecutive iterations without improvements.

4 PERFORMANCE EVALUATION

We evaluate the performance of the proposed approaches via system-level simulations on our custom-made Python simulator. Different than our previous work [8], we analyze the placement of operators and brokers for a given quantifiable metric for the clustering of publishers and subscribers. Moreover, we want to verify that our heuristics are scalable and efficient also for larger network topologies.

We take the cloud-based IoT operation as our baseline (CLOUD) and report the performance of our proposals (i.e., GREEDY and TABU) in comparison to this baseline. In CLOUD, all brokers and analytics operators are hosted in the cloud. We define the *cloud gap* metric as the ratio of the utility function in (3) obtained by the deployment according to our heuristic(s) over the utility function obtained by CLOUD. Our key goal is to develop insights on the performance of our heuristics under an increasing degree of clustering and under an increasing network size. More importantly, we aim at finding the cases where our heuristics outperform the baseline significantly. In the following, we will first briefly discuss the models used for the underlying topology and the application graphs. Next, we will present the results of our simulations of two scenarios. In the first scenario, we evaluate the impact of the degree of clustering on the cloud gap achieved by our heuristics. The second scenario provides a closer look to the impact of the network size. A more detailed explanation with the exact parameters and the reasoning behind them is provided in [8].

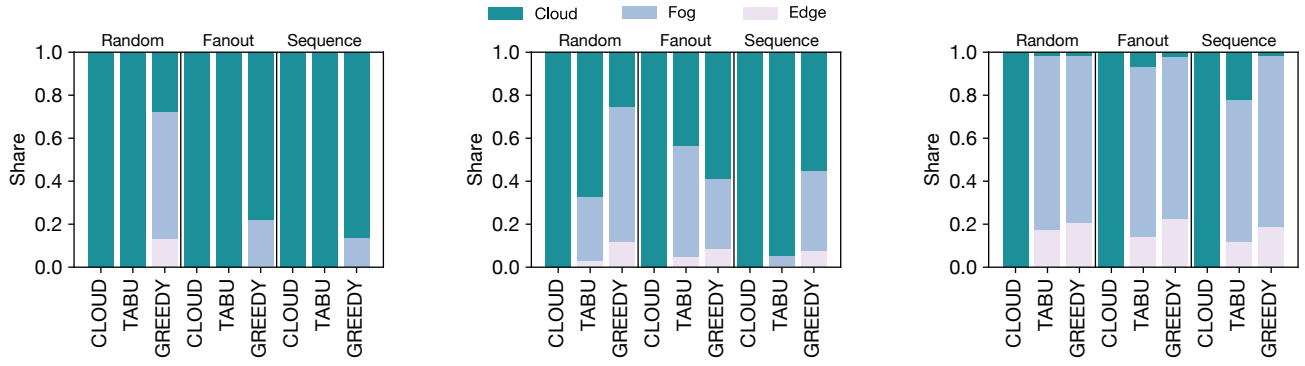
Network topology: We model the latency between different network nodes considering a country or region of a similar size as the continental United States. For the available edge-to-fog bandwidth, we set the minimum and maximum bandwidth to be 1MBit/s and 100MBit/s respectively. Since residential connections are usually asymmetric, we set the uplink bandwidth to one fourth of the downlink bandwidth. For the bandwidth in the fog and between fog and cloud, we use common values for 100BASE-TX, 1000BASE-T and 10GBASE-T Ethernet. Regarding the round-trip-time (RTT), we consider a uniform distribution in [10 ms, 30 ms] for links between edge and fog, [5 ms, 70 ms] for fog-fog links, and [5 ms, 35 ms] for the connection between fog and cloud. To model the CPU computation resources available at each layer, we take the Passmark score[1] as an indicator. Passmark scores for edge and fog are chosen between 420 and 3800 for edge, and between 5500 and 12000 for fog. The amount of memory available is between 1GB and 4GB, and 8GB and 32GB respectively. The cloud has practically infinite resources.

IoT analytics applications: An IoT application is typically modeled as a dependency graph with additional attributes for output rates, CPU and memory requirements, and other constraints such as a fixed network location. While some real-world application graphs are available in the literature, such as [1, 15, 17], we prefer to keep our analysis as generalizing as possible representing a broader spectrum of applications. Hence, we consider the following three topologies as our IoT application graphs: i) *a random topology* with no cycles; ii) *a fanout topology* where a set of processing operators needs the input of every publisher in the system and gives its output to every subscriber; iii) *a sequence topology* where the processing operators form a chain of operators that must run as a sequence. To represent the resource requirements of an application, we set the following parameters: CPU score$\sim U(100, 2000)$, RAM$\sim U(0.05\,\text{GB}, 1\,\text{GB})$, and output rate$\sim U(0.001\,\text{MBit/s}, 1\,\text{MBit/s})$. Publishers and subscribers are fixed across the topology. In the following scenarios, we consider six applications with eight operators. We consider both cases where all applications are of the same type or a mixture of the three types.

4.1 Impact of Clustering

First, we analyze the impact of degree of clustering β on the cloud gap metric for a network with 300 nodes. We use clustering factor $\beta = \{0, 3, 6, 9, 12, 15, 18\}$. When $\beta = 0$, publishers and the subscribers are distributed uniformly while setting $\beta = 18$ results in almost all publishers and subscribers being connected to the same node in the network.

[1] https://www.passmark.com/



(a) Clustering factor $\beta = 0$ **(b) Clustering factor $\beta = 12$** **(c) Clustering factor $\beta = 18$**

Figure 4: Operator location for clustering factor $\beta = \{0, 12, 18\}$ across cloud, fog, and edge.

Fig. 3 depicts the cloud gap with increasing clustering for each application type. In the figures, a horizontal line at $y = 1$ shows the point where CLOUD and other approaches have the same performance, i.e., cloud gap equals 1. Hence, values below this line represents the scenarios where our heuristics outperform CLOUD as they result in lower total delay for the considered IoT application subscribers. From the figures, we have the following observations. First, for every type of application, CLOUD performs at least as good as and usually better than a GREEDY for $\beta < 9$. With $\beta \geqslant 12$, however, GREEDY starts to outperform CLOUD.

While for the sequence application graph, the clustering has to be 15 or higher to benefit from fog or edge deployment, the results indicate a lower threshold of about 12 for random application graphs. For the fanout case, even a clustering factor of 9 can in certain cases benefit from a deployment across edge and fog. The possible improvement is quite widespread at higher clustering factors, ranging from no improvement (cloud gap equals 1) to significant improvement with gaps as low as 0.19, which translates to around 80% lower latency. In a nutshell, under high clustering, CLOUD with its high delay becomes inefficient compared to our proposals. Comparing TABU and GREEDY, unsurprisingly, TABU outperforms GREEDY as it takes the best of the two solutions driven from GREEDY and CLOUD as the initial solution. While GREEDY becomes a promising solution only under high clustering cases, TABU offers improvements over CLOUD under almost all settings. However, the achieved improvement becomes the most visible under high clustering.

We now want to develop a deeper understanding on which layer is preferred for the operators and brokers. Since the placement of the brokers show similar results, we depict only the operator locations. Fig. 4 depicts the distribution of operators across each layer for clustering factors 0, 12, and 18 representing the following cases: no clustering, a medium degree, and a high degree of clustering, respectively.

Since our deployment algorithms have no control over the locations of the fixed operators, i.e. publishers and subscribers, we exclude them from this analysis. For no-clustering case, we observe that TABU resorts to the provided solution of placing all the operators to the cloud and does not find a better solution. For the random application graph, GREEDY has about 60% of the operators placed at the fog and even about 10% at the edge. For the other graphs, GREEDY also prefers the cloud for the majority of operators. With increasing clustering factor, we observe that GREEDY starts to favor nodes closer to the edge and places an increasing amount of operators across the edge and fog for all application topologies. Meanwhile, TABU improves on the placement by shifting operators. However, there is no general trend on which network layer is preferred for operator deployment. For example, in the random case at $\beta = 12$, less operators are placed in the edge and fog, while more are placed in the cloud. On the other hand, for the random case at $\beta = 18$, more operators are placed in the fog. In other words, there is no rule-of-thumb to place the operators on a specific layer. It depends on the dependency graph, network topology, node resources, and also the placement of the brokers.

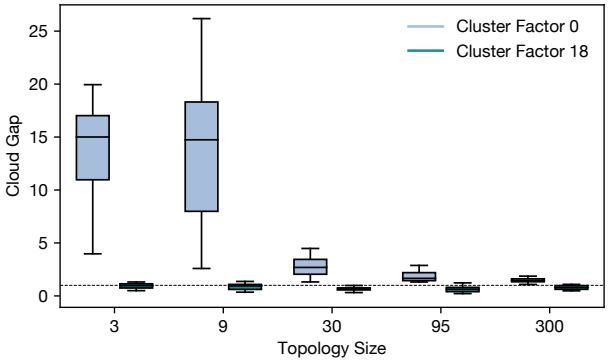


Figure 5: Cloud gap for different topology sizes.

4.2 Impact of Network Size

Next, we study the impact of the network size on the cloud gap of GREEDY to answer the question if our observations on the impact of clustering is valid also for smaller networks. We use a mixture of the three application graphs and vary the network size from 3 to 300. In Fig. 5, we observe that the improvements at a high degree of

clustering seem to be independent of the network size for GREEDY. However, GREEDY does not provide any performance improvements over CLOUD for small networks under uniform distribution of the IoT application users. But, with increasing network size, the performance of GREEDY starts to approach to that of CLOUD. On the other hand, under high clustering, GREEDY outperforms CLOUD in all cases. For example, the cloud gap for a topology with 300 nodes and a clustering factor of 18 varies between 0.48 and 1.09 with the median at 0.80. This validates the benefit of joint placement even using a simple heuristic.

5 RELATED WORK

To the best of our knowledge, ours is the first and only solution providing a joint placement of brokers and IoT applications. While there is significant research on broker-based IoT systems or optimization of broker operation [11], joint placement of brokers with other components is largely overlooked. Hong et al. [10] show the NP-hardness of the operator deployment problem and propose a heuristic for maximizing the number of satisfied IoT analytics. Considering the QoS and resource requirements of different IoT applications, [10] first identifies the scarcest resource and allocates as many IoT operators as possible with the least demand for this scarce resource. Moreover, [10] aims at keeping the distance between dependent operators minimal and limited to a certain number of hops. Brogi et al. [3] take the latency and bandwidth requirements of each application into account and deploy applications with more resource requirements the first on the nodes with the highest resources. FogFlow [4] and Foggy [20] are fog computing-based frameworks enabling IoT application providers to deploy their IoT services over cloud and edges. Different than all these centralized schemes, Guerrero et al. [7] propose a decentralized service-popularity based placement running at each fog device wherein the goal is to favor more popular services and migrate them closer to the clients in terms of hop counts. Finally, Renart et al. [18] propose a solution in which the placement objective is to minimize an aggregate cost consisting of the application latency, the network traffic, and the message overhead. Our paper differs from all these prior work in that it considers the joint placement of brokers and IoT applications and highlights the impact of the clustering of the IoT users on the performance of current cloud-based IoT service operation.

6 CONCLUSION

With the growing number of IoT applications, it becomes paramount to reconsider the cloud-based IoT operation which might not only fall short of meeting the delay requirements of some applications but also may yield a huge volume of network traffic. In this paper, we proposed two heuristics to place an IoT application consisting of multiple operators with certain dependency on the available network nodes at the edge, fog, and the cloud. Different than the prior work, we considered broker and IoT application placement jointly as broker-based IoT communication offers scalability and energy efficiency to resource-constrained IoT devices. Simulation results show that the cloud-based IoT operation starts to become inferior in its performance with increasing clustering of the users of an IoT application: when the IoT application users are in the close proximity of the IoT sensor, it becomes inefficient to deploy the IoT application and the broker in the cloud despite the cloud's high resource availability. Given the impact of clustering factor, we leave design of such a clustering-aware placement to a future work. Another potential research direction is the design of adaptive schemes that might change the application deployment based on certain performance observations and change in the network dynamics, e.g., number and location of users.

REFERENCES

[1] Ganesh Ananthanarayanan, Victor Bahl, Landon Cox, Alex Crown, Shadi Nogbahi, and Yuanchao Shu. 2019. Video Analytics - Killer App for Edge Computing. In *ACM MobiSys (MobiSys)*. 695–696.

[2] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog Computing and Its Role in the Internet of Things. In *MCC Workshop on Mobile Cloud Computing*. 13–16.

[3] Antonio Brogi and Stefano Forti. 2017. QoS-aware deployment of IoT applications through the fog. *IEEE Internet of Things Jrnl.* 4 (2017), 1185–92.

[4] Bin et al. Cheng. 2017. Fogflow: Easy programming of iot services over cloud and edges for smart cities. *IEEE Internet of Things Journal* 5, 2 (2017), 696–707.

[5] Jasenka Dizdarević, Francisco Carpio, Admela Jukan, and Xavi Masip-Bruin. 2019. A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration. *ACM Computing Surveys (CSUR)* 51, 6 (2019), 1–29.

[6] H. J. Hong et al. 2017. Supporting Internet-of-Things Analytics in a Fog Computing Platform. In *IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. 138–145.

[7] Carlos Guerrero, Isaac Lera, and Carlos Juiz. 2019. A lightweight decentralized service placement policy for performance optimization in fog computing. *Journal of Ambient Intelligence and Humanized Computing* 10, 6 (2019), 2435–2452.

[8] Daniel Happ, Suzan Bayhan, and Vlado Handziski. 2020. JOI: Joint Placement of IoT Analytics Operators and Pub/Sub Message Brokers in Fog-centric IoT Platforms. *Future Generation Comp. Sys.* (2020). under review.

[9] Daniel Happ, Niels Karowski, Thomas Menzel, Vlado Handziski, and Adam Wolisz. 2017. Meeting IoT Platform Requirements with Open Pub/Sub Solutions. *Annals of Telecommunications* 72, 1 (2017), 41–52.

[10] Hua-Jun Hong, Pei-Hsuan Tsai, An-Chieh Cheng, Md Yusuf Sarwar Uddin, Nalini Venkatasubramanian, and Cheng-Hsin Hsu. 2017. Supporting internet-of-things analytics in a fog computing platform. In *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 138–145.

[11] Michael A. Jaeger, Helge Parzyjegla, Gero Mühl, and Klaus Herrmann. 2007. Self-Organizing Broker Topologies for Publish/Subscribe Systems. In *ACM Symposium on Applied Computing (SAC '07)*. 543–550.

[12] Qiang Liu and Tao Han. 2018. Dare: Dynamic adaptive mobile augmented reality with edge computing. In *2018 IEEE 26th International Conference on Network Protocols (ICNP)*. IEEE, 1–11.

[13] Liyang Yu, Neng Wang, and Xiaoqiao Meng. 2005. Real-time forest fire detection with wireless sensor networks. In *Proceedings. 2005 International Conference on Wireless Communications, Networking and Mobile Computing, 2005.*, Vol. 2. 1214–1217.

[14] KV Mardia and PJ Zemroch. 1975. Algorithm AS 86: The von Mises distribution function. *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 24, 2 (1975), 268–272.

[15] P. Michalák and P. Watson. 2017. PATH2iot: A Holistic, Distributed Stream Processing System. In *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. 25–32.

[16] R. Mijumbi, J. Serrat, J. L. Gorricho, N. Bouten, F. De Turck, and S. Davy. 2015. Design and evaluation of algorithms for mapping and scheduling of virtual network functions. In *IEEE Conference on Network Softwarization (NetSoft)*.

[17] Pankesh Patel, Muhammad Intizar Ali, and Amit Sheth. 2017. On using the intelligent edge for IoT analytics. *IEEE Intelligent Systems* 32, 5 (2017), 64–69.

[18] Eduard Gibert et al. Renart. 2019. Distributed operator placement for IoT data analytics across edge and cloud resources. In *IEEE/ACM Int. Symposium in Cluster, Cloud, and Grid Computing (CCGrid)*.

[19] Shusen Yang. 2017. IoT stream processing and analytics in the fog. *IEEE Communications Magazine* 55, 8 (2017), 21–27.

[20] Emre Yigitoglu, Mohamed Mohamed, Ling Liu, and Heiko Ludwig. 2017. Foggy: A framework for continuous automated IoT application deployment in fog computing. In *IEEE AIMS*.

[21] Ben Zhang, Nitesh Mor, John Kolb, Douglas S. Chan, Ken Lutz, Eric Allman, John Wawrzynek, Edward Lee, and John Kubiatowicz. 2015. The Cloud is Not Enough: Saving IoT from the Cloud. In *USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*.