ISSN: 2834-7706

Review Article

Journal of Mathematical Techniques and Computational Mathematics

Autonomic IoT Application Placement in Edge/Fog Computing

Paridhika Kaya*

University of Toronto

***Corresponding Author**
Paridhika Kayal, University of Toronto.

Submitted: 2023, Dec 06; **Accepted:** 2024, Jan 03; **Published:** 2024, Jan 15

Citation: Kayal, P. (2024). Autonomic IoT Application Placement in Edge/Fog Computing. *J Math Techniques Comput Math, 3*(1), 01-08.

Abstract
Edge/Fog computing recently emerged as a novel distributed virtualized computing paradigm, where cloud services are extended to the edge of the network, thereby increasing network capacity and reducing latencies. In fog computing, applications are composed of microservices that are mapped to edge computing and communication devices (fog nodes). A crucial component in fog computing is placement algorithms that assign microservices to fog nodes since they determine the overall system performance in terms of energy consumption, communication costs, load balancing, and others. Placement strategies devised for cloud computing are generally centralized and, therefore, not well suited for decentralized fog systems. In this paper, we consider the joint optimization of two conflicting objectives, energy consumption at fog nodes and communication costs of applications, as a game between fog nodes and applications where each agent is modeled to control one objective. We follow a Markov approximation method for the design of a fully distributed autonomic service placement strategy without central coordination or global state information. Evaluation results show that the new approach provides a more optimal solution as compared to previous autonomic placement algorithms.

1. Introduction

Fog computing, also known as edge computing is a distributed computing paradigm that acts as an intermediate layer in between cloud data centers and IoT devices/sensors. It offers to compute, networking, and storage facilities so that cloud-based services can be extended closer to the IoT devices/sensors. IoT applications are designed as a collection of microservices installed in application containers [1]. Fig. 1 depicts an IoT application that is composed of five microservices. Directed edges indicate the direction of communication, and edge weights show the amount of data flow between microservices, referred to as *chaining degree*. Fig. 2a shows a network of four fog nodes. Edges between fog nodes indicate communication links with edge weights, representing the distance between the nodes. These fog nodes are resourceconstrained and cooperate with forwarding data.

In this paper, we are concerned with the strategies for assigning microservices to fog nodes with the goal of balancing energy consumption at fog nodes and network traffic costs. Placement of microservices can be done in two ways, which presents a tradeoff between two placement objectives. The first strategy is placing maximal communicating microservices on each fog node, as illustrated in Fig. 2a. This keeps the communication costs between the microservices low, but at the same time, it leads to high utilization at some fog nodes. Also, this strategy may not be feasible due to limited resources at fog nodes. On the other hand, the second strategy is to distribute

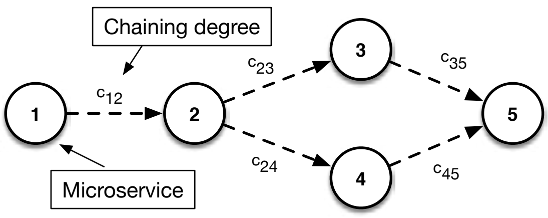
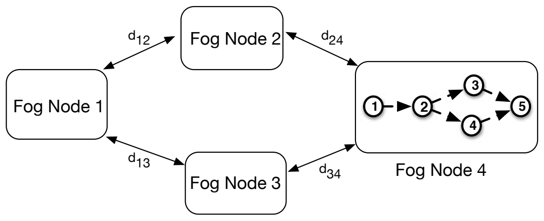
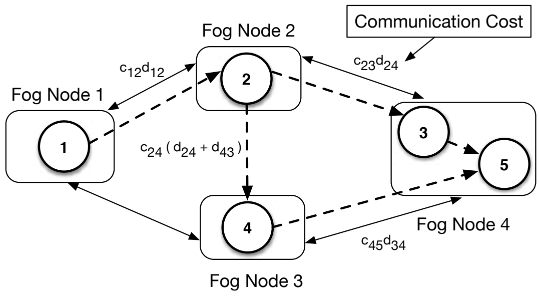


Figure 1: Microservices of a fog application.

Communicating microservices over a network of fog nodes, as shown in Fig. 2b, which leads to data exchange between the fog nodes (*communication cost*). This strategy results in a load-balanced system, but at the same time, increases communication costs.



(a) Application on a single fog node.



(b) Application distributed over fog network.

Figure 2: Placement of IoT Application.

While service placement in terms of Virtual Machine placement in cloud datacenters and Network Function Virtualization (vNF) placement has been extensively studied, the placement of microservices in fog computing has received far less attention [2-5]. Due to more stringent resource constraints, a higher degree of heterogeneity, the geographical distribution of fog nodes, and, above all, a decentralized system architecture, placement strategies devised for cloud platforms are not ideal for fog computing. This motivates the search for distributed solutions that do not require any central coordination or global information exchange.

In this paper, we present a distributed placement strategy that seeks to optimize energy consumption and communication costs. The joint optimization subject to resource constraints presents a combinatorial optimization problem with two conflicting objectives. We model the optimization problem as a game between two agents: fog nodes and applications, where each agent is modeled to control one objective. We make the problem tractable by applying a Markov approximation [6]. We exploit the iterative structure of a Markov approximation by letting individual fog nodes and applications make decisions on the next step of the iteration. Then, by limiting the state changes between iterations to moving or swapping a single microservice between fog nodes, we arrive at a fully distributed solution to the placement problem. Our main contribution is an autonomic placement method that does not require any centralized coordination or control and does not require any global information exchange. An evaluation of our algorithm shows the convergence properties and makes comparisons with our previous autonomic placement methods.

The remainder of the paper is organized as follows. In Sec. II, we discuss prior work on distributed service placement in cloud and fog computing. In Sec. III, we present our system model and formulate microservice placement as an optimization problem. In Sec.IV, we present the Markov approximation of the optimization problem. In Sec. V, we present our distributed service placement algorithm. We evaluate the algorithm in Sec. VI and we conclude in Sec. VII.

2. Related Work

A centralized approach to resource allocation in the fog can take advantage of a large set of methods that were developed for the cloud but does not exploit the opportunities for self-management of the distributed resources. Distributed solutions for fog networks have emerged only recently [2-5]. For an exhaustive literature survey, we refer the readers to [7,8].

In this paper, we work with the Markov approximation an iterative approximation method for combinatorial optimization problems, which lends itself to a distributed realization [6]. Markov approximations have been previously applied for both VM and vNF placements, however, without leveraging the potential of achieving a fully distributed solution and involving a central control [9,10]. In recent work a decentralized placement algorithm is proposed for microservice-based IoT application placement in a fog environment in which every fog node contains the application placement logic and contributes in making placement decisions instead of having a centralized entity [11]. However, the algorithm only considers the resource requirements of microservices for making placement decisions without considering energy-efficient placement decisions and cost optimizations. In we propose a fully distributed service placement strategy that jointly optimizes the energy consumption at fog nodes and the communication cost of the applications without the requirement of any central controller or global information exchange [7]. The evaluation of the algorithm shows that the proposed algorithm is comparable to the existing heuristics. However, in spite of having a joint optimization objective, the algorithm only considers the fog nodes as the autonomous decision making entities that try to optimize both energy consumption and communication cost. Moreover, in we found that a lower cost is achieved when both fog nodes and applications are considered as two agents and get involved in making placement decisions [8]. Thereupon, this work extends the proposed idea of designing a Markov approximation based autonomic placement method where both fog nodes and applications act as autonomous decision-making entities and get turns to perform the state transitions [12].

3. System Model and Problem Formulation

We consider a system of N fog nodes and M microservices from with FN_n denoting the nth fog node and MS_m is the mth microservice [7]. Let AP_r denote the rth application and M_r be the set of all microservices belonging to the application r. Lastly, K denotes an index set of resources. Matrix $A = (A_{kn})$ consists of the resource availability at fog nodes and Matrix $R = (R_{km})$ stores the resource requirements of microservices. Dependent on the type of resource, the A_{kn} and R_{km} are measured in million instructions per second (MIPS), bytes, or bits per second (bps). Let P denote the allocation matrix where each element P_{mn} is a binary variable with $P_{mn} = 1$ if MS_m is placed on FN_n and 0 otherwise. We use the column vector $P_n = (P1n,...,P|_M|_n)^T$ to denote the assignment of microservices on FN_n. Table I summarizes the notation.

A_{kn}:	Availability of resource type k at fog node FN_n.
R_{km}:	Amount of resource type k required by microservice MS_m.
C_{ij}:	Chaining cost between microservices MS_i and MS_j.
D_{lk}:	Distance between any fog nodes FN_l and FN_k.
P_{mn}:	$P_{mn} = 1$ if MS_m is placed on FN_n, $P_{mn} = 0$ otherwise.

Table 1: Summary of Notation.

An allocation matrix of a feasible placement satisfies

$$\begin{aligned} &RP \preceq A\,, \\ &\sum_{n\in N} P_{mn} = 1\,, \quad \forall m \in M\,, \end{aligned} \tag{1}$$

where $\preceq$ indicates that the inequality holds componentwise. The utilization of the kth resource at FN_n, denoted by ρkn is determined by the ratio of allocated to available resources, given by

$$\rho_{kn} = \frac{1}{A_{kn}} R_k P_n\,.$$

The utilization of the CPU at a fog node largely determines its energy consumption and has been shown to be a useful linear approximation of energy consumption across a wide range of applications and platforms [13]. Assuming that $c \in K$ is the resource index for the CPU, the energy consumption for a given placement matrix P at all fog nodes is given by

$$E(P) = \sum_{n\in N} \rho_{cn}\,. \tag{2}$$

The fog network is represented by an undirected graph. Let D denote the symmetric matrix representing the distance between each pair of fog nodes, where the cost D_{lk} accounts for the distance and/or the available network capacity between FN_l and FN_k. The distance between two fog nodes FN_l and FN_k is set to

$$D_{lk} = \frac{\text{hop count between } \text{FN}_l \text{ and } \text{FN}_k}{\max_{i,j}\{\text{ hop count between } \text{FN}_i \text{ and } \text{FN}_j}\}\,, \tag{3}$$

so that the distance metric is proportional to the number of traversed fog nodes on the routing path. The chaining degree between the microservices are tabulated in matrix C, such that C_{ij} denotes the chaining degree from MS_i to MS_j. The total communication cost between two microservices MS_i and MS_j that are mapped to fog nodes FN_k and FN_l, respectively, is $D_{kl}C_{ij}$, the product of the chaining cost of the microservices and the distance between the fog nodes where the microservices are located. Given a placement matrix P expressing the mapping of microservices to fog nodes, the total communication cost by all microservices, denoted by $G(P)$, is given by

$$G(P) = \sum_{m_1,m_2\in M} C_{m_1 m_2} \sum_{n_1,n_2\in N} P_{m_1 n_1} D_{n_1 n_2} P_{n_2 m_2}\,. \tag{4}$$

With the constraints of a feasible allocation in Eq. (1), and expressions for energy consumption (Eq. (2)) and communication cost (Eq. (4)), we can formulate a multi-objective optimization problem, called *Fog Service Placement* (*FSP*), as

$$
\begin{aligned}
\text{FSP}: \quad & \min_{P} \quad T(P) = \sigma E(P) + (1-\sigma) G(P) \\
& \text{s.t.} \quad RP \preceq A \\
& \qquad \sum_{n \in N} P_{mn} = 1\,, \quad \forall m \in M\,, \\
& \qquad P_{mn} \in \{0,1\}\,, \quad \forall m \in M\,, \forall n \in N\,.
\end{aligned} \tag{5}
$$

The parameter σ with $0 \leq \sigma \leq 1$ calibrates the weights of energy consumption and communication cost in the optimization. A larger value of σ prioritizes energy consumption over the communication cost, while a smaller σ stresses the importance of low communication costs. Due to the integral constraints, the FSP problem is NP hard. We will therefore resort to approximation methods.

4. Markov Approximation of FSP

In this section, we present approximation algorithm for FSP given in Eq. (5). Our point of departure is the Markov approximation method for iteratively solving combinatorial problems [6]. The method constructs a Markov chain such that the steady-state of the Markov chain provides an approximate solution to the optimization problem. Let $\mathcal{P}$ denote the set of all placement matrices defined as

$$
\mathcal{P} = \left\{ P \in \{0,1\}^{|M| \times |N|} \mid \sum_{n \in N} P_{mn} = 1,\ \forall m \in M \right\}.
$$

To estimate the cardinality of $\mathcal{P}$, consider the worst case where all fog nodes have sufficient capacity to accommodate all microservices. Then, the number of all possible placements is given by $|N|^{|M|}$. With [6, Theorem 1] we can give an approximate formulation of FSP by

$$
\begin{aligned}
& \min_{P_1 \in \mathcal{P}} \sum_{P_1 \in \mathcal{P}} \left(p_{P_1} E(P_1) + \frac{1}{\beta}\, p_{P_1} \log p_{P_1} \right) \\
& + \min_{P_2 \in \mathcal{P}} \sum_{P_2 \in \mathcal{P}} \left(p_{P_2} G(P_2) + \frac{1}{\beta}\, p_{P_2} \log p_{P_2} \right) \\
& \text{s.t.} \quad \sum_{P_1 \in \mathcal{P}} p_{P_1} = 1\,, \quad \sum_{P_2 \in \mathcal{P}} p_{P_2} = 1\,, \\
& \qquad 0 \leq p_{P_1} \leq 1\,, \quad 0 \leq p_{P_2} \leq 1, \quad \forall P_1, P_2 \in \mathcal{P}\,, \\
& \qquad \text{all constraints from Eq. (5)}\,,
\end{aligned} \tag{6}
$$

where β is a positive constant, p_{P_1} & p_{P_2} can be interpreted as the probability of choosing the allocation matrix P_1 & P_2 respectively, and $\frac{1}{\beta}\ p_{P_1} \log p_{P_1}$ & $\frac{1}{\beta}\ p_{P_2} \log p_{P_2}$ are the entropy terms. The optimality gap of the approximation is bounded by $\frac{1}{\beta}\ \log|\mathcal{P}|$, which gives us an upper bound of $\frac{1}{\beta}\ |M| \log|N|$ [6]. The approximation becomes exact for $\beta \to \infty$; however, there are problem-specific constraints on setting β arbitrarily large. We consider fog nodes and applications as autonomous decision-making entities where each entity is modeled to control one objective. Fog nodes try to decrease the energy consumption $E(P)$ and applications minimize the communication cost $G(P)$. Fog nodes and applications construct independent Markov chains and perform microservice exchange to explore different states of the Markov chains, such that each new placement lies in $\mathcal{P}$. The steady-states of the two Markov chains are given by

$$
\begin{aligned}
p^*_{P_1} &= \frac{e^{-\beta E(P_1)}}{\sum_{Q \in \mathcal{P}} e^{-\beta E(Q)}}\,, \quad \forall P_1 \in \mathcal{P}\,, \\
p^*_{P_2} &= \frac{e^{-\beta G(P_2)}}{\sum_{Q \in \mathcal{P}} e^{-\beta G(Q)}}\,, \quad \forall P_2 \in \mathcal{P}\,.
\end{aligned} \tag{7}
$$

Let P_1^* and P_2^* denote the placements obtained at the steady-state of the two Markov chains. Then optimal placement for the FSP problem, denoted by P^*, is

$$
P^* = \operatorname*{arg\,min}_{P_1^*, P_2^*} \quad (T(P_1^*), T(P_2^*))\,. \tag{8}
$$

The next step is the design of the time-reversible ergodic Markov chains, where each state corresponds to a placement $P \in \mathcal{P}$ with stationary distributions as given in Eq. (7).

5. Autonomic Placement Algorithm

In this section, we present the distributed algorithm that approximates the FSP problem. Each fog node FN_n maintains a local energy cost, and each application AP_r keeps a local communication cost due to the communication between its microservices under placement P, which are denoted by $\text{E}_n(P)$ and $\text{G}_r(P)$, respectively. The local values are determined as

$$E_n(P) = \rho_{cn}\,, \tag{9}$$

$$G_r(P) = \sum_{m_1,m_2 \in M_r} C_{m_1 m_2} \sum_{n_1,n_2 \in N} P_{m_1 n_1} D_{n_1 n_2} P_{n_2 m_2}\,. \tag{10}$$

Fog nodes compete with each other to decrease their energy consumption $E_n(P)$, and applications compete to minimize their communication costs $G_r(P)$. Consider two placements $P_1, P_2 \in \mathcal{P}$ be the states of the Markov chains constructed by the fog node FN$_n$ and the application AP$_r$, respectively. Let $\mathcal{P}'_n \subset \mathcal{P}$ be the set of all possible placements that fog node FN$_n$ can achieve by performing a state transition from placement P_1 and $\mathcal{P}'_r \subset \mathcal{P}$ represents the set of all possible placements that application AP$_r$ can obtain after performing a transition from state P_2. Let $P'_1 \in \mathcal{P}'_n$ and $P'_2 \in \mathcal{P}'_r$ be the configurations obtained by FN$_n$ and AP$_r$ respectively, as a result of state transitions in their respective Markov chains. We have $E_n(P_1)$ and $E_n(P'_1)$ (respectively, $G_r(P_2)$ and $G_r(P'_2)$) denoting the energy consumption (respectively, communication costs) of these placements. Letting $p_{n,P_1P'_1}$ and $p_{r,P_2P'_2}$ denote the transition probabilities from placement P_1 to P'_1 at node FN$_n$ and from P_2 to P'_2 by application AP$_r$, respectively, and by using [6, OPT 4], we have

$$p_{n,P_1P'_1} = \frac{e^{-\beta(E_n(P'_1)-E_n(P_1))}}{\sum_{Q'\in\mathcal{P}'_n} e^{-\beta(E_n(Q')-E_n(P_1))}}\,, \tag{11}$$

$$p_{r,P_2P'_2} = \frac{e^{-\beta(G_r(P'_2)-G_r(P_2))}}{\sum_{Q'\in\mathcal{P}'_r} e^{-\beta(G_r(Q')-G_r(P_2))}}\,. \tag{12}$$

The transition rate between the two states is proportional to the difference between the objective function in these states. Hence, the system is more likely to switch to a placement with better performance. Starting with an arbitrary initial placement, denoted by P_{init}, and changing placements according to the above transition probabilities, the Markov approximation iteratively approaches the steady-state of the Markov chain.

The time of a state transition is governed by a timer that is run at each fog node FN$_n$ and by each application AP$_r$. With placement P_1, a fog node FN$_n$ remains in its current state for an exponentially distributed random time proportional to $e^{-\beta E_n(P_1)}$ with E$_n(P_1)$ from Eq. (9). This is motivated by the fact that the time spent in a state of the created Markov chain is proportional to the steady-state probability of that state. A higher utilization on the fog node leads to a shorter time out value; thus, fog nodes with higher utilization get timeouts more often. Hence the fog nodes compete with each other to reduce their energy consumption resulting in a load balanced system. Similarly, for an application AP$_r$ under placement P_2, the timer value is proportional to e$^{-\beta G_r(P_2)}$ with $G_r(P_2)$ from Eq. (10). In order to perform a joint optimization for the FSP problem in Eq. (5), we set the value of timers for fog nodes and applications, inversely proportional to the weights given to the energy consumption and the communication cost respectively. Thus, a larger value of σ lowers the time out values at fog nodes and increases the sojourn time in a state for each application. As a result, fog nodes observe more frequent timer expiration, and at every time out, a fog node tries to decrease its energy consumption, thereby prioritizing optimization of energy consumption over communication cost. The timer values at fog node FN$_n$ with placement P_1 and application AP$_r$ with placement P_2, denoted by $Nti_n(P_1)$ and $Ati_r(P_2)$, respectively are set to

$$Nti_n(P_1) = \frac{e^{-\beta E_n(P_1)}}{\sigma}\,, \tag{13}$$

$$Ati_r(P_2) = \frac{e^{-\beta G_r(P_2)}}{1-\sigma}\,. \tag{14}$$

Since the sojourn time in a state is proportional to its steadystate probability, and since the probabilities in Eq. (7) favor states that minimize the objective function, the Markov approximation realizes a result that is or is close to the solution of Eq. (6).

We define a *state transition* as the movement of a single microservice from one node to another or swapping a pair of microservice between two fog nodes such that the resultant state is a feasible placement. However, an iterative placement strategy that constructs the Markov chain by considering all transitions between any pair of feasible placements in each step of the iteration may result in an unreasonable large migration of microservices. We, therefore, truncate the transitions performed by each fog node and application by considering transitions only to a *relevant set* of nodes. As long as all states remain reachable, the steady-state of the Markov chain is unaffected by the truncation. We refer this algorithm to as *Relevant Set Exchange*, abbreviated as *RSE*. Moreover, after performing a state transition, our RSE algorithm considers communication of the state transition only with the relevant agents, i.e., the fog nodes and applications that are affected by the migration of the microservices involved in the transition. Thus the RSE algorithm keeps the information about state changes private to the agents and does not require any global information exchange.

Next, we define the *relevant set* for each fog node FN$_n$, each microservice MS$_m$ and each application APr, labelled as $\mathcal{R}_n$,$\mathcal{R}_{mn}(P)$ and $\mathcal{R}_r(P)$ respectively. For a fog node FN$_n$, the relevant set consists of only its neighbors, i.e., all the fog nodes in the network that are one hop away, given by

$$\mathcal{R}_n = \left\{ n' \in N \mid \text{ hop count between } \mathrm{FN}_n \text{ and } \mathrm{FN}_{n'} = 1 \right\}.$$

If MS$_m$ is assigned to FN$_n$ in placement P, that is, $P_{mn} = 1$, we define the relevant set of MS$_m$ at fog node FN$_n$ in placement P, denoted by $\mathcal{R}_{mn}(P)$, as

$$\mathcal{R}_{mn}(P) = \Big\{ n' \in N \mid \exists n' \in N, m' \in M : \\ P_{mn} = 1 \wedge P_{m'n'} = 1 \wedge C_{mm'} > 0 \Big\}.$$

In other words, FN$_{n'}$ ∈ $\mathcal{R}_{mn}(P)$, if FN$_{n'}$ holds a microservice MS$_{m'}$ that has a positive chaining degree with a microservice MS$_m$ at fog node FN$_n$. The relevant sets are readily available in practice since a microservice must know the addresses of the fog nodes where the next microservices in the chain are located. Hence, when a microservice is migrated, it carries with it information on the relevant set. The relevant set of an application consists of all the fog nodes on which the microservices of the application are placed. We define the *relevant set of an application* AP$_r$ for MSs in placement P, denoted by $\mathcal{R}_r(P)$, as

$$\mathcal{R}_r(P) = \left\{ n' \in N \mid \exists n' \in N, m' \in M_r : P_{m'n'} = 1 \right\}.$$

Since the sets $\mathcal{R}_n, \mathcal{R}_{mn}(P)$ and $\mathcal{R}_r(P)$ will generally include only a small number of fog nodes, the search space for an exchange operation is quite contained. We present the steps of our distributed algorithm executed by each fog node FN$_n$ and each application AP$_r$ in the following subsections.

A. Relevant Set Exchange at Fog Nodes

With placement P_1, a fog node FN$_n$ computes $E_n(P_1)$ from Eq. (9) and starts its timer $Nti_n(P_1)$ with Eq. (13). If a timeout occurs at FN$_n$, the fog node computes $E_n(P_1')$ with Eq. (9), $\forall P_1' \in \mathcal{P}_n'$. Then, FN$_n$ selects a new placement $\hat{P_1'} \in \mathcal{P}_n'$ with probabilities given in Eq. (11). Note that a state transition can result in the addition or the removal of a microservice, or both. Therefore, a fog node is permitted to make three types of state transitions: offload one of its microservices, pull a microservice, or swap one of its microservices with a fog node $n' \in \mathcal{R}_n$. Hence, the state space $\mathcal{P}_n'$, consisting of all possible state transitions that must be considered by node FN$_n$, is polynomial. The change of the placement, given by $\Delta P = \hat{P_1'} - P_1$, is then communicated only to the relevant fog nodes and applications. In case of offloading a microservice MS$_m$ to FN$_l$, the fog nodes in $\mathcal{R}_{mn}(P_1)$ as well as fog node l are updated. If the neighbor exchange adds MS$_m$ to FN$_n$ from FN$_l$, then FN$_n$ informs fog nodes with index in $\mathcal{R}_{mn}(\hat{P_1'})$, that is, the nodes in the relevant set after the state transition as well as FN$_l$, the node from which microservice was obtained. In both cases, the application AP$_r$ | MS$_m \in M_r$ is also updated. Lastly, when FN$_n$ swaps MS$_m$ with MS$_{m'}$ on FN_l, then all the fog nodes in $\mathcal{R}_{mn}(P_1)$, $\mathcal{R}_{m'n}(\hat{P_1'})$, and l get an update message. In this case applications AP$_r$ | MS$_{m \,\|\, m'} \in M_r$ are updated. After that FN$_n$ restarts its timer $Nti_n(P_1)$ with Eq. (13). When a fog node FN$_k$ receives the update ΔP, it updates P_1, re-computes $E_k(P_1)$ with Eq. (9) and restarts its timer $Nti_k(P_1)$ with Eq. (13). Algorithm 1 summarizes the operations performed by FN$_n$ upon initialization, timeout, and receipt of an update. Since the timeout value in a state is proportional to the probability of the state, fog nodes in the steady-state tend to be in a placement that is close to the minimum energy consumption. Even in the steady-state, fog nodes continue to perform neighbor changes, albeit at a lower frequency. This can be prevented by adding a criterion that terminates the algorithm if a change of the total cost after an update does not exceed a given threshold.

B. Relevant Set Exchange by Applications

For an application AP$_r$ with placement P_2, we restrict the transitions of microservices only between the fog nodes in the set $\mathcal{R}_r(P_2)$, i.e., only between the fog nodes on which the microservices of application AP$_r$ are present. After every

Algorithm 1 Relevant Set Exchange at Fog Nodes

```
Initialize:
$P_1 \leftarrow P_{\text{init}}$
Compute $E_n(P_1)$ with Eq. (9) and start timer $Nti_n(P_1)$ with Eq. (13)

Upon timeout:
Compute $E_n(P_1')$ with Eq. (9) for all $P_1' \in \mathcal{P}_n'$
Choose next state $\hat{P}_1'$ with Eq. (11)
$\Delta P \leftarrow \hat{P}_1' - P_1$
if an MS$_m$ was migrated from FN$_n$ to FN$_l$ then
    Send $\Delta P$ to fog nodes FN$_k$ with $k \in \mathcal{R}_{mn}(P_1) \cup \{l\} \cup \text{AP}_r \mid \text{MS}_m \in M_r$
end if
if an MS$_{m'}$ was migrated from FN$_l$ to FN$_n$ then
    Send $\Delta P$ to fog nodes FN$_k$ with $k \in \mathcal{R}_{m'n}(P_1') \cup \{l\} \cup \text{AP}_r \mid \text{MS}_{m'} \in M_r$
end if
Perform relevant set exchange, $P_1 \leftarrow \hat{P}_1'$
Restart timer $Nti_n(P_1)$ with Eq. (13)

Upon receiving $\Delta P$:
$P_1 \leftarrow P_1 + \Delta P$
Compute $E_n(P_1)$ with Eq. (9) and restart timer $Nti_n(P_1)$ with Eq. (13)
```

transition, it is sufficient to notify nodes in $\mathcal{R}_r(P_2')$ by sending ΔP through an update message. Subsequently, the fog nodes in $\mathcal{R}_r(P_2')$ and AP$_r$ restart their timers. By restarting the timer values of relevant agents, we can largely avoid the algorithm from getting stuck in some local minima, as it ensures that the same set of fog nodes and applications does not get frequent timeouts. Algorithm 2 shows the operations of RSE run by applications.

Algorithm 2 Relevant Set Exchange by Applications

```
Initialize:
$P_2 \leftarrow P_{\text{init}}$
Compute $G_r(P_2)$ with Eq. (10) and start timer $Nti_r(P_2)$ with Eq. (14)

Upon timeout:
Compute $G_r(P_2')$ with Eq. (10) for all $P_2' \in \mathcal{P}_r'$
Choose next state $\hat{P}_2'$ with Eq. (12)
$\Delta P \leftarrow \hat{P}_2' - P_2$
Send $\Delta P$ to fog nodes FN$_k$ with $k \in \mathcal{R}_r(\hat{P}_2')$
Perform relevant set exchange, $P_2 \leftarrow \hat{P}_2'$
Restart timer $Nti_r(P_2)$ with Eq. (14)

Upon receiving $\Delta P$:
$P_2 \leftarrow P_2 + \Delta P$
Compute $G_r(P_2)$ with Eq. (10) and restart timer $Nti_r(P_2)$ with Eq. (14)
```

As an example, consider Fig. 3, where the figure in the middle depicts a placement, P, of application AP$_1$ consisting of five microservices on a network of four fog nodes. Here, the relevant set for FN$_1$ is $\mathcal{R}_1 = \{2, 3\}$, the relevant set for MS$_1$ at FN$_1$ is $\mathcal{R}_{11}(P) = \{2\}$, and that of MS$_4$ at FN$_3$ is $\mathcal{R}_{43}(P) = \{2, 4\}$. If FN$_1$ pushes MS$_1$ to FN$_2$, as shown in the Figure 3 in the left, then the migration only affects the fog nodes in the relevant set $\mathcal{R}_{11}(P)$ and application AP$_1$. Therefore, it is sufficient to notify FN$_2$, and application AP$_1$ about the transfer, subsequently, FN$_1$, FN$_2$ and application AP$_1$ restart their timers. Similarly, the relevant set of AP$_1$ is $\mathcal{R}_1(P) = \{1, 2, 3, 4\}$. Application AP$_1$, perform an exchange between the nodes in $\mathcal{R}_1(P)$ by pushing MS$_4$ from FN$_3$ to FN$_2$ as shown in the Figure 3 in the right. As a result, all the fog nodes in the new relevant set $\mathcal{R}_1(P') = \{1, 2, 4\}$ gets an update message and, subsequently, application AP$_1$, and all the fog nodes in $\mathcal{R}_1(P')$ restart their timers.

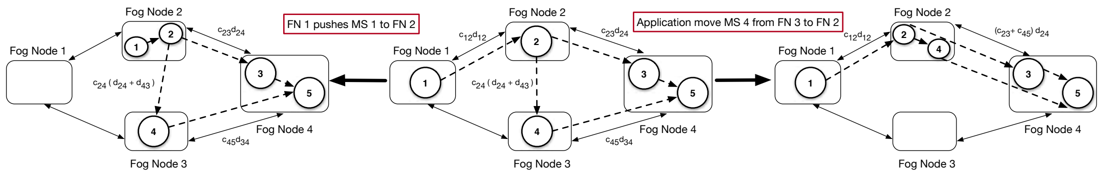
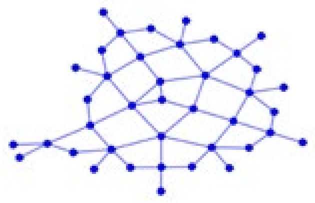
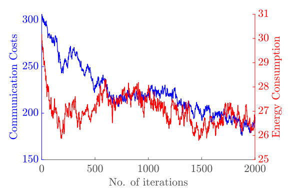


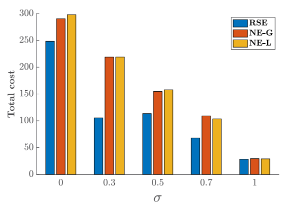
Fig. 3. Relevant Set Exchange.



(a) Network of fog nodes.



(b) Convergence of joint optimization.



(c) Comparison with previous methods.

Fig. 4. Results for fog network in Fig. 4a.

5. Evaluation

We next present a numerical evaluation of our proposed RSE algorithm, where we evaluate (1) the convergence properties of the constructed Markov chains, (2) how they achieve their objective compared to the previous autonomic methods. Since, with the given space constraints, it is not feasible to present an exhaustive evaluation, we attempt to create numerical examples that are representative and address the most pressing questions. For the examples, we consider a network with 40 fog nodes, as shown in Fig 4a. We create a heterogeneous fog network by setting the CPU capacity of fog nodes uniform random in the range [100,500] MIPS; We do not account for other resources in the examples. We consider ten applications, each with ten microservices, with heterogeneous CPU requirements selected uniform random from the range [50,150] MIPS. The chaining costs C_{ij} of two microservices MS_i and MS_j in the same application are selected uniform random from the interval (0,1) and only microservices belonging to the same application can have a nonzero chaining cost between them. We run our RSE algorithm for 2000 iterations with $\sigma = 0.5$, Fig. 4b on the left axis shows the convergence of Markov chain constructed by applications and on right axis shows the convergence of the algorithm at fog nodes. Fig. 4c compares the total cost in terms of $T(P)$ from Eq. (5) for different values of parameter σ achieved by RSE algorithm after 2000 iterations, to our previous autonomic algorithms (NE-G and NE-L) from [7]. The results show that the proposed RSE algorithm consistently outperforms the previous methods.

6. Conclusion

We have proposed a distributed service placement method of microservice-based applications in the fog environment, which seeks to jointly optimize energy consumption and utilization of network resources. Different from centralized service placements methods for clouds, the proposed Relevant Set Exchange algorithm does not require central control or global state information. The design of the algorithm models the combinatorial optimization objective as a game and applies the Markov approximation method for obtaining a nearoptimal solution. Simulation results showed that the proposed algorithm is superior to our previous autonomic placement algorithms.

References

1. Butzin, B., Golatowski, F., & Timmermann, D. (2016, September). Microservices approach for the internet of things. In 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA) (pp. 1-6). IEEE.
2. Mann, Z. Á. (2015). Allocation of virtual machines in cloud data centers—a survey of problem models and optimization algorithms. Acm Computing Surveys (CSUR), 48(1), 1-34.
3. Lopez-Pires, F., & Baran, B. (2015). Virtual machine placement literature review. arXiv preprint arXiv:1506.01509.
4. Herrera, J. G., & Botero, J. F. (2016). Resource allocation in NFV: A comprehensive survey. IEEE Transactions on Network and Service Management, 13(3), 518-532.
5. Li, X., & Qian, C. (2016, January). A survey of network function placement. In 2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC) (pp. 948-953). IEEE.
6. Chen, M., Liew, S. C., Shao, Z., & Kai, C. (2013). Markov approximation for combinatorial network optimization. IEEE transactions on information theory, 59(10), 6301-6327.
7. Kayal, P., & Liebeherr, J. (2019, June). Autonomic service placement in fog computing. In 2019 IEEE 20th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM) (pp. 1-9). IEEE.
8. Kayal, P., & Liebeherr, J. (2019, July). Distributed service placement in fog computing: An iterative combinatorial auction approach. In 2019 IEEE 39th International Conference on distributed computing systems (ICDCS) (pp. 2145-2156). IEEE.
9. Jiang, J. W., Lan, T., Ha, S., Chen, M., & Chiang, M. (2012, March). Joint VM placement and routing for data center traffic engineering. In 2012 Proceedings IEEE INFOCOM (pp. 2876-2880). IEEE.
10. Pham, C., Tran, N. H., Ren, S., Saad, W., & Hong, C. S. (2017). Traffic-aware and energy-efficient vNF placement for service chaining: Joint sampling and matching approach. IEEE Transactions on Services Computing, 13(1), 172-185.
11. Pallewatta, S., Kostakos, V., & Buyya, R. (2019, December). Microservices-based IoT application placement within heterogeneous and resource constrained fog computing environments. In Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing (pp. 71-81).
12. Kayal, P., & Liebeherr, J. (2019, October). Poster: Autonomic service placement in fog computing. In Proceedings of the 2019 on wireless of the students, by the students, and for the students workshop (pp. 17-17).
13. Rivoire, S., Ranganathan, P., & Kozyrakis, C. (2008). A comparison of high-level full-system power models. HotPower, 8(2), 32-39.