Annals of Telecommunications (2018) 73:463–474
https://doi.org/10.1007/s12243-018-0649-0

Placing it right!: optimizing energy, processing, and transport in Edge-Fog clouds

Nitinder Mohan[1] · **Jussi Kangasharju**[1]

Received: 8 May 2017 / Accepted: 8 May 2018 / Published online: 28 May 2018

Abstract

In recent years, applications such as Internet-of-Things has proliferated the Internet to a great extent. Such applications derive data from a significant number of smart sensors sensing information from the environment. Due to an extensive data footprint, the demand for cloud services to process this data has also increased. However, traditional centralized cloud model requires offloading data from these sensors over a network which induces significant network delay on these applications. Several architectural abstractions of cloud, such as Fog and Edge, have been proposed to localize some of the processing near the sensors and away from the central cloud servers. In this paper, we propose Edge-Fog cloud which distributes task processing on the participating cloud resources in the network. We develop the Least Processing Cost First (LPCF) method for assigning the processing tasks to nodes which provide the optimal processing time and near-optimal networking costs. We further provide an energy-efficient variant of LPCF, i.e., eLPCF algorithm, which optimizes energy usage while calculating task deployment in Edge-Fog cloud. We evaluate both LPCF and eLPCF in a variety of scenarios and demonstrate its effectiveness in finding the processing task assignments.

Keywords Cloud computing · Fog computing · Edge computing · Internet-of-Things · Task assignment

1 Introduction

Cloud computing has created a radical shift in application computation and has evolved to provide low-cost and highly scalable computing services to its users. The cloud service providers deploy a network of large data centers spread across the globe. Applications with high computational requirements offload its processing tasks to a cloud service which first distributes the task to one/several server(s) in a data center after which the results are sent back to the end-user. However, the time complexity requirements for processing a job are stringent for some applications more than others.

For example, Internet-of-Things (IoT) domain involves a large number of *smart* sensors sensing information from the environment and uploading it to a cloud service for processing. A recent study by National Cable & Telecommunications Association (NCTA) assumes that close to 50.1 billion IoT devices will be connected to the Internet by 2020 [1]. More often than not, handling computations of IoT applications on *traditional* cloud computing models adds following delays to overall processing. (i) The network delay in offloading required data to a central cloud impacts the overall completion time of time-critical IoT applications, and (ii) as the computation is dependent on data from not one, but multiple end data generators and sensors, uploading data from all such nodes may induce network congestion and add to pre-existing network delay.

To decouple the network delay from the computation time for such data, researchers have proposed bringing the *compute cloud* closer to data generators and consumers. Several models have been proposed which achieve computation at the "network edge". *Fog computing cloud* [2] proposes network devices such as routers to run cloud application logic. The objective of Fog cloud is to perform low-latency computation/aggregation on the data while routing it to the central cloud for heavier computation [3, 4]. On the other hand, *Edge computing cloud* [5] takes inspiration from

✉ Nitinder Mohan
nitinder.mohan@helsinki.fi

Jussi Kangasharju
jussi.kangasharju@helsinki.fi

[1] Department of Computer Science, University of Helsinki, Helsinki, Finland

projects such as SETI@Home and Folding@Home [6, 7], and proposes a consolidation of human-operated, voluntary resources such as desktop PCs, tablets, smart phones, and nano data centers as a cloud. As the resources in Edge cloud usually lie in one-hop proximity to the IoT sensors; processing the data at the edge can significantly reduce the network delay [8, 9]. Several other approaches such as Cloudlets, Mobile Edge Computing (MEC) have also been considered by the researchers [10–12]. Both MEC and Cloudlets aim to incorporate computationally capable Edge device to become part of existing cloud infrastructure. Specifically, MEC enables the existing cellular infrastructure to support Edge resources whereas Cloudlets utilize end mobile devices as light-weight cloud instances at the network edge.

The cloud models mentioned above extend the pre-existing centralized cloud architecture to include the computational resources at the network edge. Such architectures significantly rely on placement algorithms to find the "optimal" task placement considers attributes such as processing requirements, network requirements, and energy consumption. However, we argue that due to the availability of highly scalable centralized data center, more often than not, the placement algorithms skew task deployment on central cloud and are unable to fully utilize the availability and potential of Edge computing resources [13]. We further argue that due to the semi-dependence of existing Edge cloud models on a central cloud, the models only utilize the Edge resources to perform pre-processing tasks at the edge and rely on a centralized cloud for heavy, computationally intensive tasks.

In this paper,[1] we present a node-oriented, fully decentralized hybrid of Edge and Fog compute cloud model, *Edge-Fog cloud*. The Edge-Fog cloud comprises of two layers of compute-capable resources, namely Edge and Fog, and the core of the cloud is only responsible for storing data. Due to its decentralized architecture, the Edge-Fog cloud is capable of decoupling processing time from network delays by effectively handling processing close to the data generators. Edge-Fog cloud offers reliable data storage of raw and computed data at the central data store located at the core of its architecture. Specifically, the contributions we make in this paper are as follows:

(i) We present Edge-Fog cloud architecture, which is based on classifying compute resources into Edge and Fog layers, depending on their capabilities, ownership, and functionality.
(ii) We design task deployment algorithms which assign tasks on the available nodes in the Edge-Fog cloud while minimizing the processing time, network costs, and energy usage per resource. We show that our algorithms achieve near-optimal networking costs in polynomial time as opposed to exponential time complexity achieved by related work.
(iii) We develop an Edge-Fog cloud simulator and integrate it with our assignment solvers. We demonstrate and compare the efficiency of our solvers with its related works across a range of parameters and simulations.
(iv) We discuss and provide insights regarding the characteristics of Edge-Fog cloud that will affect its performance in real-world.

The remainder of the paper is organized as follows. In Section 2, we describe the Edge-Fog cloud architecture. In Section 3, we propose our task deployment algorithm which minimizes several costs. In Sections 4 and 5, we evaluate and discuss the effectiveness of our algorithm. We discuss the related work in Section 6. Section 7 concludes the paper.

2 Edge-Fog cloud

2.1 Architecture

In this section, we describe the architecture of the Edge-Fog cloud. The cloud model is composed of several computationally capable resources functioning at several abstraction layers. We organize the resources in abstraction layers by categorizing them on a certain set of device attributes such as processing power and network distance from the client. We explain the Edge-Fog cloud model through two point-of-view architectures, (i) resource-oriented view and (ii) network-oriented view as shown in Figs. 1 and 2, respectively. The Edge-Fog cloud is a consolidation of three different cloud models comprising of different resource types.

Edge layer The outermost layer of the cloud is Edge layer. The Edge is a collection of loosely coupled, voluntary[2]

[1] This paper is an extended version of a paper with the same title published at International Conference on Cloudification of Internet of Things in 2016 [14]. The work proposed a unique solution to assignment problem which considered network and processing costs of jobs on the cloud. However, the proposed solution was unable to incorporate any other cost models which are highly relevant and motivate real-world deployments of edge clouds. The novel contributions of the current manuscript focus on energy dissipation which is real-world highly significant and a pressing issue, especially in battery-operated resources. This paper leverages the previously proposed solution and presents an extended algorithm which also incorporates energy dissipation costs of cloud resources while computing an assignment. We further implement, evaluate, and compare our proposed solution to our previous and related work. We also extend our Edge deployment discussion and analyze the effects of resource types on overall job deployment.

[2] Several incentive/credit mechanisms similar to crypto-currency mining mechanisms can be employed for devices to volunteer as Edge resource [15]. However, discussion of such mechanisms is currently out-of-scope of this paper.

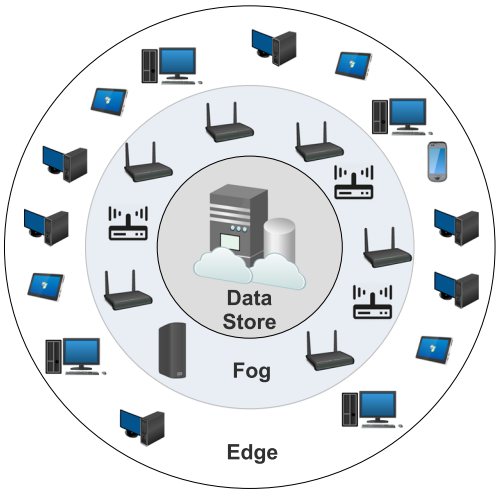


Fig. 1 Resource-oriented Edge-Fog cloud architecture

and human-operated resources such as desktops, laptops, nano data centers, and tablets. As shown in Fig. 2, the Edge layer lies extremely close to the end-clients and data sources, with Edge resources catering end-users at one/two-hop distance away. Their proximity to the network *edge* makes Edge layer highly desirable to compute and deploy highly time-sensitive data, in-network service functions, etc. [16]

Edge resources have varying ranges of computational capabilities from highly capable devices such as workstations and nano data centers to less capable such as tablets or smart phones. Edge devices, however, are extremely location coupled and have possibility to establish device-to-device connectivity with other nearby Edge resources. However, the layer itself is fragmented into several "groups" according to location availability and connectivity.

Fog layer The next layer after Edge is the Fog computing layer. Fog is a consolidation of networking devices such as routers and switches with high computing capabilities and ability to run cloud application logic on their native architecture. We envision Fog resources to be manufactured, managed and deployed by cloud vendors (such as CISCO [17]). Further, as shown in Fig. 2, the Fog layer also extends itself to the core of the network, i.e., core routers and regional switches and ends up forming the networking backbone of the Edge-Fog architecture. The computational resources in Fog layer are interconnected with high-speed, reliable links and are more geographically diverse connected.

The Edge-front of the Fog layer acts as first point-of-contact with the rest of the network. The Fog resources are not only capable of routing data to other suitable Edge/Fog resources, but also provide computation throughout it. Compared to the Edge layer, the Fog resources reside farther from the edge of the network when compared to Edge layer but significantly closer than a traditional central cloud. Due to static positioning in the model, the computational capability of Fog resources far exceeds that of the Edge. Therefore, these devices can be is used to effectively handle computationally intensive tasks offloaded by Edge resources.

Data store Unlike the traditional cloud model, we model the core of the Edge-Fog cloud to have no computational capabilities but only to serve as a repository for archiving all computational data in the cloud. Our reasoning for detaching computation from central cloud is threefold. First, the sheer accumulated power of computationally capable resources in Edge and Fog layers makes it competent to support any task. Second, as these devices have significantly lower delay to end-client/data source, the need of routing data to a central cloud due to processing requirements disregards the advantages of having Edge and Fog devices in the first place. Lastly, providing computations on Edge and Fog resources is more energy-efficient than a tradition cloud-server [18].

A centralized store provides reliability and easy access to data by any computing resources in the cloud. Being at the core of the architecture, the Data Store is accessible by both Edge and Fog layers and can act as a computational and data checkpoint for distributed processing undertaken by Edge and Fog layers [19].

2.2 Benefits of the Edge-Fog cloud

The Edge-Fog cloud offers several benefits:

1. *Reduction in network load*: The Edge-Fog cloud provides computation at the edge of the network near

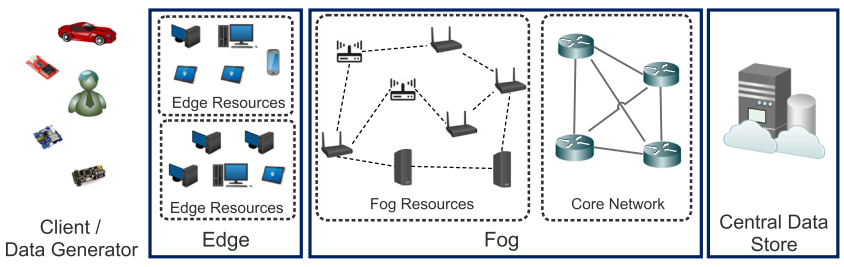
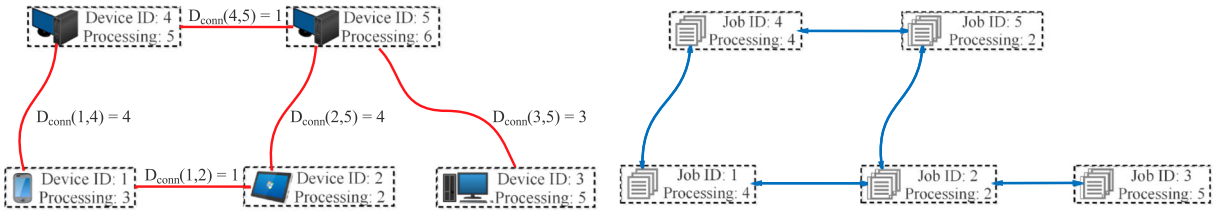


Fig. 2 Network-oriented Edge-Fog cloud architecture



(a) Graphical representation of Edge-Fog cloud comprising of five member resources

(b) Collection of tasks with their computational dependencies

Fig. 3 Task deployment example. Each job in **b** needs to be deployed on a resource in **a**

the data generators and end-client. The Edge-Fog cloud thus reduces the amount of data that flows into the network thereby decreasing the overall processing time.

2. *Context-Aware provisioning*: Resources in Edge-Fog cloud also provide contextual awareness to data generated by end-sensors. These can be received in different formats such as environment-based (location, time), user (activity, interactions with the environment), and device-based (battery, network strength). The Edge resources can utilize the pre-existing context or even add its own to data under process.
3. *Native support for mobility*: Mobility along with reliability is a quintessential requirement for many IoT applications. Edge resources such as smartphones or laptops can offer native physical and virtual mobility for supporting such mobile IoT applications.
4. *Reliability*: As computation in Edge-Fog cloud is decentralized; the model has no single point of failure. To provide added reliability in such a system, several snapshots of an application may be deployed on the cloud. Furthermore, the Edge-Fog resources can significantly improve the availability of performance, secured resource-to-resource interactions, etc.
5. *Inter-operability*: Both Edge and Fog resources can be utilized to perform traditional computational and networking activities such as user-interaction, packet routing, and switching, at both user and commercial-scale, respectively. Specific tools can be utilized in Edge-Fog which can provide self-customization to these resources in cases when they also need to fulfill their primitive duties along with cloud computation.

Applications such as connected vehicles, energy monitoring, and automated traffic control can highly benefit from Edge-Fog cloud as most of the tasks in such applications are distributed and network-constrained.

3 Task deployment on Edge-Fog cloud

The Edge-Fog cloud is a scalable platform for a large number of interconnected Edge and Fog devices and efficiently utilize the processing power they offer.[3] However, as the resources in the Edge-Fog cloud are governed by their processing and network capabilities, deploying tasks on these devices has an associated cost. A typical task deployment algorithm must map a job node from the job graph to an Edge/Fog resource. The cost of deployment is dependent on both the properties of resources and that of the deployed task itself. For example, if a job requires more coordination with its peers for completion, the associated network cost for completing that job will be higher. In order to provide a scalable and efficient solution, the task deployment algorithm for Edge-Fog cloud should find the deployment snapshot with least possible cost without unduly impacting the overall completion time of that process.

Figure 3 shows a snapshot of Edge-Fog cloud with five computing resources. Each resource is defined in the cloud with its unique device ID and *available* processing capability (in number of processing cores).[4] We use the notation D_i to refer to a device with device ID i. Certain resources in the Edge-Fog cloud are connected to a network with certain communication cost denoted by $D_{conn}(i, j)$. For example, $D_{conn}(1, 2)$ refers to network cost between devices D_1 and D_2. Figure 3b shows the job graph deployed on the Edge-Fog cloud. Every job in the graph is denoted by a job ID and minimum required processing capability to complete that job. Further, we assume two-way dependency between the jobs wherein Job J_1 and J_2 are dependent on each other if there exists a connection between them.[5]

In this paper, we assume that the number of tasks is equal to the number of devices on which it must be deployed. In case of unequal devices and tasks, the deployment algorithm can equalize the number of nodes in device and job graphs in the following manner. First, if the number of

[3] Throughout the paper, we use the terms devices and resources interchangeably.

[4] It must be noted that as a resource may be involved in more than one task deployments simultaneously, the processing ability of a resource refers to its currently available processing power prior to next task deployment

[5] In this paper, we only consider a non-weighted job dependencies and leave the weighted job graph mapping for future work.

jobs in a job graph is more than the number of available devices, the algorithm may split existing devices into virtual devices/machines to retain the 1:1 mapping. On the other hand, if the number of resources is more than the number of jobs, we ignore the exceeding devices. For the initial selection of devices, the deployment algorithm can utilize one of several sorting/selection algorithms such as first come first served and greedy. Both of the techniques as mentioned earlier have their in-grain complexities and difficulties considering an environment comprising of a diverse set of compute-capable resources [20]. However, we consider providing a formal solution to the problems mentioned above to be outside the scope of this paper and intend to tackle it in our future work.

3.1 Network-Only Cost

Previous works have tried to model task deployment algorithms which minimize the associated networking cost [21]. The formal definition of such task assignment strategies is to find an assignment which places $\mathcal{N}$ jobs on $\mathcal{N}$ devices such that the associated network cost is minimized. For the rest of the paper, we refer to such algorithms as *Network-Only Cost (NOC)* algorithms. Let $D_{conn}(i, j)$ represent the cost of connectivity between the devices D_i and D_j and $J_{conn}(i, j)$ denote the dependency between the jobs J_i and J_j. Both D_{conn} and J_{conn} are square matrices of size $\mathcal{N} \times \mathcal{N}$. *f(i)* signifies the constraint of assigning a particular job to a device. If two mutually dependent jobs are deployed on two edge resources, they will incur the network cost pre-existing between the two resources to achieve completion.

With $\mathcal{N}$ devices/jobs, the search space of possible assignments in NOC is $\mathcal{N}!$. For example, in Fig. 3, the assignment $D_1 \rightarrow J_1$; $D_2 \rightarrow J_2$; $D_3 \rightarrow J_3$; $D_4 \rightarrow J_4$; $D_5 \rightarrow J_5$ has network cost 17, whereas, the assignment $D_1 \rightarrow J_4$; $D_2 \rightarrow J_5$; $D_3 \rightarrow J_3$; $D_4 \rightarrow J_1$; $D_5 \rightarrow J_2$ has cost 13. A naive NOC implementation would iteratively search for the assignment with least possible cost in the entire search space thus having the worst case complexity of $O(\mathcal{N}!)$. On the other hand, NOC closely resembles the well-known Quadratic Assignment Problem (QAP) [22]. QAP generalizes minimal network cost assignment as:

$$NC_{min} = \sum_{i,j \in A} J_{conn}(i, j) * D_{conn}(f(i), f(j)) \quad (1)$$

where A is set of all arcs in the graph.

However, QAP is an NP-hard problem, and its solution can only be approximated by applying constraints. Computing the optimal deployment for a problem space of 30 nodes using QAP may take up to a week on a computational grid comprising of 2500 machines [23]. Branch-and-bound-based algorithms such as Gilmore-Lawler Bound (GLB) or Hungarian bounds can estimate the solution for small-sized QAP problems. Since the job scheduling on an Edge-Fog cloud may encompass computing an assignment of hundreds of devices, a more efficient algorithm for finding an optimal task assignment is needed.

3.2 Least Processing Cost First

As the Edge resources of the Edge-Fog cloud may not be highly processing-capable, the task assignment algorithm must also consider the associated processing cost of deployment. We thus propose Least Processing Cost First (LPCF), a task assignment solver which first minimizes processing cost of the assignment and further optimizes the network cost. LPCF algorithm flow is shown in Fig. 4 and explained in detail below.

3.2.1 Optimize processing time

LPCF calculates the processing cost associated with each possible assignment in the search space. The processing cost minimization function used by LPCF is:

$$PC_{min} = \sum_{i,j \in A} \left(\frac{J_{size}(i)}{D_{proc}(j)} \right) x_{ij} \quad (2)$$

where C denotes the overall cost function; J_{size} and D_{proc} are matrices of size $1 \times \mathcal{N}$ representing the job sizes and the processing power of involved devices respectively. x_{ij} is a binary job assignment variable.

Equation 2 is an objective function of Linear Assignment Problem (LAP) which unlike QAP, is polynomial [24]. Algorithms such as Kuhn-Munkres/Hungarian guarantee an optimal solution for this problem in $O(n^3)$ (worst case). At the first step of LPCF, the algorithm will output job-to-resource deployment which has the least possible processing cost. For example, the assignment $D_1 \rightarrow J_1$;

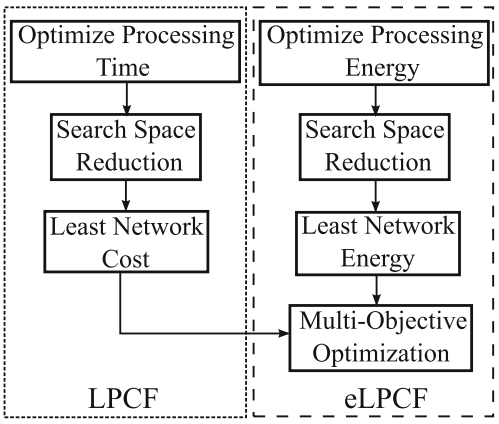


Fig. 4 LPCF and eLPCF algorithm

$D_2 \rightarrow J_2$; $D_3 \rightarrow J_3$; $D_4 \rightarrow J_4$; $D_5 \rightarrow J_5$ in Fig. 3 has the least processing cost of deployment, i.e., 5.97.

3.2.2 Problem search space reduction

As discussed in Section 2, the Edge and Fog layers in the Edge-Fog cloud model groups the compute resources on common attributes including processing capabilities. In this step of LPCF, the deployment algorithm builds a problem sub-space (much smaller than original problem space) by interchanging jobs assigned to resources of equal processing capabilities and interchanging resources of jobs requiring similar processing. In doing so, the resulting deployments retain the associated processing cost to that computed in the first step. To illustrate, in Fig. 3, the processing cost computed in first step of LPCF remains the same if we interchange the jobs deployed on D_1 and D_4 *or* interchange the resources computing J_2 and J_5. Table 1 shows the reduction in problem search space achieved by LPCF compared to a NOC deployment algorithm.

3.2.3 Selecting assignment with least network cost

In this step, LPCF *iteratively* computes the network cost associated with each assignment in the reduced problem search space and chooses the one with least network cost following Eq. 1. It should be noted that as the optimal assignment is updated at each iteration of the exhaustive search of sub-search space, a branch-and-bound variant of the algorithm can find the assignment within a time bound for large search space sizes.

Overall the job deployment computed by LPCF is assured to have the least associated processing cost and an *almost* optimal network cost. Furthermore, LPCF holds several other advantages over NOC-based algorithms. The most fundamental of them is that unlike the NOC assignment, LPCF guarantees an assignment in polynomial time thus significantly reducing the deployment calculation time. Moreover, as not all devices in the Edge-Fog cloud are highly processing capable, LPCF also takes into account the processing cost of the assignment.

3.3 Energy efficient LPCF

An added benefit of Edge cloud environments over a traditional centralized cloud is that such cloud platforms mitigate the high energy requirements imposed by large-scale data centers. Several researchers have shown mobile/edge compute resources achieve a significant reduction in energy usage without sacrificing on computational power due to their smaller sizes and efficient power management algorithms [16, 18]. However, such devices are often limited by their battery capacity and power availability which must also be considered by a deployment algorithm while placing tasks for computations.

Table 1 Problem search space reduction in LPCF

Topology size	5	10	15	30	60	100	150
NOC	5!	10!	15!	30!	60!	100!	150!
LPCF	1!	3!	>4!	>5!	>7!	>8!	>9!

In this section, we provide an energy-efficient variant of LPCF algorithm. Energy efficient LPCF (eLPCF) minimizes the total energy utilized per deployment in Edge-Fog cloud. We denote total energy usage of a device as the sum of energy used for processing the task and for communicating over the network. Formally, the model can be represented as

$$E_{total} = \sum_{i \in \mathcal{N}} [E_{proc}(i) + E_{netw}(i)] \tag{3}$$

where,

$$E_{proc}(i) = PC(i) * e_p \qquad \forall i \in \mathcal{N} \tag{4}$$

$$E_{netw}(i) = NC(i) * e_n \qquad \forall i \in \mathcal{N} \tag{5}$$

Here, $NC(i)$ and $PC(i)$ denote network and processing cost of resource i in the deployment which has been earlier formulated in Eqs. 1 and 2, respectively. Both $NC(i)$ and $PC(i)$ are dependent on the job J being deployed on a device D_i. e_n and e_p signify networking and processing cost-to-energy conversion metrics for a device D_i, the values of which are dependent on a resource's hardware. For example, e_n denotes the energy utilized for transferring a unit data via NIC. The value of e_n thus varies for chipset-to-chipset and for wired or wireless networking. Similarly, e_p denotes power dissipation of a CPU for running one CPU-cycle and varies on CPU architecture and power management algorithms. Generally, e_p of certain edge resources will be significantly lower than fog as mobile and low powered devices use stringent algorithms to better handle their power dissipation than workstations [25].

eLPCF is LPCF inspired algorithm which also minimizes the energy cost along with processing and networking costs for a deployment. The algorithm stages are depicted in Fig. 4 and explained in detail below.

3.3.1 Optimize energy due to processing

The first step of eLPCF decouples E_{total} into its two counterparts and computes a deployment which minimizes E_{proc} as modelled in Eq. 4. E_{proc} is a linear function of PC with resource dependent e_p. This transforms the energy model into a LAP, which is computed similarly as the first

step of LPCF. eLPCF does not consider minimization of E_{netw} in the first step as network energy is dependent on the task placement which is a QAP. eLPCF utilizes a Kuhn-Munkres based algorithm which guarantees deployment with least E_{proc} in $O(n^3)$.

3.3.2 Reducing problem space size

eLPCF builds on the assumption of Edge-Fog cloud consisting of several devices with common processing, energy and transport properties are executing jobs of similar processing requirements. eLPCF exploits the resulting homogeneity in resource's hardware and the deployed job graph to build its reduced sub-problem search space (similar to Step 2 of LPCF). The algorithm uses the same approach as discussed in Section 3.2.2 and swaps jobs on devices with same E_{proc}. This leads to the formation of several deployments with varied network dependencies but maintains the lowest E_{proc} as computed in the first step. It is to be noted that as E_{proc} is a product of processing cost and e_p, deployments ensuring lowest E_{proc} does not, in turn, ensure lowest processing cost.

3.3.3 Accounting energy due to networking

In this step, eLPCF computes E_{netw} of all deployments in sub-problem search space based on model in Eq. 5. The algorithm then does an iterative search of the problem space and chooses the assignment with least E_{netw} cost. As mentioned in LPCF algorithm, a *branch-and-bound* variant of this step can be deployed which can find the *least possible* E_{netw} cost within a defined time bound.

3.3.4 Multi-objective optimization

The deployment computed by last step of eLPCF ensures least E_{proc} and *almost* optimal E_{netw}. However, as total energy used is heavily dependent on the cost-to-energy metrics of resources itself (e_p and e_n), the resulting deployment can have induced high processing and network cost which may severely affect overall completion time of the application. Moreover, as eLPCF deployment is based on the initial assignment which minimizes E_{proc}, the resulting sub-problem search space may exclude several deployments with almost optimal E_{proc} and PC.

Therefore, in this step eLPCF combines the optimal cost deployments computed by LPCF and optimal energy deployments computed by the previous stage in a new search space. eLPCF then tries to find the deployment with optimal E_{total}, PC, and NC with more weight to E_{total}. With this step, eLPCF ensures that the resulting deployment not only has optimal energy cost but also does not impact the application run-time on the cloud.

4 Evaluation

We now evaluate the computation complexity for deploying jobs on several different Edge-Fog topologies. We have designed and implemented an Edge-Fog cloud simulator in Python (simulator code is available at [26]). The simulator generates a network of Edge and Fog resources and a job dependence graph based on several user-defined parameters. Table 2 shows the default parameter values we use for evaluating Edge-Fog cloud in this paper.

We further implement and integrate LPCF and eLPCF task assignment solver in the Edge-Fog cloud simulator. To compare, we measure the performance of LPCF, eLPCF, and two variants of NOC task assignment solver, permutation-based and QAP-based. For the QAP-based variant of NOC, we use an open-source implementation of Kuhn-Munkres solver available from QAPLIB [22]. We use Platypus [27] to implement the multi-objective optimization function in Python.

4.1 Completion time analysis

We analyze the overall processing time for computing an assignment by LPCF and NOC algorithms for several problem sizes. We set the maximum completion time of computation to one hour. The results are in Table 3.

It is evident from the results that LPCF performs much better than both NOC-based solvers. For $\approx$ 30 node topology, where both solvers are unable to find an optimal assignment within the time limit, LPCF computes its assignment in under a second. For large topologies of

Table 2 Default parameter values of Edge-Fog cloud simulator

Properties	Value
Total number of devices/jobs	Experiment specific
Number of Edge devices	60% of total
Number of Fog devices	40% of total
Processing power of Edge resources	2–5
Processing power of Fog resources	7–9
Connection density in Edge layer (0–1)	0.2
Connection density in Fog layer (0–1)	0.6
Connection density between Edge and Fog layer (0–1)	0.5
Lowest job size in job pool	2
Highest job size in job pool	6
Inter-dependence density between jobs (0–1)	0.2
Processing energy metric (e_p) of Edge	30–70
Processing energy metric (e_p) of Fog	60–180
Networking energy energy metric (e_n)	50

Table 3 Optimal assignment computation time

Topology size =	5	10	15	20	30	40	50	60	100	150
NOC permutation solver	0.068 s	23 m 20 s	>1 h	>1 h	>1 h	>1 h	>1 h	>1 h	>1 h	NA
NOC QAP solver	0.026 s	36.273 s	3 m 22 s	18 m 38 s	>1 h	>1 h	>1 h	>1 h	>1 h	NA
LPCF	0.0005 s	0.002 s	0.044 s	0.073 s	1.9 s	13.229 s	51.686 s	2 m 36 s	12 m 24 s	>1 h
eLPCF	0.0011 s	0.0116 s	0.08 s	0.126 s	3.28 s	22.061 s	1 m 14 s	3 m 52 s	24 m 43 s	>1 h

$\approx$ 150 nodes, LPCF exceeds the maximum allotted time for the computing an optimal assignment. The primary reason for this increased computation time is due to the large size of the reduced search space size in LPCF. The current implementation of LPCF iteratively searches for the optimal assignment in reduced problem space which can be costly. However, a branch-and-bound variant of LPCF can significantly reduce the search time thus reducing the overall computation time.

We also see from the results that eLPCF performs significantly better than both NOC-based solvers. However, eLPCF computes its deployment in over $\approx$ 1.5 to 2 times the total time achieved by LPCF. We attribute this increase in time to eLPCF computation flow also shown in Fig. 4. Essentially, eLPCF runs two LPCF algorithms in parallel and then combines their results via multi-objective optimization. The multi-objective optimization function combines the search spaces for both parent branches, both of which have varying completion time of its own. Waiting for both branches to complete their execution adds significant delay to overall run-time of eLPCF algorithm.

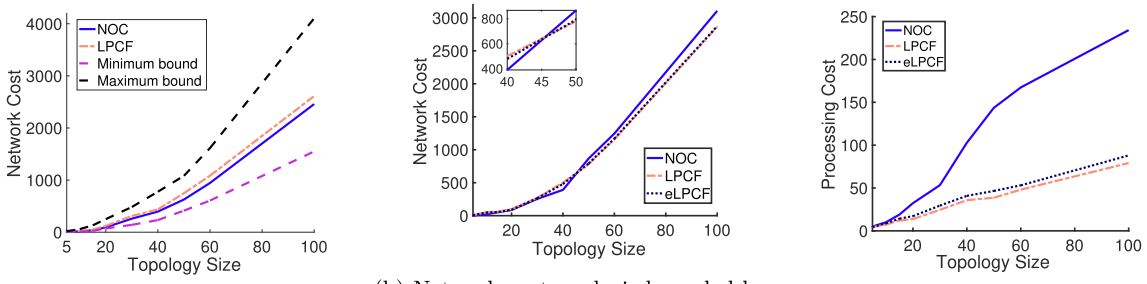
4.2 Comparative study of associated costs

In this section, we analyze and compare the performance of NOC, LPCF, and eLPCF algorithms for several attributes. It must be noted that we do not show the performance patterns of permutation-based NOC solver as the NOC-QAP solver consistently outperformed the former. Throughout the rest of the paper, we refer to NOC-QAP solver as NOC solver.

Network cost Figure 5a compares the network cost of task deployment for different topology sizes between LPCF and NOC QAP task assignment solver. The minimum and maximum bounds are obtained by choosing the $\mathcal{N}$ smallest and largest link costs in the Edge-Fog cloud resource graph respectively. It should be noted that the minimum/maximum cost depicted in the figure might not be a valid assignment as it does not consider job dependencies. From the figure, we observe that the task deployment computed by LPCF has an associated network cost within 10% range of the optimal value computed by the NOC. The performance of LPCF is highly impressive considering that, unlike NOC, the final deployment computed by LPCF is based upon the processing optimized assignment. Furthermore, as observed from Table 3, the NOC solver requires significantly larger time for calculating a deployment when compared to LPCF.

To level the playing field, we implement a branch-and-bound variant of NOC solver which approximates the best-possible solution within a pre-specified time limit. We then limit the computation time of NOC to that of LPCF and eLPCF and plot the associated network costs of the optimal assignments found by these algorithms in Fig. 5b. We observe that for large topologies, both LPCF and eLPCF achieves a lower associated network cost when compared to NOC solver. We further notice that both LPCF and eLPCF algorithms achieve quite similar performance in network cost as both algorithms optimize network cost in their search space.

Processing cost Figure 5c compares the associated processing cost of assignments computed by the three solvers. As



(a) Network cost analysis (b) Network cost analysis bounded by LPCF run time (c) Processing cost analysis

Fig. 5 Edge-Fog cloud network and processing cost analysis

unlike NOC, the assignment computed via LPCF is optimized for processing cost, the associated processing cost of the assignment computed by LPCF is always lower than that computed by NOC. Interestingly, for all topology sizes, eLPCF provides an assignment which has ≈10% higher processing cost than LPCF. This is because eLPCF optimizes its deployment on E_{proc} which favors Edge resources than Fog due to their lower e_p metric. As Edge resources have a lower processing power than Fog, the total processing cost involved is slightly higher in eLPCF deployment.

Energy cost Figure 6a compares the associated energy cost of assignment computed by the three solvers. As eLPCF optimizes the assignment on both processing and network cost, it achieves ≈10% reduction in E_{total} when compared to LPCF and NOC-based deployments for all topology sizes. Figure 6b shows the percentage reduction in energy cost of Edge and Fog resources in eLPCF deployment when compared to LPCF. We see an average of 20% energy reduction in Edge layer and around 8% in Fog. eLPCF provides efficient energy reduction as resources at the Edge are more energy constrained than Fog and require better energy management.

5 Discussion

In this section, we discuss several open questions pertaining to deployment and workings of Edge-Fog cloud in real environments.

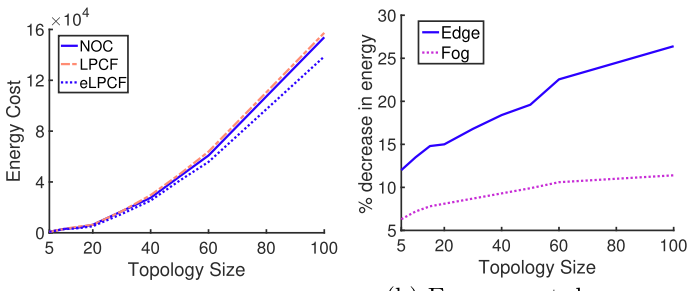
Q1. **Which node is responsible for running the assignment solver?**

Even though the proposed deployment algorithms require a centralized controller for managing their execution, however, the actual execution can be distributed and is not dependent on any single node. One node (resource in Edge-Fog cloud or an external broker) needs to be able to get the snapshot of the system state (availability of nodes and costs of links). We assume this snapshot to remain consistent throughout deployment calculation which justifies the low run-time requirement of the prescribed algorithm. For example in LPCF, calculating the individual assignment permutations for processing or networking costs in steps 1 and 3 can be distributed to other nodes or can be performed by the controller. The LPCF algorithm can thus be executed by any of the nodes in the system, whether an Edge node or a Fog node. We do not consider the cost of running the algorithm in our evaluation since the overheads are similar for all LPCF, eLPCF and NOC (namely obtaining the snapshot and iterating through the permutations).

Q2. **How does the ratio between Edge and Fog resources impact the associated cost of job deployment?**

We try to analyze the effects of Edge/Fog resource ratios on the overall cost of job deployment. The question is of utmost importance as several proposed Edge cloud models consider the architecture to be composed majorly of one type of computing resources. To find the optimal ratio between Edge and Fog resources, we vary their proportions between 20 to 80% out of 100 computing resources in the cloud. We further measure the impact of their ratios on processing, network, and energy cost of job deployment using the eLPCF algorithm. We ensure that the job network deployed on all the topologies remains consistent. The results of the analysis are shown in Fig. 7.

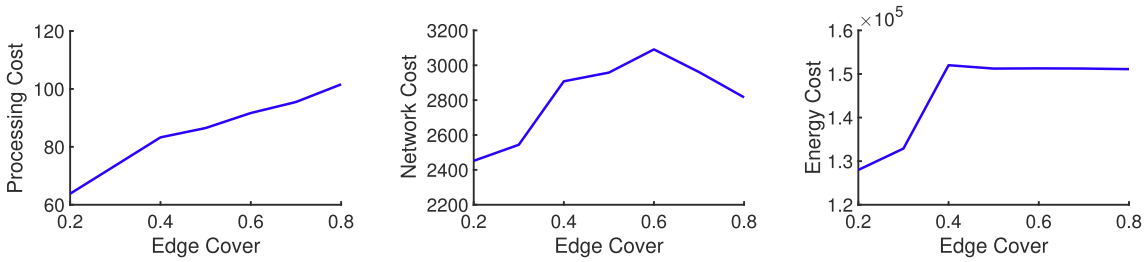
From Fig. 7a, we observe that as we increase the number of Edge resources in the cloud, the processing cost of the deployment increases linearly. We account this behavior to the lower processing power of Edge resources as compared to Fog. As we increase the number of lower-processing capable resources in the network, the deployed jobs require a longer time to complete. The effects are slightly different for associated network cost. In Fig. 7b we notice that the network cost initially grow with an increase in the number of Edge resources but after a certain *threshold* (around 60%), it starts to decrease. Up till the threshold, the overall network cost is impacted by the higher cost



(a) Energy cost analysis

(b) Energy cost decrease between Edge and Fog

Fig. 6 Edge-Fog cloud energy cost analysis



(a) Processing cost variation (b) Network cost variation (c) Energy cost variation

Fig. 7 Impact of Edge and Fog resource ratios on processing, network, and energy costs

inter-connections between Edge and Fog layers. Once past the threshold, the number of Edge resources far exceed that to Fog which replaces the inter-layer connections with lower-cost intra-Edge network connections, thus reducing the overall network cost.

In Fig. 7c, we observe that the energy utilization trend is somewhat similar to that of the network cost. As we increase the number of Edge resources, the associated energy used increases rapidly until it reaches a *plateau*. As explained in Section 3.3, the energy utilization is a composition of energy consumed in processing and networking. As we increase the number of Edge resources, both the energy consumed in processing by deploying jobs on lower powered resources and usage of inter-layer connections impacts the overall energy usage. After a certain number of Edge resources, the energy usage stabilizes as even though the energy due to processing increases, the energy consumed due to networking decreases due to the utilization of lower-cost network links.

Q3. **How well should the devices in the Edge-Fog cloud be connected to each other?**

As discussed in Section 2, the Edge layer in Edge-Fog has opportunistic, small-cost connections within itself whereas the Fog layer has dense network connections. The inter-layer connections between the Edge and the Fog are much higher cost and spans multiple hops. We try to find the optimal connection density of each layer (and inter-layer connections as well) such that the resulting assignment has low associated network cost. We increase the connection density of each layer from 20 to 80% and plot the changes in network cost of assignment computed by LPCF in Fig. 8.

As we increase the connection density of Edge, Fog and interconnections we see a decrease of ≈21, ≈9, and ≈17% in network cost, respectively. The inter-layer connections play a major part in resulting network cost; increasing their density impacts the overall cost much more. We can infer that deploying jobs in an Edge-Fog cloud which have well-connected devices in edge layer and dense connections between edge and fog layers, the overall cost of deployment is significantly reduced.

Q4. **Do the properties of job graph deployed on the Edge-Fog cloud also affect the overall cost?**

In Fig. 9, we change the interdependence of the job graph deployed on the Edge-Fog cloud from 10 to 100% and calculate the network cost associated with the deployment. We then deploy the job graph on several topology sizes of Edge-Fog cloud.

The results clearly show that higher dependence between the jobs result in a higher network cost. This is because the dependence links between the sub-jobs are mapped to the links between the devices of the Edge-Fog. Larger dependence links map to a mesh of device linkages thus

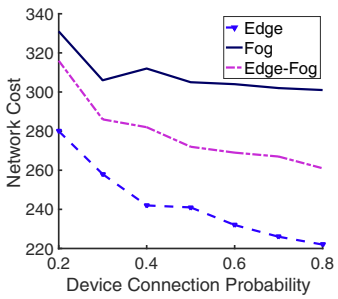


Fig. 8 Network cost variation with inter-device connection densities

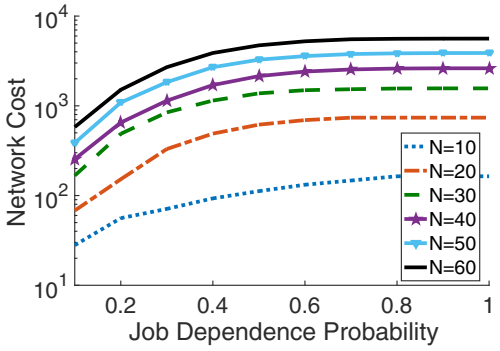


Fig. 9 Effect of job dependence on network cost of assignment

leading to an increased network cost. It can also be seen from the figure that after a particular job dependency value, the associated network cost of assignment stabilizes. This is primarily because after a particular job inter-dependence all heavy links of the device graph are part of the computed assignment and adding more links does not change the overall network cost significantly.

6 Related work

Several cloud abstractions have been proposed in the past which aim to bring the computation power near the data generators and consumers. Cloudlets [11, 12] propose a small-scale, localized cloud installed at the edge of the network along with the centralized cloud and is based on virtualization technologies. Several other works have explored combining stable peer-resources as nano data centers, micro clouds, community clouds, etc., for compute/storage tasks [28–31].

Several researchers built on the previous architectures and proposed installing a cloud abstraction of existing cloud model at the network edge. Following the Fog cloud characteristics proposed by CISCO [2], Bonomi et al. [32], and Yannuzzi et al. [3] show that the fog is the appropriate platform for loosely coupled, computationally intensive IoT-based applications, such as connected vehicles and smart cities. Hong et al. [4] provides a programming model and API for developing applications on the Fog cloud. On the other hand, unlike installing managed compute resources as Fog devices to process cloud applications, Lopez et al. [5] propose a semi-centralized cloud architecture, Edge cloud, composing of volunteer-based, user-centric compute resources. Likewise, Ryden et al. [33] proposed a dispersed cloud, Nebula, which utilizes volunteer resources for running data-intensive tasks. The authors discuss the effectiveness of their approach by deploying Map-reduce jobs on available resources. Our work differs from all discussed approaches as unlike other architectures, wherein a central entity schedules and processes several application tasks; Edge-Fog cloud proposes an entirely decentralized computing mechanism. Khan et al. [34] introduced federated, decentralized cloud models composed of community-driven computation resources but unlike proposed Edge-Fog cloud, they do not consider operator managed cloud resources to be part of the architecture.

Researchers have also proposed several deployment algorithms tackling Edge/Fog models proposed above. Most of the approaches utilize multi-objective optimization concepts to minimize network and deployment cost much similar to ours [35, 36]. Ahvar et al. [36] minimize the network cost of connectivity between data centers which are hosting the VMs involved in the computation. On the other hand, Silva et al. [35] model the VM deployment problem on Edge clouds as a generalization of the bin-packing problem and propose greedy heuristics to provide a near-optimal solution. However, unlike the approaches discussed above, our proposed LPCF and eLPCF algorithms re-model the deployment algorithms while optimizing one of several possible attributes. The algorithms cleverly avoid optimization of NP-hard problems and ensure a multi-objective optimized deployment in polynomial time.

Several researchers have proposed to integrate energy-aware task deployment mechanisms in edge networks. Authors have proposed placement algorithms which compute a task deployment maintaining low energy requirements [37, 38]. However, the works discussed above were developed while considering a semi-centralized edge cloud model which focuses towards maintaining energy efficiency in the central data center. Liu et al. [39] propose an energy efficient scheduler over distributed grid which incorporates deadline constraints of tasks. The authors discuss a scheduler design which places tasks over the grid while considering several costs. Mao et al. [40] develop a computation offloading strategy which considers execution cost and its effect on energy consumption of resources in Mobile Edge Clouds (MEC). However, to the best of our knowledge, eLPCF is first-of-its-kind optimization algorithm which optimizes all major attributes of an Edge resource, i.e., processing, transport, and energy usage.

7 Conclusion

In this paper, we proposed the Edge-Fog cloud, a decentralized cloud model for handling computation-based, high volume and distributable data such as that generated by IoT. The model builds on the existing Edge and Fog cloud approaches and provides data resilience through a centralized data store. We also provided a novel task allocation mechanism for Edge-Fog cloud which significantly reduces the deployment time without sacrificing the associated cost when compared to related approaches. We also incorporate the energy requirements of cloud resources to compute an allocation with least possible energy footprint. Further, we address several questions which might impact the real-world implementation of Edge-Fog cloud.

Funding information This research was funded by the joint EU FP7 Marie Curie Actions Cleansky Project, Contract No. 607584.

References

1. Broadband by the numbers, 'https://www.ncta.com/broadband-by-the-numbers'. Accessed: 22 Apr 2015

2. CISCO (2015) Cisco fog computing solutions: unleash the power of the internet of things (whitepaper). [Online]
3. Yannuzzi M et al (2014) Key ingredients in an iot recipe: fog computing, cloud computing, and more fog computing. In: IEEE CAMAD
4. Hong K et al (2013) Mobile fog: a programming model for large-scale applications on the internet of things. In: ACM SIGCOMM workshop on mobile cloud computing
5. Garcia Lopez P et al (2015) Edge-centric computing: vision and challenges. SIGCOMM Comput Commun Rev 45(5):37–42
6. Anderson DP et al (2002) Seti@home: an experiment in public-resource computing. Commun ACM 45(11):56–61
7. Beberg AL et al (2009) Folding@home: lessons from eight years of volunteer distributed computing. In: IEEE IPDPS
8. Chandra A, Weissman J, Heintz B (2013) Decentralized edge clouds. IEEE Internet Comput 17(5):70–73
9. Islam S, Grégoire J-C (2012) Giving users an edge: a flexible cloud model and its application for multimedia. Futur Gener Comput Syst 28(6):823–832
10. Hu YC, Patel M, Sabella D, Sprecher N, Young V (2015) Mobile edge computing—a key technology towards 5G. ETSI white paper
11. Verbelen T et al (2012) Cloudlets: bringing the cloud to the mobile user. In: ACM workshop on mobile cloud computing and services
12. Satyanarayanan M, Bahl P, Caceres R, Davies N (2009) The case for VM-based cloudlets in mobile computing. In: IEEE pervasive computing
13. Cordero IC, Orgerie A-C, Morin C (2015) GRaNADA: a network-aware and energy-efficient PaaS cloud architecture. In: IEEE international conference on green computing and communications (GreenCom)
14. Mohan N, Kangasharju J (2016) Edge-fog cloud: a distributed cloud for internet of things computations. In: Proceedings of CIoT. IEEE
15. BitCoin Mining Wiki, https://en.bitcoin.it/wiki/Mining
16. Roman R, Lopez J, Mambo M (2018) Mobile edge computing, Fog et al.: a survey and analysis of security threats and challenges. In: Future generation computer systems. Springer
17. CISCO (2015) Fog computing and the internet of things: extend the cloud to where the things are (whitepaper). [Online]
18. UbiSpark project, http://ubispark.cs.helsinki.fi/
19. Mohan N, Zhou P, Govindaraj K, Kangasharju J (2017) Managing data in computational edge clouds. In: Proceedings of the workshop on mobile edge communications (MECOMM '17). ACM
20. Xu J, Fortes JAB (2010) Multi-objective virtual machine placement in virtualized data center environments. In: Green computing and communications (GreenCom), 2010 IEEE/ACM international conference on cyber, physical and social computing (CPSCom), Hangzhou
21. Shakshuki E, Haubenwaller AM, Vandikas K (2015) Computations on the edge in the internet of things. Procedia Comput Sci 52:29–34
22. A quadratic assignment problem library, http://anjos.mgi.polymtl.ca/qaplib/. Accessed: 30 Sep 2010
23. Yurko MC (2010) A parallel computational framework for solving quadratic assignment problems exactly
24. Martello S, Minoux M, Ribeiro C, Laporte G (2011) Surveys in combinatorial optimization, vol 31. Elsevier, Amsterdam
25. List of power dissipation for CPUs, en.wikipedia.org/wiki/List_of_CPU_power_dissipation_figures
26. Edge-Fog simulator and LPCF solver, https://github.com/nitinder-mohan/EdgeFogSimulator
27. Platypus: multi-objective optimization in Python, http://platypus.readthedocs.io/en/latest/index.html
28. Kurniawan IP, Febiansyah H, Kwon JB (2014) Cost-effective content delivery networks using clouds and nano data centers. In: Ubiquitous information technologies and applications. Springer, pp 417–424
29. Shi C et al (2012) Serendipity: enabling remote computing among intermittently connected mobile devices. In: ACM MobiHoc
30. Li Y, Wang W (2014) Can mobile cloudlets support mobile applications? In: IEEE INFOCOM 2014 - IEEE conference on computer communications, pp 1060–1068
31. Mtibaa A, Harras KA, Fahim A (2013) Towards computational offloading in mobile device clouds. In: 2013 IEEE 5th international conference on cloud computing technology and science, vol 1, pp 331–338
32. Bonomi F et al (2014) Fog computing: a platform for internet of things and analytics. In: Big Data and Internet of Things: a roadmap for smart environments. Springer, pp 169–186
33. Anderson T et al (2014) A brief overview of the nebula future internet architecture. SIGCOMM Comput Commun Rev 44(3):81–86
34. Khan AM, Navarro L, Sharifi L, Veiga L (2013) Clouds of small things: provisioning infrastructure-as-a-service from within community networks. In: IEEE 9th international conference on wireless and mobile computing, networking and communications (WiMob), Lyon
35. Silva P, Perez C, Desprez F (2016) Efficient heuristics for placing large-scale distributed applications on multiple clouds. In: 16th IEEE/ACM international symposium on cluster, cloud, and grid computing, CCGrid
36. Ahvar E, Ahvar S, Crespi N, Garcia-Alfaro J, Mann ZÁ (2015) NACER: a network-aware cost-efficient resource allocation method for processing-intensive tasks in distributed clouds. In: IEEE 14th international symposium on network computing and applications, (NCA)
37. Gai K et al (2016) Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing. In: 2016 Elsevier Journal of Network and Computer Applications, vol 59
38. Gai K et al (2016) Energy-aware optimal task assignment for mobile heterogeneous embedded systems in cloud computing. In: 2016 IEEE 3rd international conference on cyber security and cloud computing (CSCloud)
39. Liu C, Qin X, Kulkarni S, Wang C, Li S, Manzanares A, Baskiyar S (2008) Distributed energy-efficient scheduling for data-intensive applications with deadline constraints on data grids. In: 2008 conference proceedings of the ieee international performance, computing, and communications conference
40. Mao Y, Zhang J, Letaief KB Dynamic computation offloading for mobile-edge computing with energy harvesting devices. IEEE J Sel Areas Commun