Annals of Telecommunications

Edge-Fog Cloud: A Distributed Cloud for Internet of Things Computations

--Manuscript Draft--

Manuscript Number:	ANTE-D-17-00139
Full Title:	Edge-Fog Cloud: A Distributed Cloud for Internet of Things Computations
Article Type:	CfP: Cloud Edge Computing in the IoT: Pave the way for an efficient data
Abstract:	Internet of Things based applications derive data from a significant number of smart sensors sensing information from the environment. As footprint of data can become extremely dense and geographically distributed, the applications utilize a cloud service for processing required computations. However, offloading data over a network induces significant network delay which severely impacts applications with sensitive run-time bounds. Several architectural abstractions, such as Fog and Edge computing, 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 energy efficient variant of LPCF, i.e. eLPCF algorithm, which incorporates an effective energy usage in Edge-Fog cloud task deployment. We evaluate both LPCF and eLPCF in a variety of scenarios and demonstrate its effectiveness in finding the processing task assignments.

Annals of Telecommunication manuscript No.
(will be inserted by the editor)

Edge-Fog Cloud: A Distributed Cloud for Internet of Things Computations

Nitinder Mohan · Jussi Kangasharju

the date of receipt and acceptance should be inserted later

Abstract Internet of Things based applications derive data from a significant number of smart sensors sensing information from the environment. As footprint of data can become extremely dense and geographically distributed, the applications utilize a cloud service for processing required computations. However, offloading data over a network induces significant network delay which severely impacts applications with sensitive runtime bounds. Several architectural abstractions, such as Fog and Edge computing, 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 energy efficient variant of LPCF, i.e. eLPCF algorithm, which incorporates an effective energy usage in Edge-Fog cloud task deployment. 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

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]. This leads to two major issues for computing IoT-generated data: i) the processing time of time-critical IoT applications is limited by the network delay for offloading data to a central cloud, and ii) uploading data from a large number of IoT generators may induce network congestion thus incurring further network delay.

To tackle network issues IoT-based cloud computation, researchers have proposed bringing the *compute cloud* closer to data generators and consumers. *Fog computing cloud* [2] proposes network devices such as routers etc. 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* [7] takes inspiration from projects such as SETI@Home, Folding@Home etc. [5,6], and proposes a consolidation of human-operated, voluntary resources such as desktop PCs, tablets, smart phones, 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].

Both cloud models, however, only propose to perform pre-processing tasks at the edge; thus relying on a centralized cloud for heavy, computationally intensive tasks. This semi-dependence on a central cloud works well for applications which require tight data and compute coupling but proves disadvantageous for applications which generate large amounts of distributed data interactive user involvement.

Nitinder Mohan · Jussi Kangasharju
Department of Computer Science
University of Helsinki, Finland
E-mail: {nitinder.mohan, jussi.kangasharju}@helsinki.fi

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 is composed of two layers of compute resources, namely Edge and Fog, and the core of the cloud is only responsible to store 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:

- We present Edge-Fog cloud architecture, which is based on classifying compute devices into Edge and Fog layers, depending on their capabilities and ownership.
- 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.
- 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.
- 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 Section 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.

[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 [23]. The paper 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 cloud 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.

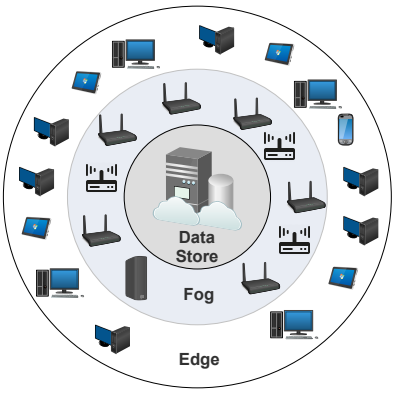


Fig. 1: Proposed Edge-Fog cloud architecture

2 Edge-Fog Cloud

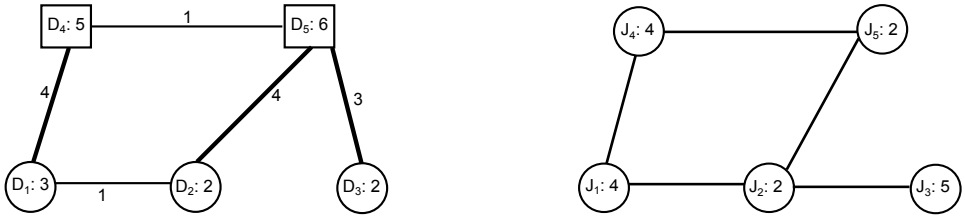
2.1 Architecture

Figure 1 shows the architecture of the Edge-Fog cloud. Unlike the network-oriented view of the traditional cloud model, the Edge-Fog cloud takes a node-oriented approach wherein the model is divided into three layers 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] and human-operated resources such as desktops, laptops, nano data centers, tablets, etc. As the name suggests, the resources reside at the edge of the network and are within one/two-hop distance from the IoT sensors and clients. Edge resources have varying ranges of computational capabilities from highly capable devices such as workstations, nano data centers etc. to less capable such as tablets or smart phones. Edge layer resources are assumed to have device-to-device connectivity within the layer and reliable connectivity to Fog layer.

Fog Layer. The Fog layer resides on top of the edge and 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 [10]). As Fog layer forms the network backbone of Edge-Fog cloud, the resources in this layer are interconnected with high-speed, reliable links. Moreover, Fog resources reside farther from the edge of the network when compared to Edge layer but closer than a

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



(a) Edge-Fog cloud resource graph

(b) Available jobs dependence graph

Fig. 2: Deployment example. Each job in Figure 2b needs to be deployed on a resource in Figure 2a.

central cloud. Fog is used to effectively handle computationally intensive tasks offloaded by Edge resources.

Data Store. Unlike the traditional cloud model, the core of the Edge-Fog cloud has no computational capabilities and only serves as a repository for archiving all data in the cloud. 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.

2.2 Benefits of the Edge-Fog cloud

The Edge-Fog cloud offer several benefits:

1. *Reduced network load:* The Edge-Fog cloud provides computation at the edge of the network near the IoT generators thus reducing the amount of data that flows in the network.
2. *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.
3. *Providing context:* Resources in Edge-Fog cloud also provide contextual awareness to data generated by sensors. Edge resources play a significant role in combining data from sensors using location or application contexts.
4. *No single point of failure:* As computation in Edge-Fog cloud is completely decentralized, the model has no single point of failure. Several snapshots of an application can be deployed on the cloud for increased reliability.

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

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. However, as the devices in the Edge-Fog cloud are governed by certain 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, the more coordination needed by task with its peers for completion, the higher will be the associated network cost. 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 2 shows a snapshot of Edge-Fog cloud of three Edge and two Fog resources. Edge and Fog is represented by circular and rectangular nodes respectively. The link weight denotes the distance/communication cost between two devices. The processing power of each device is listed along with its label. Figure 2b shows the job graph to be deployed on the Edge-Fog cloud. We assume only 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. The size of each job, listed along with its label, denotes the processing power required to complete the job.

We assume that the number of tasks and devices to be equal while task deployment. If there are more tasks than devices, we split existing devices into virtual devices such that their number is equal to the number of tasks. In the opposite case, we simply ignore the superfluous devices.

3.1 Network Only Cost (NOC)

Previous works have tried to model task deployment algorithms which minimize the associated networking

cost [11]. 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 $\mathcal{N}$x$\mathcal{N}$. $f(i)$ signifies the constraint of assigning a particular job to a device.

With $\mathcal{N}$ devices/jobs, the search space of possible assignments in NOC is $\mathcal{N}!$. For example, in Figure 2, 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) [12]. QAP generalizes minimal network cost assignment as:

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

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 [14]. 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 (LPCF)

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 LPCF, a task assignment solver which first minimizes processing cost of the assignment and further optimizes the network cost. In section 4 we show that LPCF algorithm is highly scalable when compared to NOC based algorithms. LPCF algorithm flow is depicted in figure 3 and is explained in detail below.

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

Table 1: Problem search space reduction in LPCF

3.2.1 Optimize the associated processing cost

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

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

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

Eq. 2 is an objective function of Linear Assignment Problem (LAP) which unlike QAP, is polynomial [15]. Algorithms such as Kuhn-Munkres/Hungarian guarantee an optimal solution for this problem in $O(n^3)$ (worst case). The first step of LPCF employs such an algorithm to compute an assignment which has the least associated processing cost.

3.2.2 Reducing the sub-problem space size

As the Edge-Fog cloud consists of several homogeneous devices with similar processing capabilities, interchanging jobs assigned on any such two devices does not alter the associated processing cost. The same argument is also applicable to homogeneous jobs in job graph. To illustrate, using equation 2 the assignment $D_1 \rightarrow J_1$; $D_2 \rightarrow J_2$; $D_3 \rightarrow J_3$; $D_4 \rightarrow J_4$; $D_5 \rightarrow J_5$ in Figure 2 has the processing cost of 5.97 which remains the same if we interchange the jobs deployed on D_1 and D_4.

LPCF computes all possible compositions of the assignment computed in the first step and forms a smaller search space of assignments with least associated processing cost. Table 1 shows the reduction in problem search space achieved by LPCF.

3.2.3 Accounting network cost of the assignment

In this step, LPCF computes the network cost associated with each assignment in the reduced problem search space and chooses the one with least network cost. Note 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

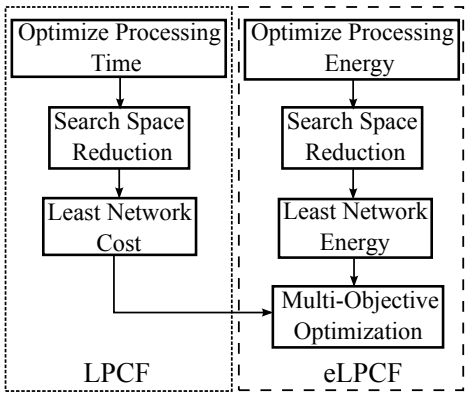


Fig. 3: LPCF and eLPCF algorithm

space sizes. Thus, the assignment computed by LPCF has the least associated processing cost and *almost* optimal network cost.

Our approach has several advantages over NOC-based algorithms. The most fundamental of them is that unlike the NOC assignment, our algorithm 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 (eLPCF)**

A major objective of edge cloud computing is to mitigate the high energy requirements imposed by large-scale data centers in current cloud architecture. Researchers have shown that due to smaller sizes and efficient power management algorithms, mobile device clouds can achieve similar computational powers while significantly reducing energy used per computation [24]. However, as most of the mobile devices are limited by their battery capacity and power availability, deployment algorithm must also consider energy requirements of a device while placing tasks.

Several researchers have proposed energy-aware task deployment mechanisms in edge networks which We propose an energy efficient LPCF (eLPCF) algorithm which computes a deployment with low energy requirements [27, 28]. However, all the above works are developed keeping in mind a semi-centralized edge cloud model with a focus towards maintaining energy efficiency in a central data center. Liu et al. [29] 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. However, the authors consider homogeneous resources over the network which does not map in Edge-Fog cloud environment.

In this section, we provide an energy-efficient variant of LPCF algorithm. eLPCF minimizes the total energy utilized per deployment in Edge-Fog cloud which is the sum of energy used for processing the task and for communicating over the network. i.e.

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

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 equation 1 and 2 respectively. e_n and e_p signify networking and processing cost-to-energy conversion metrics whose values will be dependent on resource's hardware. For example, e_n is highly dependent on energy used for sending a unit data via NIC and varies on wired or wireless networking. Similarly, e_p denotes power dissipation of a CPU 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 figure 3 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 equation 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 in similar fashion as first step of LPCF. eLPCF does not consider minimization of E_{netw} in 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 the sub-problem space size

eLPCF builds on the assumption of Edge-Fog cloud consisting of several homogeneous jobs and devices with

similar hardware specifications to build its reduced sub-problem search space. 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 formation of several deployments with varied network dependencies but maintains the lowest E_{proc} as computed in 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 equation 5. The algorithm then does an iterative exhaustive search of search 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, the resulting deployment can have induced high processing and network cost which severely affects run-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 [13]). 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.

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 2: Default parameter values of Edge-Fog cloud simulator

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 [12]. We use Platypus [26] to implement the multi-objective optimization function in Python.

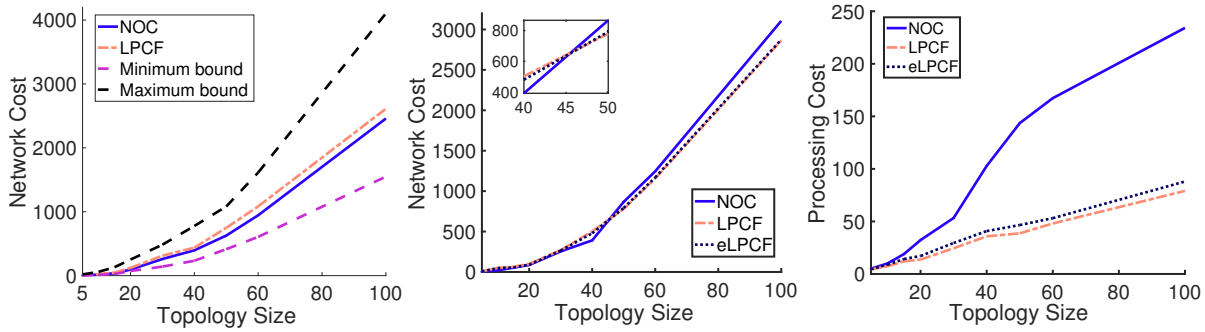
4.1 **Completion time analysis**

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 ˜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 ˜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.

Topology Size =	5	10	15	20	30	40	50	60	100	150
NOC Permutation solver	0.068s	23m 20s	>1h	>1h	>1h	>1h	>1h	>1h	>1h	NA
NOC QAP solver	0.026s	36.273s	3m 22s	18m 38s	>1h	>1h	>1h	>1h	>1h	NA
LPCF	0.0005s	0.002s	0.044s	0.073s	1.9s	13.229s	51.686s	2m36s	12m24s	>1h
eLPCF	0.0011s	0.0116s	0.08s	0.126s	3.28s	22.061s	1m14s	3m52s	24m43s	>1h

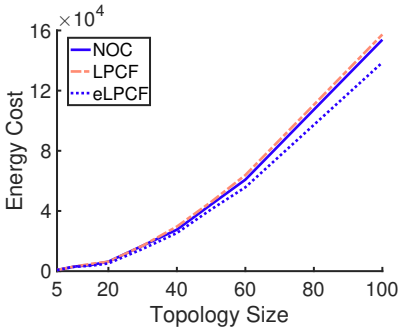
Table 3: Optimal assignment computation time



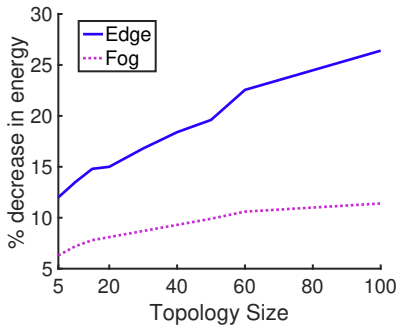
(a) Network cost analysis (b) Bounded network cost analysis (c) Processing cost analysis

Fig. 4: Edge-Fog cloud network and processing cost analysis

We also see from the results that eLPCF performs significantly better than both NOC-based solvers. When compared to LPCF algorithm, eLPCF takes about ˜1.5-2 times the total time. The major reason is because eLPCF essentially 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 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.



(a) Energy cost analysis



(b) Percentage energy cost decrease between eLPCF and LPCF

Fig. 5: Edge-Fog cloud energy cost analysis

4.2 **Comparative study of associated costs**

Network Cost: Figure 4a compares the cost minimization achieved by LPCF when compared to NOC QAP task assignment solver. The minimum/maximum bounds are obtained by choosing the $\mathcal{N}$ smallest/largest link costs in the Edge-Fog cloud resource graph. 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.It is evident from the figure that even though the assignment computed via both LPCF first optimizes processing time for assignment, the associated network cost is within 10% range of the optimal value computed by the NOC. Also, we can see from Table 2 that the QAP-based NOC solver has significantly higher computing time when compared to LPCF.

We further implemented a branch-and-bound variant of QAP solver which approximates the best solution within the specified time limit. We then limit the computation time of QAP to that of LPCF and eLPCF and

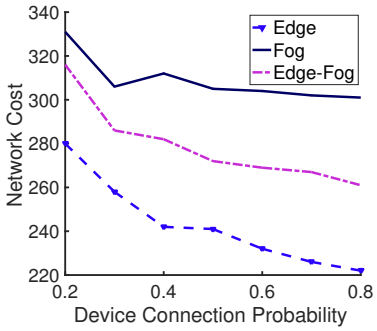


Fig. 6: Network cost variation with inter-device connection densities

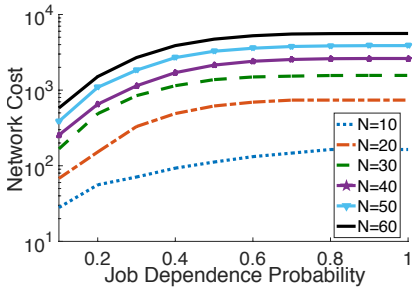


Fig. 7: Effect of job dependence on network cost of assignment

plot the associated network costs of the optimal assignments found by these algorithms. The plot is shown in Figure 4b. Here we see that for large topologies, the assignment computed by both LPCF and eLPCF has lower associated network cost than that computed by NOC. Further, the network cost of computed assignment by LPCF lies extremely close to eLPCF as both algorithms optimize network cost in their search space.

Processing Cost: Figure 4c compares the associated processing cost of assignments computed by the three solvers. As unlike NOC, the assignment computed via LPCF is optimized on 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 favours 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 5a 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 5b 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

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

The LPCF algorithm needs a centralized controller for managing the execution of the algorithm, however, its actual execution can be distributed and is not dependent on any single node. One node needs to be able to get the snapshot of the system state (availability of nodes and costs of links) and we assume this snapshot to remain constant during the execution of the algorithm. 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 both LPCF and NOC QAP (namely obtaining the snapshot and iterating through the permutations).

Q2. How well should the devices in the Edge-Fog cloud be connected to each other?

Edge layer has optimistic connections within itself whereas the Fog layer has dense network connections; but 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 Figure 6.

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.

Q3. Do the properties of job graph deployed on the Edge-Fog cloud also affect the overall cost?

In Figure 7, 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 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

Cloudlets [16] 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 [17–20].

Several researchers have proposed to bring part of the cloud closer to the edge of the network. Following the Fog cloud characteristics proposed by CISCO [2], Bonomi et al. [21] 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] provide 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. [7] propose a semi-centralized cloud architecture, Edge cloud, composing of volunteer-based, user-centric compute resources. Likewise, Ryden et al. [22] 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 these approaches as unlike them, wherein a central entity schedules and processes several application tasks; Edge-Fog cloud proposes an entirely decentralized computing mechanism. Due to its unique nodular and layered architecture, the Edge-Fog cloud natively supports computations on distributed, semi-dependent data produced by IoT.

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.

8 Acknowledgement

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: 2015-04-22.
2. CISCO, "Cisco fog computing solutions: Unleash the power of the internet of things (whitepaper)," 2015. [Online].
3. M. Yannuzzi et al. "Key ingredients in an iot recipe: Fog computing, cloud computing, and more fog computing," in *IEEE CAMAD*, 2014.
4. K. Hong et al. "Mobile fog: A programming model for large-scale applications on the internet of things," in *ACM SIGCOMM Workshop on Mobile Cloud Computing*,2013.
5. D. P. Anderson et al. "Seti@home: An experiment in public-resource computing," *Commun. ACM*, vol. 45, no. 11, pp. 56–61, Nov. 2002.
6. A. L. Beberg et al. "Folding@home: Lessons from eight years of volunteer distributed computing," in *IEEE IPDPS*, 2009.
7. P. Garcia Lopez et al. "Edge-centric computing: Vision and challenges," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, pp. 37–42, Sep. 2015.
8. A. Chandra, J. Weissman, and B. Heintz, "Decentralized edge clouds," *Internet Computing, IEEE*, vol. 17, no. 5, pp. 70–73, Sept 2013.
9. S. Islam and J.-C. GrÃĪgoire, "Giving users an edge: A flexible cloud model and its application for multimedia," *Future Generation Computer Systems*, vol. 28, no. 6, pp. 823 – 832, 2012.

10. CISCO, "Fog computing and the internet of things: Extend the cloud to where the things are (whitepaper)," 2015. [Online].
11. E. Shakshuki, A. M. Haubenwaller, and K. Vandikas, "Computations on the edge in the internet of things," *Procedia Computer Science*, vol. 52, pp. 29 – 34, 2015.
12. A quadratic assignment problem library, http://anjos.mgi.polymtl.ca/qaplib/, accessed: 2010-09-30.
13. Edge-Fog simulator and LPCF solver, https://github.com/nitinder-mohan/EdgeFogSimulator.
14. M. C. Yurko, "A parallel computational framework for solving quadratic assignment problems exactly," 2010.
15. S. Martello, M. Minoux, C. Ribeiro, and G. Laporte, *Surveys in combinatorial optimization*. Elsevier, 2011, vol. 31.
16. T. Verbelen et al. "Cloudlets: Bringing the cloud to the mobile user," in *ACM Workshop on Mobile Cloud Computing and Services*, 2012.
17. I. P. Kurniawan, H. Febiansyah, and J. B. Kwon, "Cost-effective content delivery networks using clouds and nano data centers," in *Ubiquitous Information Technologies and Applications*. Springer, 2014, pp. 417–424.
18. C. Shi et al. "Serendipity: Enabling remote computing among intermittently connected mobile devices," in *ACM MobiHoc*, 2012.
19. Y. Li and W. Wang, "Can mobile cloudlets support mobile applications?" in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, April 2014, pp. 1060–1068.
20. A. Mtibaa, K. A. Harras, and A. Fahim, "Towards computational offloading in mobile device clouds," in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, vol. 1, Dec 2013, pp. 331–338.
21. F. Bonomi et al."Fog computing: A platform for internet of things and analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments*. Springer, 2014, pp. 169–186.
22. T. Anderson et al. "A brief overview of the nebula future internet architecture," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 81–86, Jul. 2014.
23. Mohan, N., and Kangasharju, J. Edge-fog cloud: A distributed cloud for internet of things computations. In *Proc. CIoT* (November 2016), IEEE.
24. UbiSpark project, http://ubispark.cs.helsinki.fi/.
25. List of power dissipation for CPUs, en.wikipedia.org/wiki/List_of_CPU_power_dissipation_figures.
26. Platypus-Multiobjective Optimization in Python, http://platypus.readthedocs.io/en/latest/index.html
27. K. Gai et al., "Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing," in *2016 Elsevier Journal of Network and Computer Applications*, Volume 59.
28. K. Gai et al., "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)*.
29. Liu, Cong and Qin, Xiao and Kulkarni, Santosh and Wang, Chengjun and Li, Shuang and Manzanares, Adam and Baskiyar, Sanjeev, "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*.