epcAware: A Game-based, Energy, Performance and Cost Efficient Resource Management Technique for Multi-access Edge Clouds

Muhammad Zakarya, Hashim Ali*, Lee Gillam, Khaled Salah, Izaz Ur Rahman, Ayaz Ali Khan, Rahim Khan, Omer Rana, Rajkumar Buyya

Abstract—The Internet of Things (IoT) is producing an extraordinary volume of data daily, and it is possible that the data may become useless while on its way to the cloud for analysis, due to longer distances and delays. Fog/edge computing is a new model for analyzing and acting on time-sensitive data (real-time applications) at the network edge, adjacent to where it is produced. The model sends only selected data to the cloud for analysis and long-term storage. Furthermore, cloud services provided by large companies such as Google, can also be localized to minimize the response time and increase service agility. This could be accomplished through deploying small-scale datacenters (reffered to by name as cloudlets) where essential, closer to customers (IoT devices) and connected to a centrealised cloud through networks - which form a multi-access edge cloud (MEC). The MEC setup involves three different parties, i.e. service providers (IaaS), application providers (SaaS), network providers (NaaS); which might have different goals, therefore, making resource management a defficult job. In the literature, various resource management techniques have been suggested in the context of what kind of services should they host and how the available resources should be allocated to customers' applications, particularly, if mobility is involved. However, the existing literature considers the resource management problem with respect to a single party. In this paper, we assume resource management with respect to all three parties i.e. IaaS, SaaS, NaaS; and suggest a game theoritic resource management technique that minimises infrastructure energy consumption and costs while ensuring applications performance. Our empirical evaluation, using real workload traces from Google's cluster, suggests that our approach could reduce up to 11.95% energy consumption, and approximately 17.86% user costs with negligible loss in performance. Moreover, IaaS can reduce up to 20.27% energy bills and NaaS can increase their costs savings up to 18.52% as compared to other methods.

Index Terms—resource management, internet of things, mobile edge clouds, energy efficiency, performance, game theory

1 Introduction

Real-time applications such as on-line gaming and video conferencing have on-demand requirements to provide high-quality results within the agreed time e.g. shorter response time through communication with the closest application server. Using cloud platform to deploy real-time applications offers several benefits including reduced OpEx (operational costs), but not necessarily, and on-demand resource allocation - assign resources per needs of the application. However, real-time applications may be sensitive to the quality of network e.g. latency between users and services. Therefore, the real-time applications' requirements could, possibly, be addressed through combining the emerging edge computing technology with MEC and fog - which allows computations to be accomplished at the edge of the network. The rationale of commissioning this technology is to allocate services or run applications within the proximity of customers and closer to where computational results are desirable. This can be achieved through deploying small-scale or micro datacenters closer to customers, and connected to regional cloud datacenters. Note that the management systems to run and practice such infrastructures are, largely, still missing, with the notable exception of AWS outposts[1] and revised OpenStack [1].

Furthermore, with the MEC or fog framework, there are certain questions that still needs to be investigated. For example; (i) where these small datacenters (cloudlets should be deployed; (ii) which services should be installed; (iii) where and how the resources should be allocated to users' applications; (iv) how user mobility (service migration) should be handled; and (v) how the aforementioned MEC framework should be optimized to minimize or maximize various objectives such as users' monetary costs, energy consumption and workload performance in terms of latency, execution time and etc. Albeit, (iii) to (v) can be seen, largely, similar to traditional clouds; but, the management policies should be redesigned for fog infrastructure.

The aim of this research is to examine resource allocation/placement and consolidation challenges associated with edge (fog) computing platforms. The main questions that this research will answer include: (i) where small datacenters (cloudlets – at the edge of networks) should be installed to meet users' demand while abating the infras-

• *M. Zakarya, H. Ali, I.U. Rahman, A.A. Khan, and R. Khan are with the Department of Computer Science, Abdul Wali Khan University, Pakistan. L. Gillam is with the University of Surrey, UK. K. Salah is with the Khalifa University, UAE. O. Rana is with the University of Cardiff, UK. R. Buyya is with the School of Computing and Information Systems, University of Milbourne, Australia.*
Athors Correspondence: (denotes first equal author)*
{mohd.zakarya, hashimali, izaz, ayazali, rahimkhan}@awkum.edu.pk
l.gillam@surrey.ac.uk, khaled.salah@ku.ac.ae, ranaof@cardiff.ac.uk, rbuyya@unimelb.edu.au

1. https://aws.amazon.com/outposts/

tructure global cost (CapEx - capital expenditures + OpEx); (ii) once the platform has been installed, what, how and where should global control services be deployed; and (iii) at what size/scale. Furthermore, this will open opportunities for advising a generic resource allocation/placement and management/migration framework for various kind of fog/edge services.

While the model of cloud computing provided by a few mega/large providers (Google and Amazon AWS) still widely used, the beginning of innovative and emerging technologies such as IoT applications, edge computing and MECs is challenging this approach [1]. To cope with this technological change, cloud and network communities are now working in the direction of large-scale distributed, but small sized, datacenter infrastructures (known as cloudlets) that are installed at the edge of the network - closer to users and their devices i.e. fog infrastructure (hence, distributed shape applications that consist of various modules) [2], [3]. The fog, edge and cloud paradigm is attracting rising interest as it also improves services' agility and performance in terms of response time. For example, IoT applications can take benefits from edge nodes' deployment to perform real-time analysis while conserving main datacenters for in-depth data analytics [4]. This can be seen as a mixture of fog/edge/cloudlet/MEC (the latter now being multi-access rather than merely mobile) – where MEC suggests being within the radio access network (RAN) but fog/edge could relate consumer devices aggregating data from sensors (before passing to the remote cloud). Further, "cloudlet" as a mini datacenter, presumably consistent with a large cloud provider's provisions, would be able to support such aggregation but would always be further from one or more such sensors. Yet, such a cloudlet need have no relationship to the RAN.

In addition to recognizing where edge clouds should be deployed or installed, the drivers of such an evolution lie in the design of suitable management systems that will permit: (i) an operator i.e. cloud, network, or edge to aggregate, supervise and expose such massively distributed resources; and (ii) to implement new kinds of services that may be deployed and managed by the operator or by users. However, designing such a management system is challenging because fog/edge infrastructures differ from traditional clouds regarding heterogeneity, dynamicity, the possible huge distribution of resources, and economics of scale - if smaller, heterogeneity is less likely. The objective of this research is to explore the placement-related questions of a massively distributed edge cloud infrastructure. The research will be organised around the subsequent activities: (i) propose placement algorithms that can satisfy QoS expectations while optimizing different objectives such as infrastructure cost minimization, energy requirements and reliability (performance in terms of response time and QoS); and (ii) evaluate proposed algorithms through simulations by leveraging the iFogSim tool-kit [5] and Google cluster or similar datasets [6].

The rest of the paper is organized as follows. Major contributions of the work presented in this paper are stated in Section 2. In Sec. 3, we discuss MECs and resource management. In Section 4, we model the MECs resource allocation problem as a game. A game theoretic solution is presented, is Section 5, to solve the problem. We validate the proposed approach using real workload traces from Google cluster in Section 6. We offer an overview of the related work in Section 7. Finally, Section 8 concludes the paper with several future research directions.

2 Contributions

Following are the major contributions of the research conducted in this paper:

- we model the MECs resource allocation and service migration problem using a game;
- we propose resource allocation and migration algorithms using the game theory; and
- we evaluate the performance of the proposed algorithms using real workload datasets from Google's cloud.

3 Background

MECs offer finite resources at the edge of the network; making it possible to run user's application in its close proximity, as shown in Fig. 1. The resource at the edge are provisioned at the cloudlet or MEC server. Moreover, the application, in cloudlet, runs at one (or more than one) hope communication distance unlike to their native execution either in the mobile/fog device or core internet (remote cloud) - zero and two+ hop distance, respectively. Therefore, application latency might be potentially affected depending on the network services. The finite number of cloudlets' resources also put questions on their efficient allocation to connected users. Although, there are several proposals to share the resources of several cloudlets in a particular geographic area [7]. However, this will be a challenging problem when the cloudlet resources are offered by different service providers having different goals and objectives. Similarly, beside providers, mobile users, who run the applications, are also usually selfish and competitive; each user wants to optimise his/her own pay-off or application's performance [8].

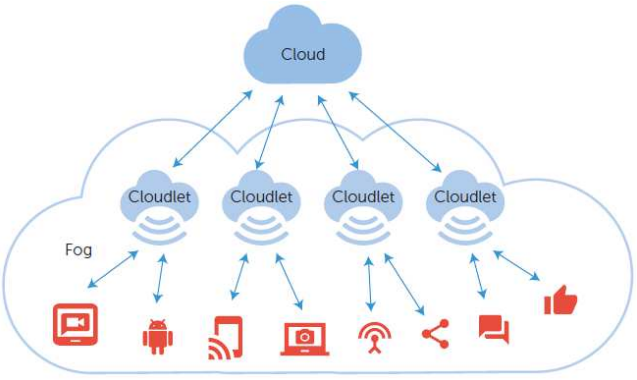

Fig. 1: Cloud, cloudlet and fog architecture [9]

Moreover, different resources can be offered at different costs, energy consumption and performance levels. Resource placement can significantly affect service providers (IaaS, network) and customers (SaaS) economics [10]. For example, reduced performance of applications increase users'

costs as well as energy consumption. Therefore, it is essential to provision appropriate resources in order to meet application QoS requirements and providers objectives. Moreover, if mobility is involved - users are moving or application modules are explicitly migrated among hosts for energy efficiency or performance gains, then resource management complexities will potentially increase. Appropriate resource management techniques are, therefore, essential to cope with various objectives. In this paper, we elaborate: (i) how the infrastructure and available resources (IaaS, SaaS, NaaS) should be managed in order to increase service agility, performance in terms of response time and minimize the energy related costs; (ii) investigate how users' workload should be run and how, where resources or services should be allocated, provisioned to it; (iii) develop algorithms and mathematical models to solve the resource allocation (network and computing) problem, efficiently, for MEC to support emerging mobile applications; and (iv) realize and implement the models and algorithms into a software to demonstrate their feasibility and practicability.

4 Problem Description

Management of resources in MECs is very challenging, because offering quality services to the end-users depends on various players, with moderately conflicting goals, such as infrastructure owners (IaaS), network operators (network as a service - NaaS), and application providers (SaaS), where each player controls a particular part of the whole system. Integral to the problem is the facts that both communication and computation capacity is needed to guarantee high QoS in terms of low response time and high throughput. Since, each player may have different objectives to optimise where the objectives of one player may potentially affect the objectives of another player and vice versa. Existing works [1], [7], [10], [11] either assume that the whole infrastructure is managed by a single player, largely, the resource providers [10], or separate the management of the network, application resources from the core edge computing capacity [7]. We believe, resource allocation in MECs should be assumed as a multi-objective optimisation problem in such a way that players' competition for their objectives can be optimised. The objectives of all parties are somehow aligned – insufficient provisions for the application would ruin revenues for all concerned (customers go elsewhere, and so do the providers), and so there is an incentive to only minimize to the point at which this is avoided and so either all 'win' together (can minimize absent impact) or all 'lose' together (minimizing has impact). Moreover, simple optimization methods cannot ensure a win-win situation for all players, in similar resource allocation scenarios [12], [13]. Therefore, we use game theory to model and solve such a complex, multi-player resource allocation problem.

Largely, SaaS providers host their applications on virtualised resources provided by an IaaS provider. Moreover, SaaS providers need to comply with every application's quality of service (QoS) requirements, as described in Service Level Agreement (SLA) with the customers, which determine the SaaS revenue on the basis of achieved level of performance. However, application performance is not only dependent on computational resources (provided by IaaS), but, as well as, on the network bandwidth provided by the network operators. Similarly, service providers would prioritise their workloads based on the nature of applications (native or third party) [7]. Furthermore, in MECs, the network operators could be: (a) the IaaS owners (internal network); and (b) third party mobile network operators (external network). Therefore, network resources from various providers should be provisioned at affordable prices. The focus of SaaS providers would be to maximize their revenues through minimising SLAs, while reducing the total cost of using compute, as well as, network resources provided by the IaaS and third party network service providers. Moreover, in case of multiple IaaS providers or cities (MECs), SaaS providers would probably compete and bid for the use of infrastructural resources based on prices. On the other hand, the IaaS and third party network service providers could maximize their revenues through providing their virtualized resources as much as possible. Moreover, IaaS providers could maximize their resource usage (utilisation) in order to minimise energy consumption.

TABLE 1: Notations used in problem formulation

Notation	Description
N	List of players so that $m \in N$
K	List of available resources
C_K^m	Type $k \in K$ resources offered by $m \in N$ provider
C	List of total offered resources by $m \in N$
M	Set of applications
$\mathcal{H}$	List of hosts in datacenter such that $h \in \mathcal{H}$
L	List of users such that $u \in U$
$\mathcal{M}$	Number of cloudlets such that $e \in \mathcal{M}$
j	A particular job that belongs to an application $\in M$
R	Resource request matrix
A	Resource allocation matrix
D	Allocation decision
U	Utility function of all providers
X_{iN}	Resource - provider mapping function, constraint
x_{ij}	VM - host mapping function, constraint
b_{cost}	Cost of network bandwidth B
T_{rate}	Rate of transmission over the B
bid_j	Bid of j^{th} provider for its resources
e_{ij}	Energy consumed at host j for VM i
t_k	Execution time of application k
X_{ij}	Mapping function of provider i to application j
p_{ij}^{IaaS}	Utility of IaaS for application j on host i
p_{ij}^{SaaS}	Utility of SaaS for application j of user i
p_{ij}^{NaaS}	Utility of NaaS for application j for i^{th} channel
E_m	Energy consumed during a VM migration
VM_{data}	The data copied during the migration of VM

4.1 Mathematical Formulation

In our game, we assume three players with conflicting goals: (i) IaaS – whose aim is to minimise energy consumption through consolidating the workload onto the fewest resources; (ii) SaaS – who want to maximise service performance and avoid SLAs (and, therefore, increase revenue); and (iii) third party network service providers and operators – whose aim is to minimise network traffic in order to ensure QoS in terms of performance (response time). We assume that $N = \{1, 2, ..., N\}$ denotes a set of all service providers that act as players. A list of all mathematical notations can be found in Table 1. Moreover, each player has a set of $K = \{1, 2, ..., K\}$ various kinds of resources including

computation (IaaS), application (SaaS), and communication (NaaS) resources. The m^{th} service provider denotes its offered resources as $C^m = \{C_1^m, C_2^m, ..., C_K^m\}$ where C_k^m is the amount of resources of type k offered by m service provider. Therefore, all offered resources at various service providers is given by:

$$C = \Big\{ \sum_{m \in N} .C_1^m, \sum_{m \in N} .C_2^m, ..., \sum_{m \in N} .C_K^m \Big\} \tag{1}$$

Moreover, every job j (i.e. application module) asks for a set of resources (in the form of coalition) that belongs to a set of all applications given by $M = \{M_1 U M_2 ... U M_K\}$ subject to the condition that every application module is allocated resources once in every $m \in N$ - an application module can run exactly once. We assume that every application module j runs in a virtual machine (VM) or container. Furthermore, it is possible that a job may comprise a multiplicity of amounts of containers/VMs/network resources which presumably can be co-located - unless there can be only one VM or container per host or application, in which case this readily simplifies. However, to simplify our formulation, we assume that each module requires one VM or container, at most and the total number of VMs or containers provisioned cannot exceed the application M resource requirements R (subject to constraint in Eq. 2).

$$\sum_{i=1}^{VMs|containers} M_i \geq R_M \tag{2}$$

Suppose an MEC system with a single cluster (cloud datacenter), several edge locations (cloudlets) and numerous mobile/fog devices. These resources are interconnected through networks such that cloudlets are in close proxity to fog devices. An application's modules are distributed and run over these resources. The cloud datacenter which consists of $\mathcal{H}$ heterogeneous hosts and each host is denoted by h, such that $1 \leq h \leq \mathcal{H}$. For $k \in K$ resources (such as CPU, memory, storage, network) each h can be denoted as a capacity vector $\mathcal{C}^h = \{c_1^h, c_2^h, c_3^h, ..., c_k^h\}$; and each kind of resource is denoted as n. For example, $h(2, 4, 10, 1)$ describes that a particular host h has 2 CPUs, 4 GB memory, 10 GB disk storage and 1 Gbps network card. Moreover, we assume that there are $\mathcal{M}$ edge locations (cloudlets) and each edge cloud e consists of several heterogeneous hosts S; and each edge host $s \in S$ resources are also represented as capacity vector $\mathcal{C}^s$ – similar to cluster host $\mathcal{C}^h$. Moreover, cloudlet resources are extremely lower than datacenter resources i.e $\sum S << \sum \mathcal{H}$ and $C^s << C^h$. The resources in datacenter and cloudlets are virtualised, therefore, offered through VMs. Each VM can run a particular application module or job. The application or job submitted by a particular user u is denoted as J_u, where $u \in \{1, 2, ..., U\}$. Every application comprises several modules that run concurrently [9]. Furthermore, a variety of VM or container types are predefined by each cloud provider (cloud and cloudlets); and each type's resources are encoded by the capacity vector $\mathcal{R}$ such that $R_u = \{r_{u1}, r_{u2}, ..., r_{uj}, ..., r_U\}$. Note that, each VM or container can run a single job (application module) at a time (subject to constraints in Eq. 12), both in the remote datacenter and cloudlets. Various resources like CPU, memory, storage and network of a host $h \in \mathcal{H}$ or $s \in S$ will be occupied, only, when a particular VM or container is created on $h \in \mathcal{H}$ or $s \in S$. Mobility of application modules is, therefore, possible through VM migration [10].

With the above terms, resource requests for a particular job j (or user) can be defined as a $u \times v$ dimensional matrix ($\mathcal{R}^j$); where rows represent the VM or container type and columns denote the amount of various resources associated with the VM/container type:

$$R^j = \begin{bmatrix} r_1^j \\ r_2^j \\ ... \\ r_u^j \end{bmatrix} = \begin{bmatrix} r_{11}^j r_{12}^j r_{13}^j ... r_{1v}^j \\ r_{21}^j r_{22}^j r_{23}^j ... r_{2v}^j \\ ... \ ... \ ... \ ... \ ... \\ r_{u1}^j r_{u2}^j r_{u3}^j ... r_{uv}^j \end{bmatrix} \tag{3}$$

Note that, the request matrix R is an augmentation of all the request matrices (from all the service providers) as given below.

$$R = \begin{bmatrix} r_1 \\ r_2 \\ ... \\ r_U \end{bmatrix} = \begin{bmatrix} r_{11} r_{12} r_{13} ... r_{1v} \\ r_{21} r_{22} r_{23} ... r_{2v} \\ ... \ ... \ ... \ ... \ ... \\ r_{u1} r_{u2} r_{u3} ... r_{UV}^j \end{bmatrix} \tag{4}$$

We assume that a particular job can be allocated to at most one host; and various resources mean CPU, RAM, storage and network. Moreover, for a particular host h, a possible resource allocation state can be described as an allocation matrix $\mathcal{A}^h$:

$$A^h = \begin{bmatrix} a_1^h \\ a_2^h \\ ... \\ a_u^h \end{bmatrix} = \begin{bmatrix} a_{11}^h a_{12}^h a_{13}^h ... a_{1v}^h \\ a_{21}^h a_{22}^h a_{23}^h ... a_{2v}^h \\ ... \ ... \ ... \ ... \ ... \\ a_{u1}^h a_{u2}^h a_{u3}^h ... a_{uv}^h \end{bmatrix} \tag{5}$$

where a_{ab}^h represents the amount of resources b on a particular host h allocated to a container or VM a. Similar to the request matrix R, the allocation metric A is an augmentation of all the allocation matrices (from all the service providers) as given below.

$$A = \begin{bmatrix} a_1 \\ a_2 \\ ... \\ a_u \end{bmatrix} = \begin{bmatrix} a_{11} a_{12} a_{13} ... a_{1v} \\ a_{21} a_{22} a_{23} ... a_{2v} \\ ... \ ... \ ... \ ... \ ... \\ a_{u1} a_{u2} a_{u3} ... a_{uv} \end{bmatrix} \tag{6}$$

For every host, an allocation decision $\mathcal{D}$ is a possible allocation status from a set of all possibilities based on the resource requirement matrix $\mathcal{R}$:

$$D = \Big[A^1, A^2, A^3, ..., A^h, ..., A^p\Big] \tag{7}$$

The aim of the resource allocation problem, given the resource requirement matrix $\mathcal{R}$ and the capacity sets of hosts $\mathcal{C}$, is to calculate a reasonable mapping from resources to user's jobs. In our game, various players (resource, application and network providers) collectively arrive at a single decision that describes VM allocations which are collectively best for the whole MEC system; and also ensuring that the allocations are both energy and performance (runtime) optimized.

If we assume the above problem as a single objective optimization problem, then each service provider, individually, wants to maximize its utility through allocating its available resources such that: (i) energy consumption and workload

performance (runtimes) are minimised (IaaS); (ii) applications' runtimes are minimised (SaaS); and (iii) network traffic is minimised (NaaS). From a single objective optimisation problem of a single service provider, the objective of all providers is given by:

$$max_A\Big(\sum_{j\in N} u_j^n.A_{i\in M}^n\Big) \tag{8}$$

Moreover, for each VM or application i its allocation matrix to each service provider $N \in \{IaaS, SaaS, NaaS\}$ is given by $X_{iN} = \{0,1\}$ such that $\sum X_{i,N} \leq 1$ i.e. each application is allocated exactly once to every provider. Once allocated, each user pay p (utility of server providers) for its application i to each service provider $n \in N$; where p is the sum of p_{in}^{IaaS}, p_{in}^{SaaS}, and p_{in}^{NaaS} that represent the cost own by IaaS, SaaS, and NaaS, respectively. Therefore, the utility of the whole MEC system from a particular user with application i is given by $U_i = X_{ij}(p_{ij}^{IaaS} + p_{ij}^{SaaS} + p_{ij}^{NaaS})$. For m applications, the total utility of the MEC system is given by:

$$U = \sum_{k=1}^{m} U_i \tag{9}$$

Therefore, the objective of the whole MEC system with respect to users' monetary costs is given by:

$$max(U) \tag{10}$$

The two constraints of the above optimisation problems are: the allocated resource capacities cannot exceed the total capacities; and each user is allocated resources exactly once in their proximities.

4.2 The Optimisation Problem

We can formulate the above problem as a Min-Max multi objective optimisation problem. Consider a MEC which comprises a datacenter, several edge locations (hosts), and users' jobs that run on a variety of VMs. Find the VM to host mapping, such that: (i) the cumulative energy consumed by the MEC is minimized; and (ii) the performance of the VM is maximized (or VM runtime is minimized). Similarly, regarding SaaS performance of various applications is ensured. Moreover, from networks point of view the available bandwidth (B) is maximised. We further assume performance as the VM runtime, the longer the VM runs the worse will be its performance and vice versa. Mathematically, we can integrate all these into Eq. 11:

$$\begin{aligned} min\Bigg(\sum_{i=1}^{N}\sum_{j=1}^{M} e_{ij}.x_{ij}.\sum_{i=1}^{N} t_{ij}.x_{ij} + \sum_{k=1}^{M} t_k\Bigg) + \\ max\sum B + max_A\Big(\sum_{j\in N} u_j^n.A_{i\in M}^n\Big) + max(U) \end{aligned} \tag{11}$$

where the mapping factor x_{ij} equals 1 if a particular VM i is mapped to host j. Furthermore, e_{ij} represents the energy consumed by host j when a VM i is run. Moreover, t_k denotes the execution time of application k and B is the available bandwidth. The constraints of the above optimisation problems are:

$$\sum_i x_{ik}^n \leq C_k^n \quad \& \quad \sum_{l\in N}^{m} x_{ik}^l \leq r_{ik}^n \tag{12}$$

where the first one indicates that various resources allocated to VMs cannot exceeds the hosts (providers) capacities while the second one ensures that only the required resources are offered to VMs. Note that, application runtimes, network performance, and energy consumption are inversely proportional to each others i.e. an increase in one value can decrease the other one's value. In the single objective optimisation, the SaaS provider wants to maximize income by minimizing monetary costs (i.e. lower costs may increase number of customers) – and so cost minimization implies minimizing SLA violations; SLA violations, then, depends on resources procured from their providers of network and IaaS (who, presumably want to achieve the same for the same reasons). In such scenarios, an assumption would be needed, so that the IaaS provider who is trying to get the SaaS provider to use more resources, unnecessarily, can be avoided (left-hand side of Eq. 12). Furthermore, it is also possible that various providers are satisfying the highest paying customers first - in which case an additional constraint would be necessary over the allocation. The later one can also be assumed similar to native applications of IaaS providers - Gmail on Google cluster will get preference than running on Azure cluster [7]. This means that the MEC allocation problem is complex and cannot be easily solved with simple optimisation techniques.

An alternative approach is to account for individual player's objectives, separately. For example, the providers (IaaS, NaaS) deploying the MEC services aim to maximize their profits through selling more resources (using certain price models) and/or reducing energy consumption (paying less energy bills). Moreover, the SaaS has to account for: (i) gains from selling their applications; and (ii) the amount paid to providers (IaaS, NaaS), when deciding on their resource demands. The providers, first, set prices for their services. The SaaS providers, decide later on their required computing and network resources for running users application, being aware of the providers' prices. The utility function U of each player comprises: amount for selling; and cost incurred in providing resources [14]. We can divide the whole game into two sub games, subject to various constraints, as discussed in Sec. 4.1:

1) every SaaS decides on the resource demand while maximizing the expected utility, given all other SaaS' demands, i.e., strategies, as well as the MEC resource prices (Eq. 8); and
2) the profit of each provider (IaaS, NaaS) is the revenue obtained from charging the SaaS for IaaS, NaaS and MEC resources (SaaS x pays a unit price p_x to each MEC provider) minus the incurred cost (Eq. 10);

where cost is a function of the resource demand, e.g., energy consumed, performance gains and etc. Moreover, the SaaS providers may want to run users' applications in their close proximity (nearest available resources) in order to ensure expected levels of performance (in terms of low latency). Furthermore, IaaS, NaaS compete each other for providing resources and the SaaS compete for provisioning better services. The game solution guarantees that the resource price or allocation, chosen by various providers (IaaS, NaaS), increases their profits, such that SaaS providers achieve

optimal performance for applications which also increases their utility.

5 Proposed Solution

Game theory is largely used for analysing competitive interaction among various providers [15]. We model the above problem as a non-cooperative game where customers - SaaS, resource providers - IaaS, and access networks (or network providers - NaaS) act selfishly according to their own particular objectives [14], [16], as described in Sec. 4. As described in [12], multi-objective optimisation problems can be solved in two ways: (i) concurrently solve all objectives; and (ii) solve one objective first, and then make it a constraint on the next one. Moreover, various objectives can be combined into a single metric, and then solved as a single objective problem [10]. To concurrently solve multi objectives, Lagrange multipliers is one of the classical technique to address such problems. The Lagrangian will converge all objectives to a single saddle point. The Hungarian method is also used to solve such problems, particularly, cooperative games where coalition can formed among various service providers [16]. Since, we assume our game of non-cooperative nature [8], [17], and auction theory is a suitable tool to solve such kinds of games [14]. Therefore, we also solve the allocation problem with an auction theory using the bidding strategy. Our game theoretical approach is inspired from the previous work, as presented in [16].

We assume the whole MEC as a multi-agent system that consists of three different layers. At the top-level, a global resource manager (broker) is responsible to assign VMs requests to a particular MEC. In the middle layer, a local manager is associated with each MEC that is responsible for assigning VMs requests to appropriate computational resources (also knows as agents). In the third layer, agents are responsible to run VMs. In our game, the local manager can submit bids for execution contracts to the global manager. Subsequently, the broker will select the winning MEC through a sealed-bid auction. The bids are computed (by a local manager) using a particular strategy at each server provider using various characteristics of the application and infrastructure. To effectively estimate the runtime for a contract bid, every local manager will ask all agents in its related MEC for estimates in order to create a runtimes matrix. The local manager then chooses which VMs it can execute and at what price. These details are then passed to the global resource manager for taking appropriate allocation decisions.

The steps involved in resource allocation are described in Algorithm 1. The core module of the proposed allocation technique is the bidding strategy. Each service provider is associated with its own and a particular biding strategy, which is described later in Sec. 5.1. All the bids from various providers i.e. IaaS, SaaS, NaaS, are computed, sorted, and are converted to a single (combinatorial) bid which is shared with the global resource manager. The global resource manager, then, chooses the local manager with the highest bid to run the VM. We can also use the Hungarian method to choose the optimal allocation strategy for a particular application (SaaS) [12]. Besides resource allocation, the global resource manager (broker) is responsible to consolidate workloads within the remote cloud and across several cloudlets. The consolidation process ensures that all cloudlets are balanced (w.r.t workloads) and can be achieved using service migration technique, as described in Algorithm 2. Moreover, appropriate service migration techniques guarantee energy savings and workload' expected levels of performance. Furthermore, VMs reallocation through service migration techniques, across various servers or MECs, can be modelled as a cooperative or semi-cooperative game in which various agents or local resource managers can help each other to run them, on appropriate resources [16]. In this paper, albeit we consider service migrations, however, they are modelled and considered as a semi co-operative game, but, not a complete co-operative game. In the near future, we will consider service migrations a complete co-operative game; an will try to ensure the existence of the Nash equilibrium.

Algorithm 1: VM placement algorithm

Input: List of MECs (N), List of hosts in n^{th} MEC (H_n), List of VM requests (V)
Output: Efficient VM placement

```
for each player p ∈ N do
    for each mec ∈ M do
        resource manager de-queues its job queue and announces that a VM_i is ready for bidding ;
        for each agent j do
            estimate runtime for VM_i ;
            tempraray en-queues VM_i into local job queue to check its possiblity of execution ;
            if VM_i is executable on j then
                bid_j = compute bid using Eq. 13
            else
                bid_j = 0 (since runtime = ∞)
            end if
        end for
        sort agents in ascending order of their bids ;
    end for
    take bid from network provider ;
    take bid from application provider ;
end for
sort all bids in ascending order w.r.t group value ;
agt ← the agent with the lowest bids (Hungarian) ;
allocate VM_i to agt ;
return output
```

5.1 The Bidding Strategy

The core component of the proposed technique is the bidding strategy that varies with respect to various service providers involved within the MEC system. Each bid represents a possible VM schedule at certain cost of energy. For example, for IaaS with H total number of hosts the bid of each server h is computed using:

$$bid_{h \in H} = \left(h_e - \frac{1}{h_e}\right) \times r_h \tag{13}$$

where h_e represents the energy consumed and r_h denotes the expected runtime (therefore, performance w.r.t SaaS) of a particular VM on host $h \in H$. The lowest bid demotes

Algorithm 2: Service migration technique

Input: $optimize()$, M, T_v, T_l
Output: migration decision d

```
for each cloudlet ∈ M do
    compute utilisation level of the cloudlet (T_e) ;
     compute network condition (T_c) ;
    if T_e ≥ T_v or T_c ≥ T_l then
        select module m from cloudlet ;
        choose cloudlet t as destination node ;
        d ← true ;
    else
        continue with the for loop ;
    end if
end for
return m, t, d
```

an optimal agent from both IaaS and SaaS perspective. For NaaS, we assume that the bandwidth is offered in sub-channels and is, largely, used for data transmission and communication. Important parameters, here, include the total distance between the IaaS and user, data size, transmission rate, execution delay, and link power consumption. These parameters should be considered in computing the NaaS bid. Furthermore, the broker is aware of the agent's distance from each user. The NaaS bid is given by:

$$bid_{b \in B} = D \times b_{cost} \tag{14}$$

where D denotes the distance between the edge cloud and the agent, while b_{cost} is the channel (bandwidth) cost. We assume that NaaS offers various channels with numerous capacities at different costs just like EC2 instance types. The above bidding strategy can be converted to combinatorial bidding approach where all bids can be computed in one go [18]. In such scenarios, each VM request $\mathcal{R}$ can be represented as a 2-tuple i.e. $\mathcal{R}(\mathcal{C}, \mathcal{B})$ where $\mathcal{C}$ denotes the instance type (size) and $\mathcal{B}$ the required number of bandwidth channels. Note that, the required number of channels B_{ij} are computed using the transmission rate T_{rate}, as given by:

$$T_{rate} = B_{ij}.log_2(1 + \frac{P_{ij}.h_{ij}}{N}) \tag{15}$$

where B_{ij} represents the bandwidth allocated to VM or user j, h_{ij} denotes the channel gain for user j at service provider i and P_{ij} is the transmission power of user j. Further, N is the background noise [18]. Due to experimental simplification, we use, here, the combinatorial bidding approach, given by $[\prod bid_{h \in H}.bid_{b \in B}]$, in order to allocate IaaS, NaaS resources to different services i.e. various modules of the applications.

6 Performance Evaluation

Resource allocation and consolidation can be seen as a kind of bin-packing problem by means of different sizes and costs of bins – where bins represent the MEC's resources (hosts) and items represent various applications for placement. Furthermore, the size of bins represent host's CPU, RAM, storage capacities and costs relate to hosts' energy consumption. Energy and performance efficient resource allocation can be assumed as a multi-objective optimisation problem with the objective(s) to reduce the number of used hosts – as fewer hosts, possibly, decrease the energy consumption. However, this statement may not hold for heterogeneous MECs [19]. Therefore, an alternative approach for heterogeneous systems is to minimise the sum of total bins costs instead of number:

$$min \sum_{i=1}^{n} C_{h_i} \tag{16}$$

where C_h is the cost of host h. We consider C as the product of energy (E) and performance – execution time (T). Usually, bin-packing problems are solved using various heuristic approaches which may not guarantee optimal results but they are considered fast enough to deal with, particularly, large problems [20]. It is possible to assume an analogous resource allocation problem as moving from a particular state of the datacenter to an ideal state – the one using the fewest hosts. We achieve a datacenter state through implementing various placement techniques (Cloud-Only, Edge-ward and bidding-based Game-theoretic - epcAware), with application packing then needing to ensure energy and performance efficiency.

6.1 Modelling Applications

To demonstrate the efficiency of the proposed technique, we use two kinds of applications: (i) near real-time – where we model the well-known electroencephalography (EEG) tractor beam game (EEGTBG); and (ii) delay-tolerant – where a video surveillance/object tracking (VSOT) application is modelled. In respect of (i), several players attempt to collect items through concentrating on them - the better the concentration, the higher chances to collect more items. A true, on-line, real-time, experience can be observed through fast processing and low response times.

The EEGTBG application has 5 modules: (a) EEG sensor; (b) display; (c) client; (d) concentration calculator; and (e) coordinator. The EEG headset probes user concentration and communicates raw data to the client module. Subsequently, the client module transmits reliable data to the concentration calculator module, which calculates the user level of concentration. Furthermore, the computed concentration level is sent back to the client module, to update the game status (display) on the player's device. The coordinator module collects and distributes measured concentration among all players. As described in [9], the three modules i.e. sensor, display, and client are placed in the mobile device. However, the other two modules i.e. the concentration calculator and coordinator could be placed either in cloudlets or datacenter. Various modules of the EEGTBG application are shown in Fig. 2.

The VSOT application depend on a set of distributed cameras that could track movement, having six modules: (a) camera; (b) motion detector; (c) object detector; (d) object tracker; (e) user interface; and (f) pan, tilt, and zoom (PTZ) control. The camera streams video to the motion detector that, subsequently, filters the streamed video and transmits the video of interest, i.e. in which motion was detected, to the object detector module. The object detector recognises

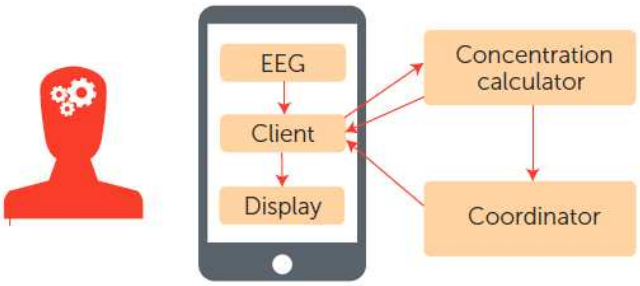

Fig. 2: EEGTBG application modules [9]

the moving objects, and sends their identification and position data to the object tracker module. Sequentially, the object tracker calculates the required PTZ and sends the command to PTZ control. We further deliberate that the two modules i.e. motion detector and PTZ control are permanently located within the camera, whereas the user interface runs in the cloud (datacenter). The other two modules i.e. the object detector and object tracker might be placed either in a cloudlet or datacenter.

The above applications can be set up in a MECs infrastructure to yield benefits of lower latency due to the use of edge devices. Moreover, VSOT can work reasonably well under datacenter-distance latencies (greater than 100 milliseconds) [9]. Alternatively, higher delays in EEGTBG application can impact the players real-time observation, making the game weird as its playability might be damaged. We contemplate that both these applications belong to two diverse classes of applications types i.e. delay-sensitive and delay-tolerant, that could benefit from a MECs infrastructure. Various modules of the VSOT application are shown in Fig. 3.

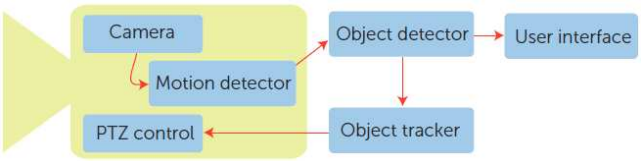

Fig. 3: VSOT application modules [9]

6.2 Experimental Set-up

We use the "iFogSim" simulator [5] to evaluate the performance of various resource placement policies because it: (i) supports the hierarchical composition of IaaS clouds, cloudlets and edge devices; (ii) runs on top of "CloudSim" [21] – the most widely used simulator in the cloud research community; and (iii) supports the measurement of application performance in terms of delays, response and execution times. We assume twelve instances of the VSOT application that run in cloudlet-2 and thirty six instances of the EEGTBG application that run in cloudlets-1 and cloudlet-3, collectively. Initially, eighteen EEGTBG users are playing the game in close proximity of cloudlet-1 location and the other eighteen players are in close proximity of cloudlet-3 location. To emulate mobility and to assess performance degradation that may cause from poor resource allocation and service migrations, we move the EEGTBG players one by one to cloudlet-2. Due to low-latency requirements of EEGTBG application, we assume that a particular player in a cloudlet plays only against other players in the same cloudlet [9]. To simulate service mobility in the context of epcAware allocation and migration policies, appropriate migration decisions are, then, triggered using Alg. 2. The migrations may happen either: (a) among hosts of the remote cloud – inter-datacenter; (b) among hosts of an individual edge cloud or across several edge clouds – inter-fog and intra-fog; and/or (c) among hosts of the edge and remote clouds – fog-datacenter. In respect of (a) and (b), migrations occur if hosts' utilisation levels drop below certain threshold value e.g. 20%. In respect of (c), application modules are moved explicitly, as described later.

Every cloudlet has a processing capability (speed) of 3 heterogeneous servers, as shown in Table 3, that maps to the notion of MIPS (millions of instructions per second), for consistency with the iFogSim simulator, and is connected to the gateway (proxy server) through a link of bandwidth equal to 10 Mbps and latency of 4ms (milliseconds). Moreover, the link between the gateway and the cloud has 10 Mbps bandwidth and 100ms latency. We further assume that edge devices, such as mobile and camera, are connected to the cloudlets through a link of bandwidth equal to 10 Mbps and 2ms latency. The maximum resource (CPU, RAM, and network bandwidth) requirements of each application's module are shown in Table 2. However, at scheduling time, prior to execution, each application's (module) resource requirements are unknown. Later on, resources could be predicted. Moreover, we assume that the application module workload (modelled as tuple) is dynamic that changes with time to time. Moreover, every tuple is assigned a particular task (i.e. fixed number of MIPS) which utilises the VM resources, using a normal distribution - most likely resource usage in Google cluster, that could possibly create variations in runtimes.

Similarly, we account for resource contention or interference on various servers, that could, possibly, degrade applications performance. The cloud consists of 12 heterogeneous hosts that corresponds to three different CPU platforms, as shown in Table 3. The idle (P_{idle}) and maximum power consumption (P_{max}) of hosts were taken from the SPECpower benchmarks. To maintain consistency with the iFogSim simulator, speed of the hosts are transformed to MIPS. Each hosts can run several VMs where each core of a particular host corresponds to a single vCPU of a VM instance. We, further, assume that every module of the fog application runs in a VM instance and the speed of VM is exactly equal to the CPU requirement of each module, as shown in Table 2. We also account for migration costs [19], resource heterogeneity (in terms of CPU architecture) and resource contention due to co-location i.e. when similar modules of same applications are placed on same host and compete for same resources [22]. The resource contention and CPU heterogeneity parameters for various hosts, as shown in Table 3, were taken from our previous work. Each migration degrades the VM performance by 10%, as investigated in [23]. Moreover, the energy cost of each VM migration E_m is modelled using Eq. 17:

$$E_m = 0.512 \times VM_{data} + 20.165 \quad (17)$$

TABLE 2: CPU, RAM and network bandwidth (BW) requirements (in MIPS, MB, and Mbps, respectively) for both applications and their various modules [9]

	VSOT				EEGTBG		
	Object detector	Motion detector	Object tracker	User interface	Client	Concentration calculator	Coordinator
CPU	550	300	300	200	200	350	100
RAM	30	25	25	20	50	60	30
BW	100	100	100	400	500	200	200

where VM_{data} denotes the amount of memory (measured in MBs) allocated to the VM plus memory pages dirtied during the migration process [10]. Note that, speed of hosts and VMs are transformed into the notion of MIPS in order to keep consistency with the CloudSim and iFogSim simulators. Moreover, each service requirements (CPU) are according to the default setting of iFogSim (number of MIPS). We assume that each module of various application utilises its provisioned VM resources in whole. Furthermore, each provider's utility is computed using a particular cost model. In our experiments, there are 12 servers in cloud, 1 proxy server, and 3 servers per cloudlet. Each service is placed on the most energy, performance and cost efficient host, using Alg. 1, in such a way that every service requirements are ensured [24], [25].

TABLE 3: Different characteristics of various hosts used in the simulated MEC system [P_{idle} and P_{max} denote the host' idle (0% utilised) and maximum (100% utilised) power consumption, respectively]

CPU model	Speed (MHz)	No of cores	Memory (GB)	P_{idle} (Wh)	P_{max} (Wh)	Total amount
E5430	2830	8	16	166	265	
E5507	2533	8	8	67	218	100
E5645	2400	12	16	63.1	200	

6.3 Evaluation Metrics

The metrics used to evaluate the energy, performance and cost efficiency of the proposed resource allocation and migration policies are: (i) total energy consumed (E) measured in KWh; (ii) execution time (T) measured in seconds – as application's performance is inversely proportional to T; (iii) delay (D) among various modules which is measured in milliseconds; (iv) total number of migrations – fewer migrations may ensure higher performance levels and lower energy consumption; and (v) provider's utility. Note that, the providers' utilities actually describe the money (in US dollars - $) for energy consumption bills (IaaS), resources being sold (NaaS, IaaS) and instances being purchased (SaaS, IaaS). We assume the energy cost at the rate of 0.08$ per KWh, VM instances and bandwidth costs at 0.07$ and 0.02$ per hour, respectively. We also assume that VMs running in cloudlets incurs an additional cost of approximately 20% greater than the cloud VMs [24].

6.4 Experimental Results and Discussion

Table 4 describes experimental results for various resource placement, consolidation and management policies. The results show that various approaches to resource placement and placement methodologies (cloud-only, edge-ward) offer variations in energy consumption, and workload performance. Largely, our findings are consistent with previous outcomes [9] that the edge-ward approach is approximately 3.24% to 8.94% energy and ∼0.81% performance efficient than the cloud-only placement method. However, if resource contention (due to co-located VMs) and platform heterogeneities (due to CPU architectures) are considered at the cloudlets, then the edge-ward approach cannot ensure performance benefits. Therefore, performance gains are obtainable if certain performance-aware placement policies such as "epcAware-NC" or "epcAware-SC" are taken into account.

TABLE 4: Experimental results - energy is the sum of both dataceneter and cloudlets usage and R means total execution time [without migrations]

Allocation Policy	Providers utility ($)			Energy (KWh)	R (hours)
	IaaS	SaaS	NaaS		
Cloud only					
Random	279.3	1.82	0.48	3,491.7	26.12
FCFS	262.8	1.75	0.5	3,285.3	24.98
Delay-priority	239.7	1.68	0.5	2,996.0	23.93
epcAware-NC	225.8	1.68	0.52	2,822.1	23.87
epcAware-SC	223.0	1.68	0.53	2,787.9	23.51
Edge-ward					
Random	270.2	1.96	0.44	3,377.0	27.95
FCFS	239.3	1.89	0.47	2,991.5	26.69
Delay-priority	231.9	1.75	0.5	2,898.9	25.07
epcAware-NC	216.1	1.68	0.52	2,701.0	23.87
epcAware-SC	215.4	1.61	0.53	2,692.5	23.32

Migrations which may happen due to cooperation among various players (service providers) can increase energy, performance, therefore costs saving. However, if migration costs, resource heterogeneities and contention are considered, then migrations could potentially degrade applications' performance up to -10.97. The figures, as described in previous paragraph, can further be improved i.e. ∼11.95% energy savings and ∼3.56% performance gains, if certain epcAware service migration techniques are considered, as shown in Table 5. The total number of migrations, as shown in Fig. 4, affect the overall performance degradation – higher number of migrations potentially lead to applications lower performance. The migrations may happen either inter-fog nodes or/and intra-fog platform. Moreover, inter-datacenter migration and fog-datacenter migrations are also possible. Our investigation suggest that, in MEC, migrations may, largely, happen inter-datacenter which is justifiable due to increased number of nodes in datacenter. Moreover, since the Random policy puts modules scattered,

therefore creating maximum opportunities for migrations, as shown in Fig. 4. However, if mobility is considered [9], then moving application modules across several cloudlets or between cloudlets and datacenter would be essential. This also demonstrates that migrations can be reduced up to 52.8% if providers cooperate and schedule workloads on appropriate resources.

TABLE 5: Experimental results - energy is the sum of both dataceneter and cloudlets usage and R means total execution time [with migrations]

Allocation Policy	Providers utility ($) IaaS	SaaS	NaaS	Energy (KWh)	R (hours)
	Edge-ward				
Random	237.9	1.96	0.44	2,973.5	28.03
FCFS	219.6	1.89	0.45	2,745.2	27.4
Delay-priority	224.3	1.96	0.44	2,803.6	27.82
epcAware-NC	215.8	1.61	0.54	2,697.7	23.02
epcAware-SC	215.4	1.61	0.54	2,691.8	23.01

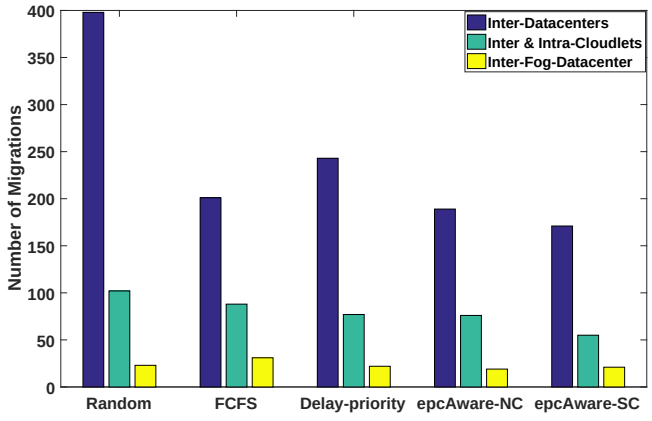

Fig. 4: Total number of migrations [including inter-datacenter, inter and intra-fog, and fog-datacenter]

Table 6 shows the percentage improvements in energy efficiency and performance gains when resource allocation problem is modelled, using a game approach, among different service providers with conflicting goals. Our findings demonstrate that cooperative-based game approaches to allocation ensure higher efficiencies. To study the impact of long-running monitoring services and short-running game applications, we changed the runtimes of applications in the above experiment, accordingly [10]. We observed that monitoring applications (VSOT) modules are more cost-effective to migrate than game application (EEGTBG) modules in terms of total number of migrations (reduced), and performance loss (reduced). Longer runtimes could guarantee recoverability of the migration costs [19] through subsequent running over the target and, therefore, cost savings. This create further gap for investigation of placing and running modules of monitoring services in VMs and game modules in containers. This will also ensure reduction in application migration times since container images are smaller than VM images.

The scheduling and migration decisions in fogs also affect the total network use, as shown in Fig. 5. For example,

TABLE 6: Percentage improvements of using game-theoretic methods to placement [Base is the delay-priority approach, E and P denote energy and performance (%), respectively]

Policy	Base E	Base P	epcAware-NC E	epcAware-NC P	epcAware-SC E	epcAware-SC P
cloud-only	-	-	5.8	0.25	6.94	1.76
edge-ward	-	-	6.83	4.79	7.12	6.98
	-	-	3.78	17.25	3.98	17.29

migrations increase the utilisation levels, that could be up to 9.65%, of the used bandwidth irrespective of the network capacity. Moreover, if network provider allocate their channels or bandwidth capacities in such a way that each application or user gets exactly what is needed for quality of service (QoS). Then the allocated bandwidth can be decreased, but, utilised more. When placement and migration decisions are taken based on the cooperation among various service providers, then approximately 3.83% to 12.86% network capacity could be saved which translate to cost saving from NaaS perfective.

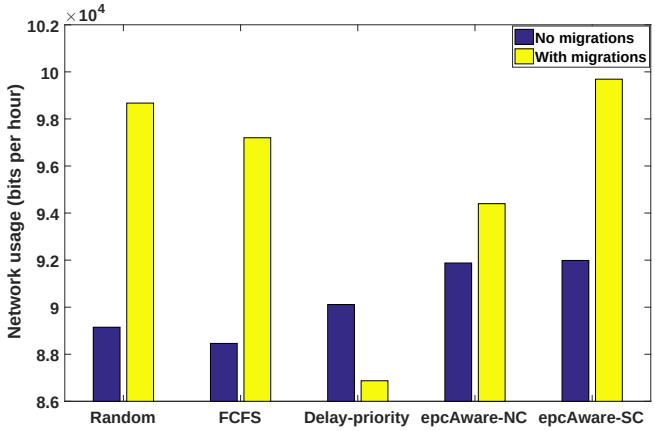

Fig. 5: Network usage for various policies [higher values are better than the lower values]

Table 7 shows average delays (measured in seconds) for various applications and resource placement and migration policies. The delay actually represents the time needed to complete a particular task (tuple). For example, in case of the VSOT application, the delay represents the time between a sensor notices an object and PTZ controller to identify the object. The edge-ward approach is somehow offering lower delays than the remote cloud-only technique - which is justifiable due to short distances. Moreover, the application delay is affected through increasing the number of users (or application modules), network usage (available bandwidth), and total number of migrations, which may possibly produce contention if same modules are placed on same resources. We observed that the classical FCFS policy offers the least application delays since it prefers to put application modules on the remote cloud. Moreover, the delay-priority policy [9] offer acceptable delays, however, for large number of users (concurrent application modules) it cannot guarantee consistent lower delays. Albeit, our proposed policies

can ensure lower delays if migrations are not taken into account. However, since the proposed algorithms look for opportunities to reduce network (bandwidth) provisioning – the NaaS objective; therefore, combined with migrations, improvement in delays is not trivial and this needs further work.

TABLE 7: Average application delays (in seconds) the ± denotes standard deviation over the average [minimum values are 'best']

Policy	epcAware-NC		epcAware-SC	
	EEGTBG	VSOT	EEGTBG	VSOT
cloud-only	18.7±1.1	17.8±1.3	23.8±1.9	20.1±2.2
edge-ward	13.6±2.4 30.0±3.3	15.1±2.1 29.9±2.8	16.7±2.3 24.6±2.6	15.9±1.8 23.2±2.6

7 Related Work

Gupta et al. [5] proposed two application placement policies: (i) cloud-only that places all modules of the application in cloud datacenter; and (ii) edge-ward that favours to run modules of the application on edge/fog devices. However, in case of (ii), if the allocation of a fog device is not appropriate, then resources from other fog devices or cloud datacenters could be provisioned. Moreover, a simulator "iFogSim" is developed to simulate mobile edge cloud platforms. Empirical evaluation of both allocation policies for two real-time applications suggests that the edge-ward policy significantly improves the application's performance and reduces the network traffic. Bittencourt et al. [9] investigated three different allocation policies; (i) concurrent – the requests at cloudlet are served in the same cloudlet; (ii) FCFS – requests are served in the order of their arrival (edge-ward); and (iii) delay-priority – the lowest delay applications are scheduled first and the remaining applications are placed according to edge-ward placement. Moreover, application (modules) migration is supported; and if a particular module is moved to a target device, all other modules are also moved. Their investigation suggests that if an application performance is the worst on a fog device, probably, due to more number of connected users, its migration to the cloud datacenter is performance efficient. Moreover, network traffic is reduced.

Guerrero et al. [26] suggested a decentralised placement policy that runs popular and most widely used applications closer to the end-users (close proximity). Popularity of the applications is computed through statistical distributions of their access rate i.e. service request rate (λ). For each device, the algorithm analyses λ of every service, and migrate the lower requested services to upper devices (in edge-ward fashion). Their experimental evaluation suggests that such a placement method significantly improves the network usage and service latency of the most widely used and popular applications. However, the existing trade-off between the network usage and applications' latencies (delays) is not investigated.

Taneja et al. [27] suggested an application placement policy that puts various modules of the fog application on suitable resources. The proposed policy first sorts all the network nodes and application modules in ascending order of their capacities and requirements, respectively. Then it searches for most efficient nodes that could meet the module requirements; and the module is run. Their research suggests that the proposed allocation scheme could significantly reduce the network usage and improve application latency as compared to traditional cloud allocation policy. Moreover, overall energy consumption varies with respect to the number of fog devices in the infrastructure. For small number of fog devices, traditional cloud allocation policies could beat fog placement. However, energy consumption of the proposed placement strategy could be optimised for large number of IoT and fog devices.

Skarlat et al. [28] have modelled the service placement problem in fog as integer linear programming – find the optimal mapping between services (applications) and computational resources; to optimise fog utilisation while meeting QoS requirement, particularly, deadlines. Moreover, services are prioritised base on their deadlines. When a service request is received at a particular node, the application is placed on it; and if cannot be accommodated there, then it is placed either: (i) in the same fog colony [28]; (ii) on the closest neighbouring nodes (fog colony); or (iii) on the cloud (in an edge-ward fashion). The experimental results show that the proposed method could utilise the fog landscape for approximately 70% of services, and could reduce the execution cost up to 35% as compared to execution in the cloud only approach.

Brogi et al. [29], [30] have proposed a software prototype "FogTorch" that could deploy applications over the fog infrastructure such that all hardware, software and QoS requirements (i.e. bandwidth, latency) are fulfilled. A smart agriculture application has been modelled [30]; and a 3-layer fog infrastructure has been suggested for its deployment. Empirical evaluation of the proposed prototype shows that it could successfully return all eligible deployments (resource provision) for several optimisation scenarios, requirements and needs. Tuli et al. [25] proposed HealthFog, a framework which integrates deep learning methods within the fog infrastructure to run health monitoring system i.e. heart disease analysis using IoT devices.

Plachy et al. [31] have discussed dynamic service placement in mobile edge clouds. Mobility of a fog user is predicted, and, instead of migrating the application (running inside a virtual machine), an alternative network path is selected to connect user at the target. Experimental results show a minimum 10% improvement in response time (delay). Furthermore, a service placement technique, based on predicted future costs of its placement, is also presented. To model user mobility, service migration between cloudlets, and, cloud datacenter is also investigated in [32]. Several prediction models are suggested to estimate the cost of running and migrating a particular service. Both, off-line and on-line service placement problems are solved using various placement algorithms.

Various techniques for migrating (live) services in fog infrastructure are proposed and evaluated in [33]. To minimise the migration time of an application, a three layer architecture is presented; where an idle copy of the application is stored at an intermediate layer. Before migrating the memory states of the application from any source, the application idle copy is migrated first. Later on, only memory pages are copied, that

TABLE 8: Summary of the related work, closest to epcAware, with respect to various evaluation criteria

Parameters		Related Work [5]	[9]	[26]	[27]	[28]	[29]	[25]	[31]	[34]	[8]	[24]	epcAware
Platform	Cloud	✓	✓	✓	✓				✓			✓	✓
	Fog	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	IoT	✓	✓		✓			✓		✓			✓
Provider	IaaS	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
	SaaS					✓						✓	✓
	NaaS				✓			✓	✓	✓	✓		✓
Evaluation metrics	Energy	✓			✓			✓					✓
	Performance	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
	Migration cost									✓			✓
	User costs					✓					✓	✓	✓
	Co-location		✓										✓
	Deadline					✓							
Placement	Single party	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	
	Bi parties										✓		
	Multi parties												✓
Management policy	Allocation	✓			✓	✓	✓	✓		✓	✓	✓	✓
	Migration									✓			✓
	Allocation+Migration		✓	✓					✓				✓
	Game-theoretic										✓		✓

could significantly reduce the migration time. Moreover, a comparison of VM and container based service migration is also elaborated. Mahmud et al. [24] proposed a profit-aware service placement policy for resource provisioning in fog infrastructure. Their outcomes suggest that cloudlet instances are approximately 20% expensive than the cloud instances. However, service migrations and user mobility are not considered.

Urgaonkar et al. [34] have also discussed various strategies for migrating services, in mobile edge clouds, in order to minimise operational costs. The "never migrate" policy places each application at a particular cloudlet with no reconfiguration that may happen due to user mobility. User requests are always routed to the original location of the application. In the "always migrate" policy, user requests are routed to the closest cloudlet with reconfiguration, in such a way that queues with the largest backlogs are served first. Moreover, if the arrival rate of requests at a particular cloudlet exceed its capacity, they are routed towards the cloud (in an edge-ward fashion). The "myopic" policy accounts for reconfiguration, transmission and routing costs; and takes appropriate migration decision such that the sum of these costs could be minimised. The work presented in [34], assume the user mobility as a Markovian process that is solved using Markov Decision Process (MDP) technique. However, as mentioned in [35], users mobility cannot be accurately predicted. A mobility-aware dynamic service placement technique (heuristic based on the Markov approximation) is presented in [35], that accounts for: (i) costs of migrating services; and (ii) the trade-off between performance and operational cost.

Gillam et al. [36] have also discussed VMs, containers and code/functions (Function as a Service – FaaS) in order to explore edge computing for on-vehicle and off-vehicle computation that will be needed to support connected and automated/autonomous driving. To minimise end-to-end latency, the authors suggest that it is essential that computation should be more local to vehicles. However, vehicle mobility will create opportunities for application/code migration, and with notable exception of [33], it is rarely discussed. Zafari et al. [7] modelled the resource allocation problem in MECs and, in particular, when various edge cloud service providers share their extra resources with each others. Moreover, various service providers have their own utility functions which they want to improve through coalition. To solve this multi-objective optimisation problem, a cooperative game theoretic approach is proposed which suggests that service providers can increase their utilities through resource sharing. The same idea has been implemented in clouds [11], where datacenter resources are shared among various IaaS providers that have their own objectives. In both cases, it is ensured that each provider allocates only required services to their native users first. After that, free and unused resources, if available, are shared with other IaaS providers. Compared to our approach, the players are always IaaS providers (therefore, same objectives) while we account for various kinds of players i.e. IaaS, SaaS and NaaS (therefore, multiple objectives which often conflict with each other).

Ahmed et al. [37] used game theory for scheduling tasks in a multi-core system such that energy consumption is minimised and performance is ensured. Similarly, Khan et al. [16] studied various game-theoretical methods for resource allocation in multi-agent computational grids. The work in [12] extends [16] in order to optimize grid energy consumption and workload performance through game-based resource allocation techniques. All these proposals consider, only, a single service providers; and have ignored allocation when services are offered by various providers, in particular, having conflicting goals. Moreover, service migrations are not taken into account. Li et al. [8] formulated the task offloading problem in MEC system as a non-cooperative game; where each player can selfishly minimise his own pay-off through using an appropriate strategy. Moreover, they proposed various algorithms to find the Nash equilibrium. Table 8 describes summary of the related work. We believe, information in this table will help our readers to quickly identify gaps for further research, investigation and improvement.

8 Conclusions & Future Work

MEC is an evolving paradigm that combines computational, storage and communication (network) capacities at the edge of the network through datacenters, in an elastic infrastructure. MECs have potentials to accommodate and run various application types such as: (i) throughput-oriented that need huge computational capacity and network bandwidth; and (ii) latency-oriented applications that need low latency communication and computation in user' proximity. Usually, the computational capacity in edge locations and the wireless access are managed either distinctly or by a single player. Nevertheless, bringing the full potential of MECs entails that edge locations, IaaS, and wireless networks are to be managed in concert. In this paper, we modelled the resource allocation problem in MECs as a non-cooperative game. Further, resource reallocation were taken into account at IaaS layer. We, then, proposed a bidding-based resource allocation and consolidation technique "epcAware" in order to effectively place various applications across various service providers. Empirical evaluation of the proposed algorithm suggests that it is able to manage resources energy, performance and, therefore, cost efficiently. Furthermore, our proposed approach could reduce up to 11.95% energy, and approximately 17.86% user costs at non-significant loss in performance. Moreover, IaaS can reduce up to 20.27% energy bills and NaaS can increase their costs savings up to 18.52% as compared to other methods.

This emerging technology has still a research gap for identifying the locations where such small datacenters should be installed. For example, potentially these can be deployed in universities, hospitals, mobile base stations and/or shopping malls – where their operation and management is more affective or possible. Once installed, what kind of services should they host and how the available resources should be allocated to customers' applications. Similarly, if mobility is involved, then how the available resources or running services should be managed or migrated amongst small datacenters (cloudlets), and geographic areas. The aim of our further research would be to investigate and answer these kinds of questions from a geographic area perfective – while accounting for resource and energy costs variations. The first objective would be to investigate, where cloudlets should be deployed such that service agility and performance is guaranteed. Our second objective would be to study the resource allocation and placement policies in order to decrease energy and network usage, while performance and user costs are not affected. The third objective would be to propose a novel management framework, in order to efficiently manage resources using distributed schedulers instead of a single scheduler. Moreover, in the future, we will reconsider the aforementioned problem for further investigation, and mathematical proof for finding a Nash equilibrium.

Acknowledgements

This work is supported, in part, by the Abdul Wali Khan University Mardan (AWKUM), Pakistan and, in part, by the Higher Education Commission (HEC), Pakistan. Furthermore, this work is partially supported by an Australian Research Council (ARC) Discovery Project. We are thankful to Dr. Luiz F. Bittencourt, from the University of Campinas (UNICAMP), Brazil, for helping us to setup essential simulation platform in order to run the experiments performed in this manuscript. The research problem tackled in this paper was also discussed with Prof. Erik Elmorth from Umea University, Sweden; during an interview for the post-doc position.

References

[1] A. Lebre, J. Pastor, A. Simonet, and F. Desprez, "Revising openstack to operate fog/edge computing infrastructures," in *Cloud Engineering (IC2E), 2017 IEEE International Conference on*. IEEE, 2017, pp. 138–148.

[2] M. Abderrahim, M. Ouzzif, K. Guillouard, J. Francois, and A. Lèbre, "A holistic monitoring service for fog/edge infrastructures: a foresight study," in *Future Internet of Things and Cloud (FiCloud), 2017 IEEE 5th International Conference on*. IEEE, 2017, pp. 337–344.

[3] M. Chiang and T. Zhang, "Fog and iot: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, 2016.

[4] M. Ali, A. Anjum, M. U. Yaseen, A. R. Zamani, D. Balouek-Thomert, O. Rana, and M. Parashar, "Edge enhanced deep learning system for large-scale video stream analytics," in *Fog and Edge Computing (ICFEC), 2018 IEEE 2nd International Conference on*. IEEE, 2018, pp. 1–10.

[5] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[6] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format+ schema," *Google Inc., Mountain View, CA, USA, Technical Report*, 2011.

[7] F. Zafari, J. Li, K. K. Leung, D. Towsley, and A. Swami, "A game-theoretic approach to multi-objective resource sharing and allocation in mobile edge," in *Proceedings of the 2018 on Technologies for the Wireless Edge Workshop*. ACM, 2018, pp. 9–13.

[8] K. Li, "A game theoretic approach to computation offloading strategy optimization for non-cooperative users in mobile edge computing," *IEEE Transactions on Sustainable Computing*, 2018.

[9] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-aware application scheduling in fog computing," *IEEE Cloud Computing*, vol. 4, no. 2, pp. 26–35, 2017.

[10] M. Zakarya and L. Gillam, "Energy and performance aware resource management in heterogeneous cloud datacenters." Ph.D. dissertation, University of Surrey, 2017.

[11] F. Zafari, K. K. Leung, D. Towsley, P. Basu, and A. Swami, "A game-theoretic framework for resource sharing in clouds," *arXiv preprint arXiv:1904.00820*, 2019.

[12] S. U. Khan and I. Ahmad, "A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids," *IEEE Trans. Parallel Distrib. Syst.*, vol. 20, no. 3, pp. 346–360, 2009.

[13] Q. He, G. Cui, X. Zhang, F. Chen, S. Deng, H. Jin, Y. Li, and Y. Yang, "A game-theoretical approach for user allocation in edge computing environment," *IEEE Transactions on Parallel and Distributed Systems*, 2019.

[14] Z. Xiong, Y. Zhang, D. Niyato, P. Wang, and Z. Han, "When mobile blockchain meets edge computing," *IEEE Communications Magazine*, vol. 56, no. 8, pp. 33–39, 2018.

[15] H. Zhang, Y. Zhang, Y. Gu, D. Niyato, and Z. Han, "A hierarchical game framework for resource management in fog computing," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 52–57, 2017.

[16] S. U. Khan and I. Ahmad, "Non-cooperative, semi-cooperative, and cooperative games-based grid resource allocation," in *20th International Parallel and Distributed Processing Symposium (IPDPS 2006), Proceedings, 25-29 April 2006, Rhodes Island, Greece*, 2006.

[17] W. Cai, F. Chi, X. Wang, and V. C. Leung, "Toward multiplayer cooperative cloud gaming," *IEEE Cloud Computing*, vol. 5, no. 5, pp. 70–80, 2018.

[18] H. Zhang, F. Guo, H. Ji, and C. Zhu, "Combinational auction-based service provider selection in mobile edge computing networks," *IEEE Access*, vol. 5, pp. 13 455–13 464, 2017.

[19] M. Zakarya and L. Gillam, "An energy aware cost recovery approach for virtual machine migration," in *International Conference on the Economics of Grids, Clouds, Systems, and Services*. Springer, 2016, pp. 175–190.

[20] T. C. Ferreto, M. A. S. Netto, R. N. Calheiros, and C. A. F. De Rose, "Server consolidation with migration control for virtualized data centers," *Future Generation Computer Systems*, vol. 27, no. 8, pp. 1027–1034, 2011.

[21] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[22] F. Xu, F. Liu, and H. Jin, "Heterogeneity and interference-aware virtual machine provisioning for predictable performance in the cloud," *IEEE Transactions on Computers*, vol. 65, no. 8, pp. 2470–2483, 2016.

[23] A. Beloglazov and R. Buyya, "Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 7, pp. 1366–1379, 2013.

[24] R. Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya, "Profit-aware application placement for integrated fog–cloud computing environments," *Journal of Parallel and Distributed Computing*, vol. 135, pp. 177–190, 2020.

[25] S. Tuli, N. Basumatary, S. S. Gill, M. Kahani, R. C. Arya, G. S. Wander, and R. Buyya, "Healthfog: An ensemble deep learning based smart healthcare system for automatic diagnosis of heart diseases in integrated iot and fog computing environments," *Future Generation Computer Systems*, 2019.

[26] C. Guerrero, I. Lera, and C. Juiz, "A lightweight decentralized service placement policy for performance optimization in fog computing," *Journal of Ambient Intelligence and Humanized Computing*, Jun 2018.

[27] M. Taneja and A. Davy, "Resource aware placement of iot application modules in fog-cloud computing paradigm," in *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on*. IEEE, 2017, pp. 1222–1228.

[28] O. Skarlat, M. Nardelli, S. Schulte, and S. Dustdar, "Towards qos-aware fog service placement," in *Fog and Edge Computing (ICFEC), 2017 IEEE 1st International Conference on*. IEEE, 2017, pp. 89–96.

[29] A. Brogi and S. Forti, "Qos-aware deployment of iot applications through the fog," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1185–1192, 2017.

[30] A. Brogi, S. Forti, and A. Ibrahim, "How to best deploy your fog applications, probably," in *Fog and Edge Computing (ICFEC), 2017 IEEE 1st International Conference on*. IEEE, 2017, pp. 105–114.

[31] J. Plachy, Z. Becvar, and E. C. Strinati, "Dynamic resource allocation exploiting mobility prediction in mobile edge computing," in *Personal, Indoor, and Mobile Radio Communications (PIMRC), 2016 IEEE 27th Annual International Symposium on*. IEEE, 2016, pp. 1–6.

[32] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, "Dynamic service placement for mobile micro-clouds with predicted future costs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1002–1016, 2017.

[33] A. Machen, S. Wang, K. K. Leung, B. J. Ko, and T. Salonidis, "Live service migration in mobile edge clouds," *IEEE Wireless Communications*, vol. 25, no. 1, pp. 140–147, 2018.

[34] R. Urgaonkar, S. Wang, T. He, M. Zafer, K. Chan, and K. K. Leung, "Dynamic service migration and workload scheduling in edge-clouds," *Performance Evaluation*, vol. 91, pp. 205–228, 2015.

[35] T. Ouyang, Z. Zhou, and X. Chen, "Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing," Tech. Rep., 2018. [Online]. Available: https://mega. nz, Tech. Rep.

[36] L. Gillam, K. Katsaros, M. Dianati, and A. Mouzakitis, "Exploring edges for connected and autonomous driving," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2018, pp. 148–153.

[37] I. Ahmad, S. Ranka, and S. U. Khan, "Using game theory for scheduling tasks on multi-core processors for simultaneous optimization of performance and energy," in *22nd IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008, Miami, Florida USA, April 14-18, 2008*, 2008, pp. 1–6.