Edge Affinity-based Management of Applications in Fog Computing Environments

Redowan Mahmud, Kotagiri Ramamohanarao and Rajkumar Buyya
Cloud Computing and Distributed Systems (CLOUDS) Laboratory
School of Computing and Information Systems
The University of Melbourne, Australia
Email: mahmudm@student.unimelb.edu.au

***Abstract*—Fog computing overcomes the limitations of executing Internet of Things (IoT) applications in remote Cloud datacentres by extending computation closer to data sources. Since most of the Fog nodes are not resource-enriched, accommodation of every IoT application within Fog environments is very challenging. Hence, we need to efficiently identify which set of applications should be deployed in such constrained environments. It becomes even more complicated when application characteristics in terms of urgency, size and flow of input data are considered simultaneously. The necessity for time-optimized execution further intensifies the application management problem. In this work, we propose a policy for Fog environments that distributes application management tasks across the gateway and infrastructure level. It classifies and places applications according to their *Edge affinity*. Edge affinity of an application denotes the relative intensity of different attributes coherent with its characteristics such as user-defined deadline, amount of data to be processed and sensing frequency of IoT devices, which are required to be addressed within Fog environments to meet its Quality of Service (QoS). The proposed policy also minimizes the service delivery time of applications in Fog infrastructure. Its performance is compared with existing application management policies in both *iFogSim*-simulated and *FogBus*-based real environments. The experiment results show that our policy outperforms others in combined QoS enhancement, network relaxation and resource utilization.**

***Index Terms*—Internet of Things, Application placement, Fog computing, Quality of Service, Application classification.**

I. Introduction

The Internet of Things (IoT) devices operating in physical environments continuously generate data. Processing of IoT-data using Cloud datacentre-based applications is subjected to their multi-hop distance from IoT devices [1]. It increases data propagation delay, degrades application's service delivery time and congests the network. To overcome these limitations, Fog computing engages different components at the edge network to provide infrastructure for hosting IoT applications and process data in proximity of sources. Thus, it facilitates applications with reduced service time and lower network load compared to their Cloud-centric execution [2]. In Fog, the computing components, known as Fog nodes, such as personal computers, gateway routers and micro datacentres are deployed in a dispersed manner. They are heterogeneous and most of them are equipped with limited CPUs, RAM and Bandwidth [3]. Therefore, it is difficult to accommodate every IoT application within Fog infrastructure. Inclusion of more nodes to resolve this issue can affect the economic aspects of Fog computing and intensify the communication complexities [4]. In such constrained scenario, infrastructure providers are often instigated to offer execution of IoT applications in Fog environments as a utility. It also urges users to provision a certain number of applications through Fog instances such as virtual machines and containers according to their affordability. However, a system that deals with various applications; in particular, for a remote health monitoring system, it becomes complicated to determine the competent set of applications for Fog-based placement. Assurance of their time-optimized service delivery using Fog infrastructure also turns into a challenging task. To address these cases, management of applications based on their Quality of Service (QoS) requirements is regarded as one of the potential solutions [5].

Distinctive characteristics of IoT applications play important roles in identifying their different QoS requirements. For example, user-defined deadline indicates whether an application is latency-sensitive or latency-tolerant. Reduced data propagation delay is required for latency-sensitive applications to ensure their QoS [6]. Similarly, based on the data sensing frequency of associated IoT devices, execution of an application can be event-driven or stream-oriented. Streaming applications demand congestion-less data propagation so that their QoS can improve [7]. Moreover, applications that deal with images, audios, videos and large text files are required to process a huge amount of data per input than trivial applications addressing boolean data and short messages. They are usually known as data-intensive applications and encapsulate multiple data preprocessing operations such as data filtration, conversion and consolidation along with the actual data analysis operation [8]. Therefore, it is expected to execute them closer to data sources. Otherwise, the amount of data to be transferred through global Internet will increase, and both the computation and communication load on remote computing resources will aggravate. As a consequence, QoS of these application will degrade [4]. However, for a particular application, these characteristics are independent, and their intensity can vary from one to another. Therefore, it is not feasible to take management decision for different IoT applications based on a single characteristics.

There exist several policies that focus on service time, resource and workload-aware management of IoT applications in Fog environments [9] [10] [11]. They barely explore different application characteristics simultaneously and investigate their

influence on application QoS requirements. In some cases, the Fog gateway devices that reside at the user premises and connect the IoT devices to Fog infrastructure, are assumed to perform all required tasks for managing the applications such as their selection and placement [12] [13]. When a large number of gateway devices interact with a Fog infrastructure, it is time-consuming to share the status of Fog instances among all gateways. For a gateway, it is also difficult to cope up with the dynamism of Fog infrastructure. Consequently, the synchronization problem amplifies, and the overhead of resource-constrained gateways increases.

Taking cognizance of these issues, we propose an application management policy for Fog environments that exploits the characteristics of applications in terms of urgency, input size and data flow for their classification and placement. The core innovation of the policy is to handle these multi-dimensional characteristics and their uneven level of dominance through the non-dominated sorting of application's *Edge affinity*. Here, Edge affinity is defined as the relative intensity of various attributes coherent with an application's characteristics such as user-defined deadline, amount of data to be processed and data sensing rate of associated IoT devices; those need to be supported within network edge for enhanced QoS of the application. The proposed policy also places applications on Fog instances using an integer linear programming model and ensures their time-optimized service delivery in Fog environments. Furthermore, it facilitates application management task distribution by selecting the competent applications for Fog-based placement at the gateway level and identifying the actual application-instance mapping at the infrastructure level.

The major **contributions** of this paper are:

- Proposes a policy for Fog environments that manages applications based on multiple characteristics and requirements across the gateway and the infrastructure level.
- Selects applications for Fog-based placement as per their different character-driving attributes and optimizes their service delivery time in Fog infrastructure.
- Evaluates the performance of proposed policy in a *iFogSim*-simulated [14] and a *FogBus*-based [15] real environment, and demonstrates the improvement in QoS satisfaction, network relaxation, resource utilization and data management compared to existing policies.

The rest of this paper is organized as follows: after discussing related work in Section II, the application context and system model are presented in Section III. Section IV proposes the Edge affinity-based application management policy. Section V evaluates the performance of proposed policy in respect to existing policies. Finally, Section VI concludes the paper with directions for future work.

II. Related Work

Different application management policies have already been developed for Fog environments. Binh et al. [9] and Choudhari et al. [16] propose separate policies to optimize excution time and cost by prioritizing applications based on user expectations and service delivery deadline respectively.

TABLE I: A Summary of related work and their comparison

Work	Application characteristics			Prioritized selection	Optimizes	
	Data flow	Input size	Urgency		Time	Load
Binh et al. [9]		✓		✓	✓	
Choudhari et al. [16]	✓		✓	✓		
Nan et al. [17]	✓		✓		✓	✓
Venticinque et al. [6]	✓		✓		✓	✓
Stavrinides et al. [11]	✓	✓	✓	✓		
Dang et al. [18]	✓		✓	✓	✓	
Skarlat et al. [19]		✓	✓		✓	✓
Rehman et al. [21]		✓			✓	✓
Xu et al. [20]		✓			✓	✓
Taneja et al. [10]		✓	✓	✓		✓
Li et al. [22]	✓	✓		✓		✓
Guerrero et al. [23]	✓			✓	✓	
Edge affinity (This work)	✓	✓	✓	✓	✓	✓

Nan et al. [17] conduct trade-off among service time and request loss rate while placing the applications. Venticinque et al. [6] model a policy that classifies applications as per their resource and energy requirements, and maximizes QoS by meeting deadline. Stavrinides et al. [11] prioritize applications based on workload and ensures least completion time. The policy of Dang et al. [18] optimizes application service time and enhances user's experience by organizing Fog nodes in different regions. Skarlat et al. [19] also highlight time-optimized execution of applications with high resource utilization.

Furthermore, Xu et al. [20] discuss a management framework that classifies applications based on deadline, and assists service migration and load distribution. The application management policy of Rehman et al. [21] optimizes energy usage of instances while executing the applications. Taneja et al. [10] also develop a policy that prioritizes application placement on robust Fog nodes to enhance resource utilization. The policy of Li et al. [22] allocates resources according to user-driven popularity of applications and executes them locally based on a threshold value of computing cost. Similarly, Guerrero et al. [23] through their policy, place the most requested applications in Fog and improve network utilization and service latency.

A summary of related works along with the proposed policy is given in Table I. In existing works, different characteristics of applications are not exploited simultaneously for identifying their various QoS requirements. User-defined deadline, amount of data to be processed and data sensing rate of IoT devices are also disregarded while determining placement option for the applications. Consequently, they often fail to leverage the capabilities of Fog computing in dealing with different sorts of applications. In this work, we classify applications and facilitate their placement based on the relative intensity of different attributes those are coherent with their characteristics and required to be supported through Fog infrastructure for meeting their QoS. Our proposed policy also optimizes service delivery time of applications in Fog infrastructure.

III. Application Context and System Model

A. Motivating Scenario

The application context realized in this work is similar to a real-world scenario from Netflix. Netflix is a streaming service where based on the category of subscription, a user can watch

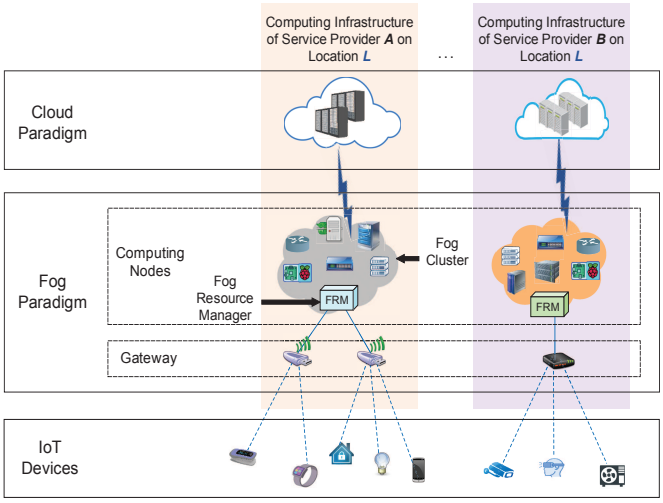

Fig. 1: Fog computing environments

one, two or three different media contents at a time. Netflix-users do not care about what sorts of resources are used to enable these media contents; all that matters to them is whether they can access the allowable number of contents on demand. If an user ask for more media contents at the same time, the user usually sets the preferences according to the quality of contents on Netflix and obtain the extra contents from other streaming services like YouTube or Stan. Such service provisioning is economical for users and assists providers to manage their resources efficiently [24]. We extend this scenario from a localized perspective where Netflix resembles the Fog infrastructure and media contents are the applications. Users can execute a certain number of applications through Fog infrastructure based on their requirements, affordability and resource availability. When more applications are needed to be executed, the allowable number of applications for Fog-based placement are selected from them. Our proposed Edge affinity-based management policy is capable of dealing with such application context in Fog environments. It facilitates the selection of applications having stringent QoS requirements so that the capabilities of Fog infrastructure can be harnessed extensively. Moreover, it forwards the applications with unmet demand to other Fog or Cloud infrastructure for execution.

B. Fog Environments

Different providers can deploy cluster of Fog nodes in various locations. Fig. 1 presents the Fog Clusters (FCs) deployed by provider A and B on location L. In the devised system model, they act as Fog infrastructure. FCs are accessible through Fog Gateways (FGs) located at the user premises. Each Fog node within an FC is capable of hosting different number of Fog instances such as virtual machines and containers as per its capacity. In an FC, the assignment of applications on Fog instances is managed by a specialized node named Fog Resource Manager (FRM) [25]. FRMs maintain a persistent communication with FGs that helps to bind the IoT devices with FCs. FGs receive application placement requests from end users. Placement request for an application includes the details of its character-driving attributes such as user-defined deadline, average amount of data per input and sensing frequency of corresponding IoT devices. On the other hand, FRMs extract the developer-specified minimum resource requirements of applications along with necessary meta-data from a catalogue service [15]. However, due to resource and budget constraints, users are allowed to provision a fixed number of applications on a particular FC. When this limit exceeds, based on user's subscription, FGs communicate with FRM of other FCs or remote Cloud to forward the references of additionally requested applications. The summary of notions used in the system model is shown in Table II.

C. Definition of Edge Affinity

Fig. 2 presents the characteristics of different applications in a three-dimensional space of user-defined deadline, amount of data per input and sensing frequency of associated IoT devices. In the modelled system, whenever the placement request for any application q is received, the specifications of its character-driving attributes are represented by the FG as a vector ϕ_q. For example, if user-defined deadline $\delta_q = 0.250$ seconds, average amount of data per input $\psi_q = 300$ kilobytes and data sensing frequency of IoT devices $\lambda_q = 7$ input per seconds for application q, its $\phi_q =< 0.250, 300, 7 >$. Numerical domain and unit of these attributes are different. Therefore, their values are normalized within [0,1] by FGs using Eq. 1, 2 and 3 in terms of maximum and minimum value for the respective attribute in all application placement requests.

$$\overline{\delta_q} = \frac{\delta_q - min(\delta_{\forall q' \in Q})}{max(\delta_{\forall q' \in Q}) - min(\delta_{\forall q' \in Q})} \tag{1}$$

$$\overline{\psi_q} = 1 - \frac{\psi_q - min(\psi_{\forall q' \in Q})}{max(\psi_{\forall q' \in Q}) - min(\psi_{\forall q' \in Q})} \tag{2}$$

TABLE II: Notations

Sign	Definition
P	Set of available Fog instances in an FC
Γ	Set of all applications selected for placement on an FC
G	Set of all FGs interacting with an FC
Q_g	Set of applications requested to an FG g for placement
R	Set of resources such as CPUs, RAM and Bandwidth
Ω_p^r	Availability of resource $r \in R$ in instance $p \in P$
ω_q^r	Minimum requirements of resource $r \in R$ for application $q \in Q_g$
ϕ_q	Vector of character-driving attributes for application $q \in Q_g$
η_q	Edge affinity of application $q \in Q_g$
δ_q	User-defined service delivery deadline for application $q \in Q_g$
ψ_q	Average amount of data per input for application $q \in Q_g$
λ_q	Sensing rate of associated IoT devices for application $q \in Q_g$
τ^i	Set of i^{th} order non-dominated applications, $\tau^i \subset Q_g$
υ_q	Number of applications that dominate application $q \in Q_g$
Υ_q	Set of applications dominated by application $q \in Q_g$, $\Upsilon_q \subset Q_g$
χ_{cg}	Set of applications selected for placing on FC c by FG g, $\chi_{cg} \subset Q_g$
N	Total number of non-dominated application order
ϑ_q	Value of bottleneck character-driving attribute for application $q \in Q_g$
μ_q	Number of instructions in application $q \in \Gamma$
σ_q	Output data size of application $q \in \Gamma$
Φ_p	Downlink speed of instance $p \in P$
Λ_p	Processing speed of instance $p \in P$
Ψ_p	Uplink speed of instance $p \in P$
t_{pq}^{ι}	Input propagation time for application $q \in \Gamma$ on instance $p \in P$
t_{pq}^{ε}	Execution time of application $q \in \Gamma$ on instance $p \in P$
t_{pq}^{o}	Output propagation time for application $q \in \Gamma$ on instance $p \in P$
ρ_{cg}	Number of applications allowable for FG g to provision in FC c
x_{pq}	Equals to 1 if application $q \in \Gamma$ is mapped to $p \in P$, 0 otherwise.

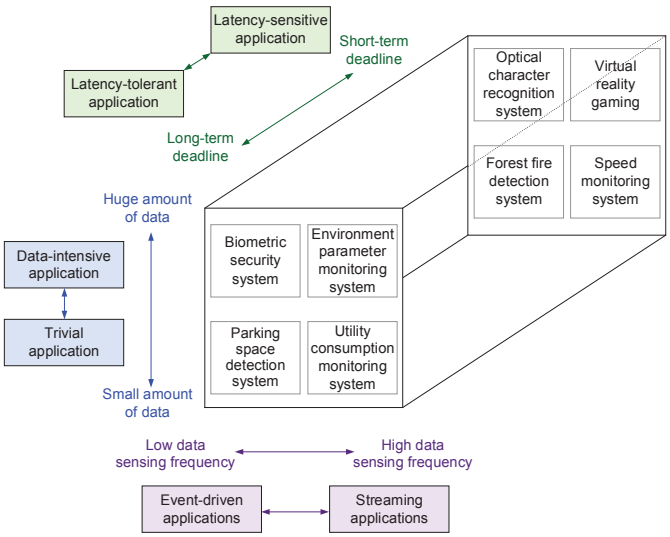

Fig. 2: Variations of IoT applications

$$\overline{\lambda_q} = 1 - \frac{\lambda_q - min(\lambda_{\forall q' \in Q})}{max(\lambda_{\forall q' \in Q}) - min(\lambda_{\forall q' \in Q})} \quad (3)$$

For application q, if the normalized user-defined deadline $\overline{\delta_q}$, normalized average amount of data per input $\overline{\psi_q}$ and normalized input data sensing frequency of IoT devices $\overline{\lambda_q}$ remain closer to 0, application q is considered more latency-sensitive, data-intensive and stream-oriented than other requested applications. Conversely, if they are closer to 1, then application q is regarded as more latency-tolerant, trivial and event-driven compared to others. By definition, vector $\eta_q =< \overline{\delta_q}, \overline{\psi_q}, \overline{\lambda_q} >$ refers to the Edge affinity of application q that contains relative intensity of different character-driving attributes for q in respect of other applications. For any two applications q and q', if Edge affinity are specified as $\eta_q =< 0.10, 0.15.0.20 >$ and $\eta_{q'} =< 0.75, 0.80.0.90 >$ respectively, then application q should get higher priority for Fog-based placement compared to application q' because of its stringent QoS requirements.

However, for a single application q, its $\overline{\delta_q}$ can be closer to 1 whereas value of other two attributes $\overline{\psi_q}$ and $\overline{\lambda_q}$ can be closer to 0. Similarly, for any two applications q and q', $\overline{\lambda_q}$ can be greater than $\overline{\lambda_{q'}}$, although both $\overline{\delta_q}$ and $\overline{\psi_q}$ can be smaller than $\overline{\delta_{q'}}$ and $\overline{\psi_{q'}}$ respectively. These conflicting requirements resist efficient management of applications in Fog environments. To ensure the enhanced QoS of applications, their management policies are required to deal with such cases deliberately.

IV. Edge Affinity-based Application Management

The proposed Edge affinity-based application management policy functions in distributed manner across the gateway and infrastructure level of Fog environments (Fig. 3). It is divided into three phases. At first, FGs classify applications according to their Edge affinity. Later, the allowable number of applications for Fog based placement are selected. FGs forward the references of selected applications to the FRM of subscribed FCs. Finally, FRMs determine the time-optimized application-instance mapping and assign them accordingly. In the following subsections, these phases are described in detail.

A. Classification of Applications

At any FG g, the proposed policy sorts the requested applications in non-dominated order of their Edge affinity. Non-dominated sorting is applied to identify Pareto optimal solutions for multi-objective optimization problems. It also organizes the solutions in different ranks based on the dominance relationship [26]. The proposed policy adopts non-dominated sorting to deal with the conflicting cases in Edge affinity of different applications and classify them in numerical order so that their prioritized selection can be made for Fog-based placement. According to the adopted non-dominated sorting approach, an application q dominates another application q' when their Edge affinity η_q and $\eta_{q'}$ respectively meet the following conditions.

1. η_q is not greater than $\eta_{q'}$ for all normalized character-driving attributes.
2. η_q is strictly smaller than $\eta_{q'}$ for at least one normalized character-driving attribute.

If an application is not dominated by any other applications, its QoS requirements are considered more stringent than theirs. Set of such applications are known as first-order non-dominated applications τ^1. The *ApplicationClassification* procedure shown in Algorithm 1 determines the non-dominated order of different applications based on the dominance conditions. It takes the set Q_g of all applications requested to FG g for placement as arguments (line 1) and consists of two parts:

1. The set of first-order non-dominated applications τ^1 is initialized (line 2). For each application $q \in Q_g$, another set Υ_q and a variable υ_q are introduced (line 3-5). Υ_q refers to the applications dominated by q. On the other hand, υ_q

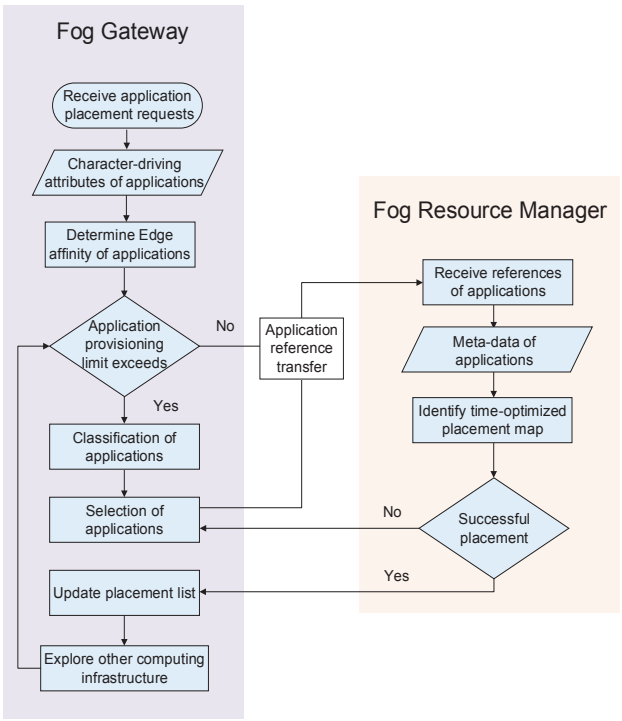

Fig. 3: Flowchart of proposed application management policy

counts the number of applications that dominate q. If all normalized character-driving attributes such as $\overline{\delta_q}$, $\overline{\psi_q}$ and $\overline{\lambda_q}$ of application q are not greater than the same attributes of an application $q' \in Q_g$ and one of the attributes is strictly smaller than that of application q', then q' is considered dominated by q. Hence, it is included in Υ_q (line 6-8). Conversely, if application q' dominates q, υ_q is incremented by 1 (line 9-10). After checking with all $q' \in Q_g$, if υ_q still holds the initial value, it signifies application q as non-dominated in respect of the rest. Therefore, application q is added to the set of first-order non-dominated applications τ^1 (line 11-12).

2. ApplicationClassification procedure exploits the dominance relationship between i^{th} order non-dominated applications and others to determine the set of $(i+1)^{th}$ order non-dominated applications τ^{i+1}. It starts from τ^1 by setting $i = 1$ (line 11). However, τ^{i+1} is initialized only when τ^i exists (line 14-15). Since each $q' \in \Upsilon_q$ is dominated by application $q \in \tau^i$, implicit isolation of q will surely decrease the value of $\upsilon_{q'}$ by 1. For each application $q \in \tau^i$, this technique is applied to all $q' \in \Upsilon_q$ (line 16-18). After such operation, if $\upsilon_{q'}$ becomes 0 for any $q' \in \Upsilon_q$, then it defines q' to be dominated by only application q. Hence, q' is marked as the next ordered non-dominated application to that of application q and q' is added to the set for τ^{i+1} (line 19-20). After exploring all $q \in \tau^i$, i is incremented by 1 so that the set of following non-dominated ordered applications can be traversed in similar way (line 21).

Thus, Algorithm 1 classifies applications in different non-dominated order. For illustration, we consider five applications with $\eta_{q_1} =< 0.84, 0.60, 0.61 >$, $\eta_{q_2} =< 0.33, 0.7, 0.79 >$, $\eta_{q_3} =< 0.68, 0.38, 0.39 >$, $\eta_{q_4} =< 0.14, 0.12, 0.25 >$ and $\eta_{q_5} =< 0.19, 0.16, 0.67 >$ respectively, and find the outcome of Algorithm 1 specifying q_4 as first-order, q_3 and q_5 as second order, and q_1 and q_2 as third order non-dominated application. However, in worst-case, it can have $\mathcal{O}(N \cdot |Q_g|^2)$ iterations where N is the number of non-dominated orders and $|Q_g|$ is the number of applications.

B. Selection of Applications

After classification, FG g executes the *ApplicationSelection* procedure shown in Algorithm 2 to select the allowable ρ_{cg} number of applications for provisioning on a particular FC c. It takes the sets of all different ordered non-dominated applications as arguments (line 1) and contains two parts:

1. A list χ_{cg} and a variable φ_χ are initialized to refer and count the selected applications respectively (line 2-3). A boolean variable κ is also marked with $false$ (line 4). Later, the set τ^i of each i^{th} order non-dominated applications starting from $i = 1$ are explored (line 5). If selection of all applications in τ^i does not surpass the number of allowable applications ρ_{cg}, τ^i is appended to χ_{cg} and φ_χ is updated with the cardinality of τ^i (line 6-8). Otherwise, it is regarded that all applications in τ^i can not be selected for placement in FC c. Hence, κ is updated with $true$ and exploitation of other application sets are postponed (line 9-11). Later, based on the state of κ, τ^i is traversed further to identify which applications from τ^i are competent for selection (line 12-13).

Algorithm 1 Algorithm for classifying applications

```
1:  procedure ApplicationClassification(Q_g)
2:      τ^1 ← ∅
3:      for q := Q_g do
4:          Υ_q ← ∅
5:          υ_q ← 0
6:          for q' := Q_g do
7:              if (δ̄_q < δ̄_q' & ψ̄_q ≤ ψ̄_q' & λ̄_q ≤ λ̄_q') ||
                   (δ̄_q ≤ δ̄_q' & ψ̄_q < ψ̄_q' & λ̄_q ≤ λ̄_q') ||
                   (δ̄_q ≤ δ̄_q' & ψ̄_q ≤ ψ̄_q' & λ̄_q < λ̄_q') then
8:                  Υ_q ← Υ_q ∪ q'
9:              else if (δ̄_q' < δ̄_q & ψ̄_q' ≤ ψ̄_q & λ̄_q' ≤ λ̄_q) ||
                        (δ̄_q' ≤ δ̄_q & ψ̄_q' < ψ̄_q & λ̄_q' ≤ λ̄_q) ||
                        (δ̄_q' ≤ δ̄_q & ψ̄_q' ≤ ψ̄_q & λ̄_q' < λ̄_q) then
10:                 υ_q ← υ_q + 1
11:         if υ_q = 0 then
12:             τ^1 ← τ^1 ∪ q
13:     i ← 1
14:     while τ^i ≠ ∅ do
15:         τ^{i+1} ← ∅
16:         for q := τ^i do
17:             for q' := Υ_q do
18:                 υ_q' ← υ_q' − 1
19:                 if υ_q' = 0 then
20:                     τ^{i+1} ← τ^{i+1} ∪ q'
21:         i ← i + 1
```

2. For each application $q \in \tau^i$, value of its bottleneck character-driving attribute ϑ_q is identified (line 14). For example, if $\overline{\delta_q} = 0.10$, $\overline{\psi_q} = 0.15$ and $\overline{\lambda_q} = 0.20$ for application q, ϑ_q is set to 0.10. It happens because $\overline{\delta_q}$ is the most stringent attribute of q. Later, all application $q \in \tau^i$ are sorted to $\hat{\tau^i}$ in ascending order of their ϑ_q (line 15). For each application $q \in \hat{\tau^i}$, it is checked whether its inclusion for placement in FC c surpasses the allowable number ρ_{cg} (line 16-17). If it is negative, application q is selected and other parameters are updated accordingly (line 18-19). Otherwise, it is regarded that the allowable number of applications are already selected. Hence, their further exploitation is postponed (line 20-21).

Low complexity techniques can be used to perform the

Algorithm 2 Algorithm for application selection

```
1:  procedure ApplicationSelection({τ^1, τ^2, τ^3, ...., τ^N})
2:      χ_cg ← ∅
3:      φ_χ ← 0
4:      κ ← false
5:      for i = 1.....N do
6:          if φ_χ + |τ^i| ≤ ρ_cg then
7:              φ_χ ← φ_χ + |τ^i|
8:              χ_cg ← χ_cg ∪ τ^i
9:          else
10:             κ ← true
11:             break
12:     if κ = true then
13:         for q := τ^i do
14:             ϑ_q ← findMinimum(δ̄_q, ψ̄_q, λ̄_q)
15:         τ̂^i ← ascendingSort(τ^i, ϑ_{∀q∈τ^i})
16:         for q := τ̂^i do
17:             if φ_χ + 1 ≤ ρ_cg then
18:                 φ_χ ← φ_χ + 1
19:                 χ_cg ← χ_cg ∪ q
20:             else
21:                 break
```

operations mentioned in line 14-15. Apart from them, there will be $\mathcal{O}(N + |Q_g|)$ iterations in Algorithm 2 during worst case scenarios. Here, N and $|Q_g|$ denote the number of non-dominated orders and requested applications respectively. However, after executing Algorithm 2, FG g forwards the references of selected applications χ_{cg} to the FRM of FC c for placing them in Fog instances. The applications which are not selected for placement in c are forwarded to other FCs following the same approach or sent to Cloud. If a user is subscribed with multiple FCs, at the FG, their order of exploitation is set based on the preferences of that user.

C. Placement of Applications

Each FG $g \in G$ interacting with an FC c forwards a reference list of selected applications χ_{cg} to the corresponding FRM. The FRM accumulates the received application lists in Γ using Eq. 4. Thus, Γ refers to the set of all applications selected for placement on FC c.

$$\Gamma = \bigcup_{\forall g \in G} \chi_{cg} \tag{4}$$

In FC c, before placing an application $q \in \Gamma$ on a Fog instance $p \in P$, FRM calculates the input data propagation time t^{ι}_{pq}, execution time t^{e}_{pq} and output data transfer time t^{o}_{pq} of application q on that instance using Eq. 5, 6 and 7 respectively. They explicitly depend on the downlink speed Φ_p, processing speed Λ_p and uplink speed Ψ_p of instance p, and the average input data size ψ_q, number of instruction μ_q and output data size σ_q of application q. Based on these metrics, the expected service delivery time t_{pq} of application q on instance p is also determined using Eq. 8.

$$t^{\iota}_{pq} = \frac{\psi_q}{\Phi_p} \tag{5}$$

$$t^{e}_{pq} = \frac{\mu_q}{\Lambda_p} \tag{6}$$

$$t^{o}_{pq} = \frac{\sigma_q}{\Psi_p} \tag{7}$$

$$t_{pq} = t^{\iota}_{pq} + t^{e}_{pq} + t^{o}_{pq} \tag{8}$$

An FRM aims to place an application on that instance which minimizes its service delivery time. For the set of all selected applications Γ, this objective is formulated using a constrained Integer Linear Program (ILP) model as shown in Eq. 9. Solution of the ILP model is defined by a binary decision variable x_{pq} that becomes 1 if application q is mapped to instance p and 0 otherwise. Constraints of the ILP model ensure that an application will not be placed to multiple instances (Eq. 10), its service delivery time will meet the deadline (Eq. 11) and its host instance will have sufficient resources to meet its minimum requirements (Eq. 12).

$$\min \sum_{q \in \Gamma} x_{pq} t_{pq} \tag{9}$$

subject to,

$$x_{pq} \leq 1; \forall q \in \Gamma \tag{10}$$

$$t_{pq} \leq \delta_q; \forall q \in \Gamma \tag{11}$$

$$\omega^{r}_{q} \leq \Omega^{r}_{p}; \forall q \in \Gamma, \forall r \in R \tag{12}$$

The optimization problem in Eq. 9 deals with fixed number of applications and instances. They are usually set according to the resource availability in an FC and the capacity of corresponding FRM in addressing the optimization problem within acceptable time limit using ILP solvers like SCIP [27]. However, if an application misses placement due to the constraints, another application is selected by the associated FG using Algorithm 2.

V. Performance Evaluation

Performance of the proposed policy is evaluated in both real-world and simulated Fog environments. It is also compared with several existing application management policies. Among them, the *Time-aware* management policy [9] optimizes application service time in respect of user's budget. The *Resource-aware* management policy [10] reduces the scope of resource over provisioning while placing applications on Fog instances and meets their minimum requirements. The *Workload-aware* management policy [11] schedules less compute-intensive applications with high bandwidth requirements in Fog infrastructure as per their deadline constraints. Details of experiment environments, performance metrics and evaluation results are discussed in the following subsections.

A. Experiments on Real Environment

Fig. 4 presents a sample setup of the real Fog environment. We organize the environemnt using FogBus framework [15]. FogBus helps to integrate IoT devices and Fog infrastructure through a dedicated software system and supports the creation of scalable Fog environments. In our real experimental setup, eight different smart phones act as IoT devices. They are connected with an *AMD Dual-Core M320 2.10 GHz 2.00 GB RAM* configured computer which is regarded as an FG. The FG communicates with a cluster of computers that plays the role of FC. Within the cluster, there exists two *Intel Core i7-6700T 2.80 GHz 16.00 GB RAM* and three *Intel Core i7-7700T 3.80 GHz 16.00 GB RAM* configured computers acting as Fog nodes along with an *Intel Core i3-2350M 2.30 GHz 4.00 GB RAM* configured computer performing the duty of FRM. The Fog nodes are capable of hosting twelve different Fog instances through VirtualBox [28]. The instances adapt the bridged networking mode so that they can be accessed by all components within the Local Area Network (LAN). Using NetLimiter [29] software the uplink and downlink speed within Fog infrastructure are controlled and its resource utilization is monitored by Process Explorer [30] software.

Moreover, we profile the execution time of two applications in this environment. One of the applications analyses histogram of an image file whereas another counts the number of words in a text file. We define three different file sizes for their inputs. Each smart phone launches placement requests to the FG for placing these applications in Fog infrastructure with inputs having any of the defined file sizes. Besides, a

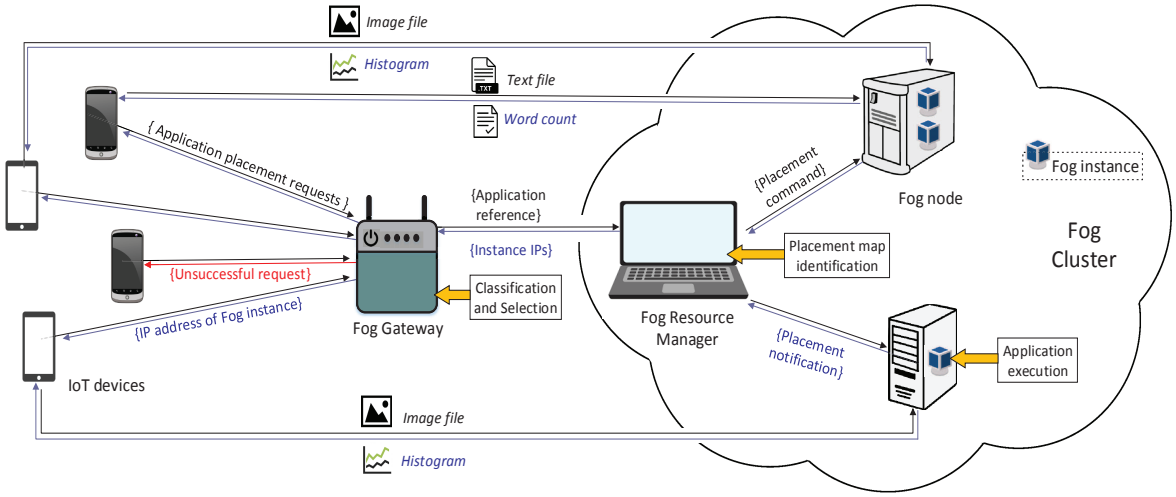

Fig. 4: An illustration of real Fog environment

placement request denotes the data sensing frequency and expected application service delivery time of the associated smart phone. Since application service requirements vary from one request to another, we treat each request as the demand for a separate application. We also enforce the FG to provision at most ten such applications in the Fog infrastructure. Different settings of this environment are listed in Table III.

1) Performance Metrics: The following metrics are used to evaluate the proposed policy in this experimental setup.

- Average Amount of Data Handled (*Avg. ADH*): If an application management policy utilizes the Fog infrastructure extensively, value of this metric increases. It also denotes the lower amount of load sent to other computing infrastructure.
- Average Management Load (*Avg. ML*): It denotes the average CPU usage of FG and FRM while classifying, selecting and identifying application-instance placement map. The balanced Avg. ML between FG and FRM reflects the efficacy of a policy in distributing the application management tasks across the gateway and infrastructure level.
- Average Delay from Request to Placement (*Avg. ADRP*): Lower value of this metric points to the enhanced performance of a policy in reducing waiting of IoT devices while accessing Fog infrastructure services and initiating data processing.

TABLE III: Settings of real Fog environment

Total instances: 12	
CPUs:	6 instances with 1 core 5 instances with 2 cores 1 instance with 4 cores
Bandwidth:	3 instances with 2 MBPS 5 instances with 3 MBPS 4 instances with 4 MBPS
RAM:	7 instances with 2 GB 3 instances with 4 GB 2 instances with 8 GB
Total requested applications: 16	
Allowable applications in Fog	10
Average size of text files (MB)	$S_1 = 0.20$, $S_2 = 0.50$, $S_3 = 0.80$
Average size of image files (MB)	$I_1 = 0.38$, $I_2 = 0.74$, $I_3 = 1.10$
Amount of data per input:	2 applications with S_1 4 applications with S_2 2 applications with S_3 3 applications with I_1 3 applications with I_2 2 applications with I_3
Sensing frequency of phones:	2 applications with 0.25 input/sec 4 applications with 0.50 input/sec 5 applications with 1 input/sec 3 applications with 2 input/sec 2 applications with 3 input/sec
Deadline:	2 applications with 0.40 sec 3 applications with 0.70 sec 4 applications with 1 sec 4 applications with 1.20 sec 3 applications with 1.50 sec

2) Result Analysis: Along with our proposed policy, we implement the basic concepts of Time, Workload and Resource-aware application management policy in the modelled Fog environment. The Time-aware policy applies evolutionary algorithm to determine application-instance map. Compared to Time-aware policy, the proposed policy performs better in improving Avg. ADRP as it conducts low complexity approaches to classify and select applications for Fog-based placement and reduces the dimension of optimization problem significantly. However, the Workload and the Resource-aware policy performs well in terms of Avg. ADRP than all others (Fig. 5). It happens because the Workload-aware policy adapts simplified earliest deadline first and earliest completion time first approaches for placement map identification, and the Resource-aware policy conducts multi-phase sorting and searching method for the similar operation.

Moreover, as the proposed policy explicitly prioritizes applications for Fog-based placement according to their input

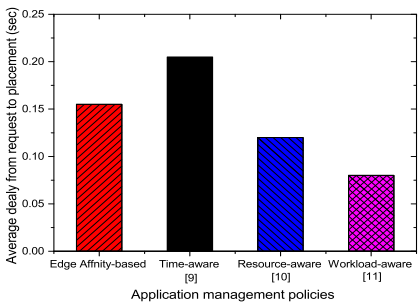

Fig. 5: Avg. ADRP for different management policies

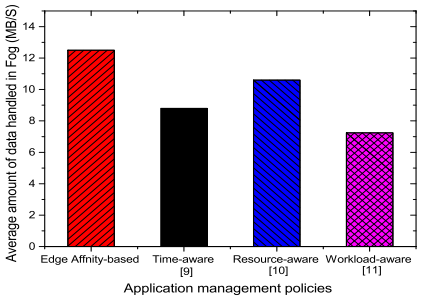

Fig. 6: Avg. ADH for different management policies

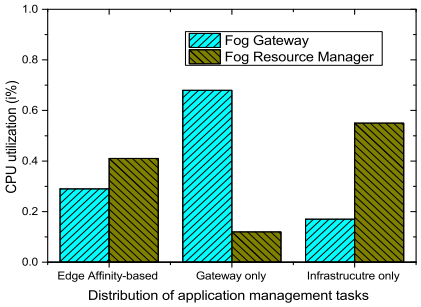

Fig. 7: Avg. ML in different distribution of management tasks

data size, it increases Avg. ADH in Fog infrastructure (Fig. 6). By reducing the scope of resource over-provisioning, the Resource-aware policy also improves Avg. ADH compared to the rest. However, for enhancing application service delivery time and deadline-prioritized placement, the Time and the Workload-aware policy often places applications having small amount data on powerful computing instances. As a result, they fail to increase Avg. ADH like other policies.

In addition, to illustrate the efficacy of our policy in distributing application management tasks, we compare its performance with two more variations namely *Infrastructure only* and *Gateway only*. In Infrastructure only, all management tasks are executed by the FRM whereas in Gateway only, the opposite happens. Nevertheless, our policy facilitates balanced Avg. ML on both gateway and infrastructure while conducting application management tasks (Fig. 7). Hence, it neither increases computational burden on resource poor FGs like Gateway only approach nor overwhelms the FRMs with additional responsibilities as Infrastructure only approach.

B. Experiment on Simulated Environment

Besides the real setup, several experiments are also conducted in iFogSim-simulated [14] Fog environment so that we can demonstrate the large-scale comparisons between our proposed and other application management policies easily. Since real workload is not available for simulating different scenarios in Fog environments, synthetic workload is used for the experiments. It is also considered that the arrival rate of placement requests for different applications and numerical value of their character-driving attributes follow the Poison distribution. Furthermore, there exists a linear relationship between the number of instructions of an application and its input data size. In the simulated setup, if an application is not selected for Fog-based placement, it is forwarded to a Cloud datacentre for execution. The simulation experiments are conducted on an *Intel Core 2 Duo CPU @ 2.33-GHz 2GB-RAM* configured computer. Different simulation parameters used in the experiments are listed in Table IV.

TABLE IV: Parameters of simulated environment

Parameter	Value
Instance:	
Computing capacity	3-7 CPUs
Downlink bandwidth	4-20 MBPS
Uplink bandwidth	2-14 MBPS
RAM	6-10 GB
Processing speed	4000-12000 MIPS
Application:	
Computation requirements	2-5 CPUs
Network requirements	6-12 MBPS
Memory requirements	2-8 GB
Number of instructions	300 - 1300 MI
Input data size	0.300-1.5 MB
Output data size	0.100-1 MB
Service deadline	0.300-1.2 seconds
Sensing frequency of IoT devices	1-8 input/second
Simulation time	200 Seconds
Number of instances	30
Sensing duration of IoT devices	1-4 Seconds
Arrival rate of placement requests	15-35 requests/second

1) Performance Metrics: The performance metrics used in the simulation experiments are listed below:

• Percentage of QoS Satisfied Applications (*Per. QSA*): Increased value of this metric refers to the enhanced performance of management policies in meeting application service delivery deadline. If Y and Z denote the set of deadline satisfied and the set of placed applications in both Fog and Cloud instances respectively, Per. QSA is calculated using Eq. 13:

$$\textit{Per. QSA} = \frac{|Y|}{|Z|} \times 100\% \tag{13}$$

• Average Network Relaxation Time (*Avg. NRT*): Increased value of this metric signifies reduced communication overhead among the instances that consequently decreases the possibility of network congestion. If ζ_p is the set of all placed applications on instance p during the simulation round, Avg. NRT is referred by Eq. 14:

$$\textit{Avg. NRT} = \frac{1}{|P|} \sum_{\forall p \in P} \frac{\sum\limits_{\forall q \in \zeta_p} \frac{1}{\lambda_q} - t^{\iota}_{pq}}{|\zeta_p|} \tag{14}$$

• Average Resource Utilization Ratio (*Avg. RUR*) of Fog instances: Higher value of this metric denotes improved performance of a placement policy in increasing resource utilization of Fog instances. If F is the set of all Fog instances ($F \subset P$), Avg. RUR is determined through Eq. 15:

$$\textit{Avg. RUR} = \frac{1}{|F|} \sum_{\forall p \in F} \frac{\sum\limits_{\forall q \in \zeta_p} \frac{\lambda_q \times \mu_q}{\Lambda_p}}{|\zeta_p|} \tag{15}$$

2) Result Analysis: In this work, the results of simulation experiments are analysed in two phases.

• *Impact of Varying Number of Placed Applications:* The Workload-aware application management policy mainly focuses on delivering application services within the deadline.

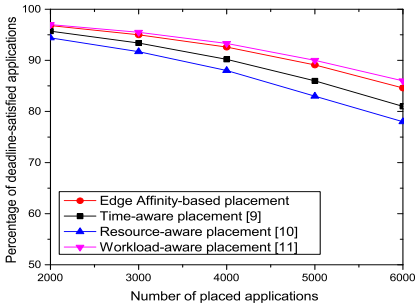

Fig. 8: Per. QSA vs number of placed applications

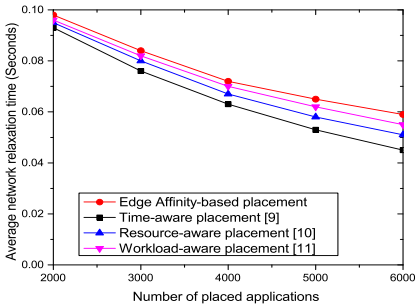

Fig. 9: Avg. NRT vs number of placed aplications

Therefore, for increased number of placed applications, it performs better in terms of Per. QSA than the proposed policy. However, our policy not only considers application deadline but also exploits their input data size and sensing frequency of IoT devices during application placement (Fig.8). Conversely, the Time-aware policy optimizes service time of all applications regardless their deadline criticality and the Resource-aware policy targets to meet the minimum resource requirements of applications without explicitly prioritizing them. Hence, with the increasing number of placed applications in computing environments, these policies fail to achieve the same level of Per. QSA as the proposed policy.

Furthermore, our policy places applications having high frequency of IoT devices and larger data size in Fog instances. Thus, it reduces the overhead of distant communication even when the number of placed applications in computing environments is increasing. Consequently, it helps to offer improved Avg. NRT than others (Fig. 9). Moreover, due to exploiting Fog instances with lower possibility of resource over provisioning and facilitating the applications having high bandwidth requirements, the Resource and the Workload-aware policy

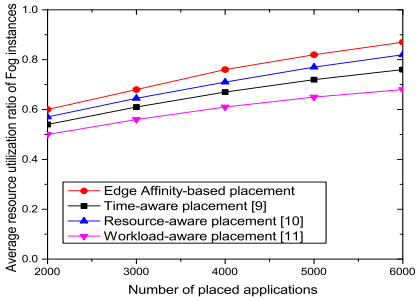

Fig. 10: Avg. RUR vs number of placed applications

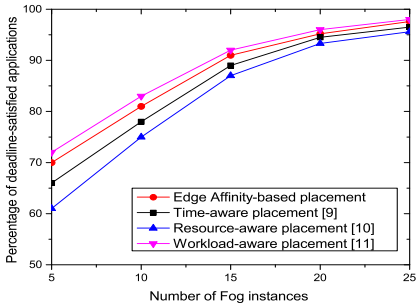

Fig. 11: Per. QSA vs number of Fog instances

perform nearly as the proposed policy. On the other hand, the Time-aware policy fails to improve Avg. NRT like others since it barely considers the data flow characteristics of applications while placing them in Fog infrastructure.

Moreover, the huge amount of data handled by our policy helps to increase Avg. RUR of Fog instances as the number of placed applications increases. It also works in favour of the Resource-aware policy (Fig. 10). However, for executing less compute intensive applications in Fog environments and optimizing application service time without setting any precedence, the Time and the Workload-aware policy often fail to exploit the Fog instances comprehensively. As a result, Avg. RUR degrades for these policies compared to others.

- *Impact of Varying Number of Fog Instances:* As the number of Fog instances increases, the scope of placing applications in proximity of data sources expands. It reduces data propagation delay for a large portion of applications and increases Per. QSA for all application management policies. However, due to prioritizing applications based on their deadline constraints, the proposed and the Workload-aware policy performs better in this case compared to the rest (Fig. 11).

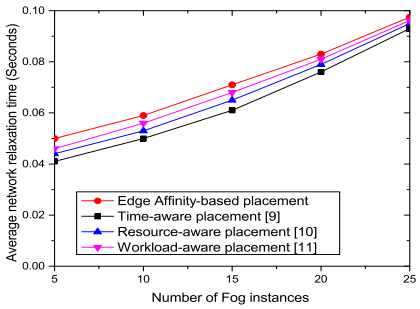

Fig. 12: Avg. NRT vs number of Fog instances

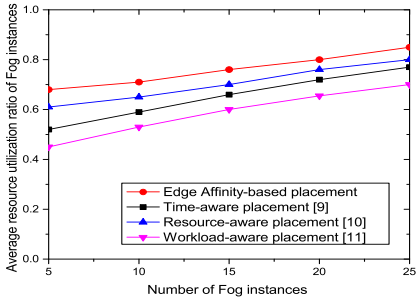

Fig. 13: Avg. RUR vs number of Fog instances

In addition, the increased number of Fog instances resists the transfer of huge amount of data to other infrastructure. Although it increases data exchange rate at the network edge, compared to the reduction in distant communication overhead, it is trivial. Hence, Avg. NRT increases for all policies (Fig. 12). Since our policy and the Workload-aware policy explicitly handle the data flow and bandwidth issues of applications, they perform better than others in improving Avg. NRT with the increment of Fog instances. Furthermore, the proposed and the Resource-aware policy comprehensively engage the increased number Fog instances in executing the applications having high data load and stringent resource requirements. Therefore, the idle time of instances decreases and Avg. RUR enhances for these policies than others (Fig. 13).

VI. Conclusions and Future Work

Multidimensional constraints resist the accommodation of every IoT applications in Fog environments. It urges to determine the competent set of applications for Fog-based placement. In this work, we proposed an application management policy that explores application characteristics in terms of urgency, input size and data flow, and identifies their necessity for Fog-based placement in form of Edge affinity. Edge affinity of an application depends on its service delivery deadline, amount of data to be processed per input and sensing frequency of IoT devices. The proposed policy classifies applications through non-dominated sorting of their Edge affinity and selects a set of applications with stringent QoS requirements for placement in Fog instances. An ILP model also ensures their minimized service time in Fog environments. Performance evaluation conducted in both real and simulated setup illustrate that the proposed policy outperforms others in enhancing QoS, network relaxation and resource utilization.

In future, we plan to apply application characteristics and their driver attributes in enhancing providers profit and user experiences in Fog computing environments.

References

[1] M. Afrin, M. R. Mahmud, and M. A. Razzaque, "Real time detection of speed breakers and warning system for on-road drivers," in *International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)*. IEEE, 2015, pp. 495–498.

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog Computing and Its Role in the Internet of Things," in *First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC '12. ACM, 2012, pp. 13–16.

[3] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: A taxonomy, survey and future directions," in *Internet of everything*. Springer, 2018.

[4] R. Mahmud, K. Ramamohanarao, and R. Buyya, "Latency-Aware Application Module Management for Fog Computing Environments," *ACM Trans. Internet Technol.*, vol. 19, no. 1, pp. 9:1–9:21, Nov. 2018.

[5] 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.

[6] S. Venticinque and A. Amato, "A methodology for deployment of IoT application in fog," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 5, pp. 1955–1976, 2019.

[7] V. Cardellini, G. Mencagli, D. Talia, and M. Torquati, "New Landscapes of the Data Stream Processing in the era of Fog Computing," *Future Generation Computer Systems*, 2019, 10.1016/j.future.2019.03.027.

[8] R. Mahmud, F. L. Koch, and R. Buyya, "Cloud-fog interoperability in IoT-enabled healthcare solutions," in *19th International Conference on Distributed Computing and Networking*. ACM, 2018, p. 32.

[9] H. T. T. Binh, D. B. Son, P. A. Duc, B. M. Nguyen *et al.*, "An Evolutionary Algorithm for Solving Task Scheduling Problem in Cloud-Fog Computing Environment," in *9th Intl' Symposium on Information and Communication Technology*. ACM, 2018, pp. 397–404.

[10] M. Taneja and A. Davy, "Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm," in *IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. IEEE, 2017, pp. 1222–1228.

[11] G. L. Stavrinides and H. D. Karatza, "A hybrid approach to scheduling real-time IoT workflows in fog and cloud environments," *Multimedia Tools and Applications*, pp. 1–17, 2018, 10.1007/s11042-018-7051-9.

[12] N. Verba, K.-M. Chao, J. Lewandowski, N. Shah, A. James, and F. Tian, "Modeling industry 4.0 based fog computing environments for application analysis and deployment," *Future Generation Computer Systems*, vol. 91, pp. 48–60, 2019.

[13] Q. Qi and F. Tao, "A Smart Manufacturing Service System Based on Edge Computing, Fog Computing, and Cloud Computing," *IEEE Access*, vol. 7, pp. 86 769–86 777, 2019.

[14] 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, 2017.

[15] S. Tuli, R. Mahmud, S. Tuli, and R. Buyya, "FogBus: A Blockchain-based Lightweight Framework for Edge and Fog Computing," *Journal of Systems and Software*, vol. 154, pp. 22 – 36, 2019.

[16] T. Choudhari, M. Moh, and T.-S. Moh, "Prioritized task scheduling in fog computing," in *ACMSE 2018 Conference*. ACM, 2018, p. 22.

[17] Y. Nan, W. Li, W. Bao, F. C. Delicato, P. F. Pires, and A. Y. Zomaya, "A dynamic tradeoff data processing framework for delay-sensitive applications in cloud of things systems," *Journal of Parallel and Distributed Computing*, vol. 112, pp. 53–66, 2018.

[18] D. Hoang and T. D. Dang, "FBRC: Optimization of task scheduling in fog-based region and cloud," in *IEEE Trustcom/BigDataSE/ICESS*. IEEE, 2017, pp. 1109–1114.

[19] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski, and P. Leitner, "Optimized IoT service placement in the fog," *Service Oriented Computing and Applications*, vol. 11, no. 4, pp. 427–443, 2017.

[20] X. Xu, S. Fu, Q. Cai, W. Tian, W. Liu, W. Dou, X. Sun, and A. X. Liu, "Dynamic resource allocation for load balancing in fog environment," *Wireless Communications and Mobile Computing*, vol. 2018, 2018.

[21] S. Rehman, N. Javaid, S. Rasheed, K. Hassan, F. Zafar, and M. Naeem, "Min-Min Scheduling Algorithm for Efficient Resource Distribution Using Cloud and Fog in Smart Buildings," in *International Conference on Broadband and Wireless Computing, Communication and Applications*. Springer, 2018, pp. 15–27.

[22] G. Li, J. Wu, J. Li, K. Wang, and T. Ye, "Service popularity-based smart resources partitioning for fog computing-enabled industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4702–4711, 2018.

[23] 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*, vol. 10, no. 6, pp. 2435–2452, 2019.

[24] L. He and J. Walrand, "Pricing and revenue sharing strategies for internet service providers," in *24th IEEE Annual Joint Conference of the IEEE Computer and Communications Societies.*, vol. 1. IEEE, 2005.

[25] L. F. Bittencourt, M. M. Lopes, I. Petri, and O. F. Rana, "Towards virtual machine migration in fog computing," in *10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*. IEEE, 2015, pp. 1–8.

[26] M. Afrin, J. Jin, and A. Rahman, "Energy-Delay Co-optimization of Resource Allocation for Robotic Services in Cloudlet Infrastructure," in *International Conference on Service-Oriented Computing*. Springer, 2018, pp. 295–303.

[27] T. Achterberg, "SCIP: solving constraint integer programs," *Mathematical Programming Computation*, vol. 1, no. 1, pp. 1–41, 2009.

[28] P. Li, "Selecting and using virtualization solutions: our experiences with VMware and VirtualBox," *Journal of Computing Sciences in Colleges*, vol. 25, no. 3, pp. 11–17, 2010.

[29] C. Rodbro and S. Strommer, "Controlling data transmission over a network," Feb. 16 2016, US Patent 9,264,377.

[30] M. E. Russinovich and A. Margosis, *Windows Sysinternals administrator's reference*. Microsoft Press, 2011.