Journal of Network and Computer Applications 165 (2020) 102674

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

QoS-aware service provisioning in fog computing

Faizan Murtaza [a], Adnan Akhunzada [a,b,*], Saif ul Islam [c], Jalil Boudjadar [d], Rajkumar Buyya [e]

[a] DTU Compute - Department of Applied Mathematics and Computer Science, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark
[b] RISE SICS Vasteras AB, Sweden
[c] Department of Computer Science, KICSIT, Institute of Space Technology, Islamabad, Pakistan
[d] ECE, Department of Engineering, Aarhus University, Denmark
[e] Cloud Computing and Distributed Systems (CLOUDS) Laboratory School of Computing and Information System, The University of Melbourne, Parkville, Australia

ARTICLE INFO

Keywords:
Fog computing
Cloud computing
Quality of service
LRFC
Quality of experience
Internet of everything

ABSTRACT

Fog computing has emerged as a complementary solution to address the issues faced in cloud computing. While fog computing allows us to better handle time/delay-sensitive Internet of Everything (IoE) applications (e.g. smart grids and adversarial environment), there are a number of operational challenges. For example, the resource-constrained nature of fog-nodes and heterogeneity of IoE jobs complicate efforts to schedule tasks efficiently. Thus, to better streamline time/delay-sensitive varied IoE requests, the authors contributes by introducing a smart layer between IoE devices and fog nodes to incorporate an intelligent and adaptive learning based task scheduling technique. Specifically, our approach analyzes the various service type of IoE requests and presents an optimal strategy to allocate the most suitable available fog resource accordingly. We rigorously evaluate the performance of the proposed approach using simulation, as well as its correctness using formal verification. The evaluation findings are promising, both in terms of energy consumption and Quality of Service (QoS).

1. Introduction

In the emerging Internet of Everything (IoE) paradigm, billions of devices are being connected to the Internet. The number, types and nature of Internet-connected devices will also increase for the foreseeable future. For example, it was reported that more than 50 billion devices will be linked to the Internet by 2020 (Mohan and Kangasharju, 2017), with an estimated market worth of $7.1 trillion (Wortmann and Flüchter, 2015). Quality of Experience (QoE) is one of several key metrics for the IoE users (Mahmud et al., 2019). To deal with limitations inherent in a cloud computing environment (e.g. privacy of users sourcing the data to the cloud, particularly hosted in an overseas jurisdiction, and performance issues such as latency), there has been attempts to 'push' the computing to the edge of the network via fog nodes (Choo et al., 2018; Osanaiye et al., 2017). Hence, the fog computing paradigm has been emerged to handle time/delay-sensitive IoE applications (Deng et al., 2016; Bitam et al., 2017; Toor et al., 2019a; Intharawijitr et al., 2016; Zeng et al., 2016; Dar et al., 2019).

As shown in Fig. 1, only low-latency computational tasks are processed by the fog nodes but the computationally demanding tasks will still be routed to the cloud servers for processing (Mohan and Kangasharju, 2017). Fog nodes can be any computing devices that has some computational capabilities in terms of data management, analytics, computation, storage and networking, and examples range from an intelligent home assistant (e.g. Amazon Echo) in a smart home to a military vehicle in an Internet of Battlefield Things environment. The connections between fog nodes and the cloud can be wired and/or wireless (Dastjerdi and Buyya, 2016) (see Fig. 2).

While there are many challenges associated with a fog computing environment (Ficco et al., 2017), we will only focus on one of these challenges. Specifically, we will study the task scheduling problem for better fog resource management, particularly with the exponential

* Corresponding author. DTU Compute - Department of Applied Mathematics and Computer Science, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark.
E-mail addresses: adnak@dtu.dk (A. Akhunzada), akhunzadaadnan@gmail.com (A. Akhunzada).

https://doi.org/10.1016/j.jnca.2020.102674
Received 2 February 2019; Received in revised form 29 October 2019; Accepted 21 April 2020
Available online 16 May 2020
1084-8045/

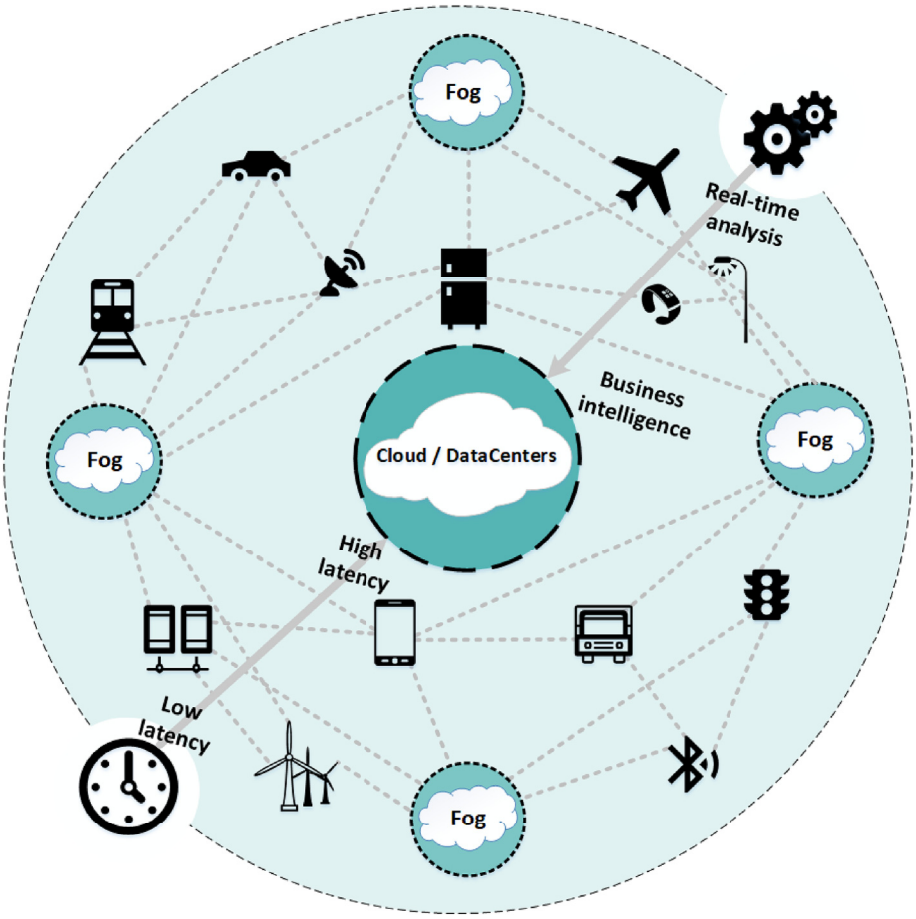


Fig. 1. An example fog-cloud environment.

increase in the number of IoT devices while achieving a satisfactory QoE.

The existing proposed techniques in fog computing lacks the adaptive and intelligent behavior. They only consider the direct communication of end devices preferably with closest fog nodes. However, this type of architecture only supports individual node queue level task scheduling. Further, it is quite complex to implement adaptive and intelligent learning-based task scheduling in this kind of architecture. Subsequently, we cannot take the real advantage of heterogeneity of fog nodes surrounded by a set of IoT devices. In order to facilitate intelligent and adaptive task scheduling policies in this dynamic and heterogeneous environment, we need a smart layer (Aazam and Huh, 2014) between end devices and fog nodes that should have three main capabilities: 1) the ability to define whether the incoming request should be served by a cloud or a fog node, 2) the capability to schedule the incoming task to most appropriate fog node among the available fog devices, and 3) and for efficient task scheduling this layer should have the functionality that extensively implements adaptive and intelligent learning-based task scheduling. In this paper, we propose an intelligent and adaptive task based learning technique for optimal task scheduling in a fog computational paradigm. The key contributions of the paper are summarized as follows:

- A proposition and implementation of a smart layer between IoE/IoT devices and Fog nodes.
- We propose a QoS-Aware approach (hereafter referred to as the learning repository fog-cloud - LRFC). The proposed service has been provisioned at multiple geographically distributed gateways deployed among IoE devices and their corresponding Fog nodes proximity. Consequently, making our proposed scheme highly scalable, and thus reliving potential performance bottlenecks. Finally, our proposed deployment model significantly suits nearly all existing and futuristic environments (i.e., IoT, IoE, smart X (smart city, smart grid, smart building, smart forest etc.)).
- For a comprehensive and unbiased comparison, we implement the state-of-the-art task scheduling policies on our proposed smart layer.
- We thoroughly evaluate the performance of the proposed approach using simulation, and prove its correctness through formal verification. The proposed approach shows promising results in terms of processing delay, overall network propagation time and power consumption.

The remainder of this paper is organized as follows. In the next section, we present the background of the proposed approach and relevant works. Section 3 describes the complete methodology (i.e., system model and algorithms) of the proposed approach. The formal verification of our proposed scheme is detailed in Section 4. Performance evaluation is comprehensively elaborated in Section 5. Finally, Section 6 concludes the paper along with thoughts for future work.

2. Related work

The section briefly explains the background and related research.

2.1. Learning based approaches

Rule-based learning is the simplest form of artificial learning (Núñez et al., 2006). In general, rules are expressed in IF-THEN condition, and they can be simple or multi conditional IF-ELSE statement (Ligêza, 2006) (Keshtkar et al., 2014). It has been shown that rule-based learn-

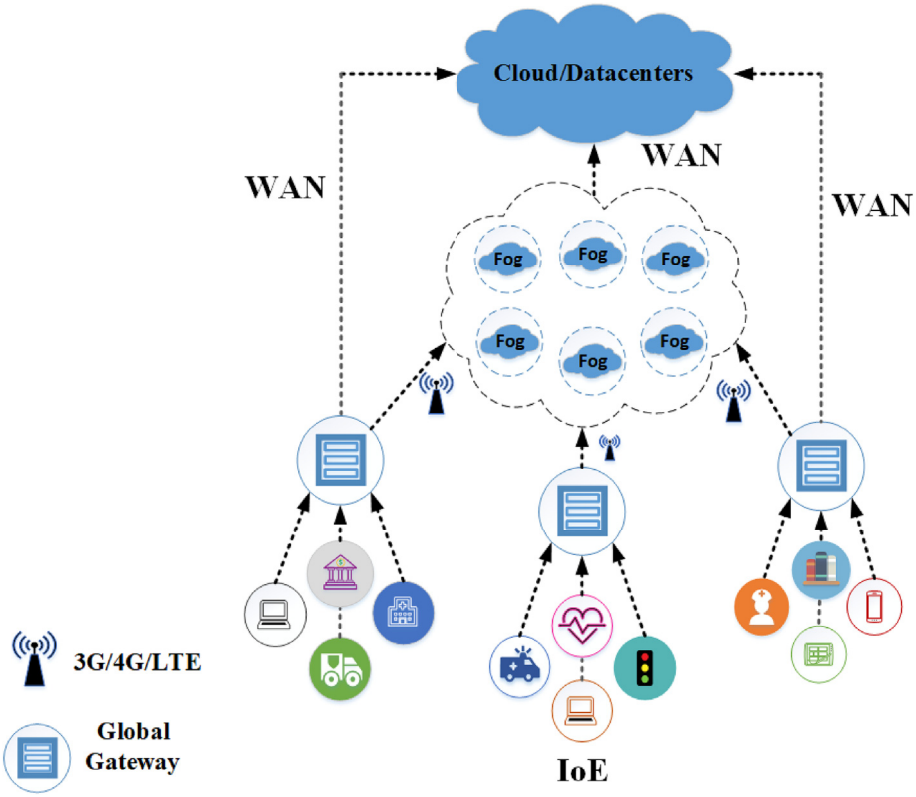


Fig. 2. High level architecture.

ing can be applied in a wide range of applications, such as to reduce and predict the energy gap between predicted and actual energy consumption in buildings (Yuce and Rezgui, 2017), big data classifications problems (Elkano et al., 2017), user behavior classifications (Alrashed, 2017), developing decision support system for risk assessment and management strategies in distributed software development (Aslam et al., 2017), and so on.

Case-based Learning (CBR), a lazy learning approach, works well where rich structured knowledge is not available earlier (e.g. in autonomic computing) (Aha, 1991) (Khan et al., 2011). For learning purpose, it involves the processing of a set of cases to train the system and predict values from these cases (and prior knowledge). In (Amin, 2017), for example, the authors proposed an architecture for sharing of experience using an agent-based system architecture layout (SEASALT). The latter works with diverse data repositories to maintain, retrieve, adopt and retain cases. Similarly, in (Brown et al., 2017), the authors proposed a temporal CBR for diabetes insulin by looking at prior events such as blood pressure level, physical activities and carbohydrate ingestion. Additionally, similarity metrics like Euclidean distance have also used with CBR for pattern classification (Yan et al., 2017).

Hybrid approaches can be more effective than either a rule- or case-based learning approach to solve complex problems (Van Den Bossche et al., 2010), (Prentzas and Hatzilygeroudis, 2003). In (Kumar et al., 2009), the author used a hybrid approach for domain-independent clinical decision support in a hospital's intensive care unit. The potential for a hybrid approach was demonstrated in a medical diagnosis system by the authors in (Sharaf-El-Deen et al., 2014) and (Tung et al., 2010).

2.2. Task scheduling in fog-cloud environment

Efficient task scheduling in fog computing to maximize resource management is one of several research focuses in recent years (Mouradian et al., 2017; Bitam et al., 2018). We mainly classify the task scheduling approaches into two main categories in Fog-Cloud environments.

2.2.1. QoS-aware task scheduling in fog-cloud environment

In (Bitam et al., 2017), the authors proposed a bio-inspired optimization approach (i.e., Bees Life Algorithm (BLA)), seeking to address the job scheduling challenge in a fog computing environment. The approach is based on the optimized distribution of a set of tasks among fog nodes to deal with users' excessive requests to computational resources. This approach also seeks to minimize energy consumption and CPU execution time. In a different work, the authors of (Intharawijitr et al., 2016) proposed three different strategies for optimum resource utilization. Firstly, a random methodology is used to select the fog nodes to execute tasks upon arrival. Secondly, the focus will be on the lowest latency fog devices. Finally, the fog resources having the maximum available capacity should be primarily considered. The three proposed policies were then evaluated using a mathematical model. The authors in (Zeng et al., 2016) introduced a joint optimization task scheduling and image placement algorithm, which is designed to minimize the overall completion task time for better user experience. The first sub-problem investigated is to balance the workload both on client devices and computational servers. The second sub-problem investigated is on the placement of task image on storage servers, and the final sub-problem is to balance the input-output interrupt requests among the storage servers. The research (Bittencourt et al., 2017) discusses applications scheduling in the fog computing process and focuses on the influence of user mobility on application performance. Different policies have been used in scheduling, when different application requests arrive these policies decides to execute them on cloudlet or on the cloud. Policies include concurrent policy, in this policy all request received at the cloudlet or allocated to cloudlet without measuring the usage. First Come First Server (FCFS) policy works in a traditional way serving the requests on arrival until it consumes all

resources. The third was Delay-priority policy in which all request that requires lower delay was scheduled first. In (Deng et al., 2016), the authors proposed a workload allocation framework to balance the computational latency and power consumption in a fog-cloud environment. Likewise (Deng et al., 2016), the authors in (Zeng et al., 2016) studied the trade-off between power consumption and computational latency in fog. The approach is based on the convex optimization technique such as interior-point method (He et al., 2014), the mixed integer nonlinear programming problem using generalized benders decomposition (Li and Sun, 2006), and the Hungarian method (Kuhn, 2010) to address the problem in a fog-cloud environment. Furthermore, A. Toor et al. (2019b) proposed and evaluated energy and performance aware fog computing scheme using Dynamic Voltage Frequency Scaling (DVFS) technique while utilizing the green renewable energy resources. Similarly, effective resource utilization and computing service of delay sensitive applications were considered in (Song et al., 2016). The study proposed a graph representation base system and a task-oriented dynamic load balancing algorithm that maps the physical resources to virtualized resources. Each resource is represented by a node and has a certain capacity. On the arrival of a new fog node, the algorithm reallocates the load in its nearby neighborhood to maintain the balance and accounting task distribution degree and the links among nodes. A reverse strategy was adopted to remove edges not having sufficient resources.

2.2.2. QoE-aware task scheduling in fog-cloud environment

QoE refers to the user experience towards the various service aspects. It considers user needs, perceptions and intentions regarding provided services (Mahmud et al., 2019). Our published work (Mahmud et al., 2019) mainly focuses on QoE-aware task scheduling in Fog computing environment. The article uses a learning fuzzy logic based approach to enhance QoE in hierarchical, distributed and heterogeneous Fog-IoT environment. The technique follows the prioritized application placement to the suitable Fog servers using fuzzy logic models. Similar to (Deng et al., 2016), the authors in (Oueis et al., 2015) also studied load balancing while focusing on quality of experience (QoE). The proposed algorithm uses clustering in order to meet the computation demands and minimize the power consumption. The first in first out (FIFO) mechanism is used for task scheduling and earliest deadline first (EDF) policy is used for resource allocation. The authors in (Aazam, 2015) and (Aazam and Huh, 2015) considered multiple factors and formulated resource management on the basis of changing the relinquishing probability of the customer, service type, service price. However in (Aazam and Huh, 2015), resources were taken into account on the nature of devices. A loyalty based task scheduling model, a service-oriented resource management model to perform efficient and fair management of resources for IoT deployment, was proposed that incorporates the user's history of resource usage to increase the fairness and efficiency, when the resources were actually consumed.

Table 1 summarizes the literature discussed in this section.

In today's dynamic and heterogeneous environment necessitates a smart layer between end devices and fog nodes to encourage intelligent and adaptive task scheduling approaches. The main capabilities of the smart layer should include: a) the capacity to define whether the approaching request ought to be served by a cloud or a fog node, b) the layer should be able to assign the incoming task to most fitting fog node among the accessible fog nodes in a geographical proximity; and c) to extensively implement adaptiveness and learning-based intelligence for varied task scheduling processes.

3. Methodology

The methodology of the proposed scheme is detailed below.

3.1. System model

In this paper, we have designed the complete architecture of the Fog Cloud system as shown in 2. The architecture contains three layers. The first layer comprises of several IoE devices generating N number of requests. The second layer includes geographically distributed gateways deployed among the proximity of IoE devices and their corresponding Fog nodes. Provisioning of our proposed service at varied geographically distributed gateways makes our technique highly scalable and thus avoiding performance bottleneck. The proposed deployment model suits the existing and futuristic environments (i.e., IoT, IoE, smart X (smart city, smart building, smart forest etc.)).

3.2. Learning repository fog-cloud

To ensure effective job scheduling, we propose an intelligent and adaptive approach, named, Learning Repository Fog-Cloud (LRFC) that is a soft solution deployed at various gateways in the second layer. The basic sequence and operations of our proposed system are shown in Fig. 3. The sequence starts with the generation of asynchronous tuples to the LRFC layer. Jobs are decomposed into tasks in second step. Further, the Learning Repository creates its Meta in step three. In step four, best fitted fog servers are selected to serve the jobs. Hence, the LRFC schedules the task either for Fog or Cloud (i.e., if no suitable fog server is found or all the fog servers are completely occupied, then the tasks will be sent to cloud for further processing) in step five. In step six, Fog executes the task and returns the response details to LRFC layer; whereas, the cloud executes the received requests and returns response to LRFC layer in step seven, respectively. LRFC receives the response and updates the information iteratively in step eight. Finally, the results are generated accordingly.

3.3. An adaptive and intelligent task scheduling approach

We propose a hybrid approach based on the idea of both rule-based learning and case-based reasoning to produce efficient results. For this purpose, a learning repository is created that stores the particulars of each incoming task such as tuple identification (ID), tuple type, and the information of the resource where the task is served. Moreover, it also maintains the information such as propagation time, execution time, energy consumption of the tuple at a specific resource. The tasks are scheduled to the available resources based on the information stored in learning repository. Additionally, the selection of the service type (i.e., Fog, Cloud) is also made through the learning repository information. The learning repository is regularly updated, and the scheduling decisions are made accordingly. We have a tp_{total} total number of tuples and fg_{total} total number of fog servers. Initially, we are creating learning repository. Five percent of the tp_{total} are used for training our proposed approach. The remaining ninety-five percent (95%) of tp_{total} is used for testing purpose. Algorithms 1 and 2 provide a detailed and self-explanatory description of our proposed approach. The learning repository storage F_S and ${}_CS$ are created for Fog and Cloud, respectively. The initial tuples used for training are abbreviated as tp_{ini}. The tuples used for are testing are presented as tp_{final}. The *SR* stands for Storage Repository. Whereas, D_T represents the type of IoT job. Internal Processing Time *ITP* is the time that a tuple/job takes for processing using a resource of Fog or Cloud. The link propagation time is abbreviated as PT_L. In the LR_{exe} process, the selection of fog server is decided on the distance between the request generated from and the server location where it is physically placed. The nearest server is selected with its server id fg_{id}. The fg_{id} of the serving server will be saved in *LR* storage as well. In the case of cooperation, if the selected server fg_i is busy then the tuple will wait in the queue or will be sent to the Cloud.

Table 1
Summary of related work.

Ref	Year	Problem	Methodology	Pros	Cons
Mahmud et al. (2019)	2019	QoE in the hierarchical, distributed and heterogeneous Fog-IoT environment	Prioritized application placement to the suitable Fog servers using fuzzy logic models	QoE-aware application placement	Energy consumption is not evaluated
Bittencourt et al. (2017)	2017	Scheduling in the fog computing.	Concurrent, FCFS and DelayPriority scheduling policies.	CPU execution time and delay	Non cooperative simulation environment
Intharawijitr et al. (2016)	2016	Computational and communication delays in high load condition.	5G fog-based infrastructure is simulated.	Communication latency, Computing latency.	Simulated on partial fog computing system.
Bitam et al. (2017)	2017	Higher traffic and efficiency.	Bio-inspired optimization approach for job scheduling, Bees Life Algorithm (BLA)	CPU execution time, Allocated memory in term of service cost	Static execution of jobs, considered limited jobs.
Zeng et al. (2016)	2016	Task scheduling, resource management, and I/O together formulated as a mixed-integer nonlinear programming problem.	Joint optimization task scheduling and image placement algorithm.	Resource Management, Power consumption, Computational latency	Large-scale requests are ignored during task assigning.
Deng et al. (2016)	2015	Optimal resource allocation between fog and cloud.	Convex optimization technique (interior point method), Bender decomposition, and Hungarian method.	Power consumption, Computational latency	Centralized fog computing architecture
Song et al. (2016)	2016	Resource utilization and computing service of delay sensitive applications	Graph representation base, task-oriented dynamic load balancing algorithm that maps the physical resources to virtualized resources	Network flexibility, Dynamic load balancing	High load balancing complexity
Oueis et al. (2015)	2015	QoE in terms of fog load balancing.	Low complex task scheduling algorithm	Energy efficiency, Task latency, Power consumption	High complexity for large scale fog architecture.
Aazam (2015)	2015	Resource management on the basis of changing relinquish probability of the customer, service type, service price, and variance of the relinquish probability	Loyalty base task scheduling algorithm	Resource management	Scalability
Aazam and Huh (2015)	2015	Resource allocation	Loyalty base task scheduling model, a service-oriented resource management	Resource management, Performance measure	The large-scale request is ignored.
Cardellini et al. (2016)	2015	Exploiting local resources in data stream processing	An extension to storm open source data stream processing as QoS-aware scheduler.	Network latency, Inter-node traffic	Availability of applications, Not suitable for composite fog computing

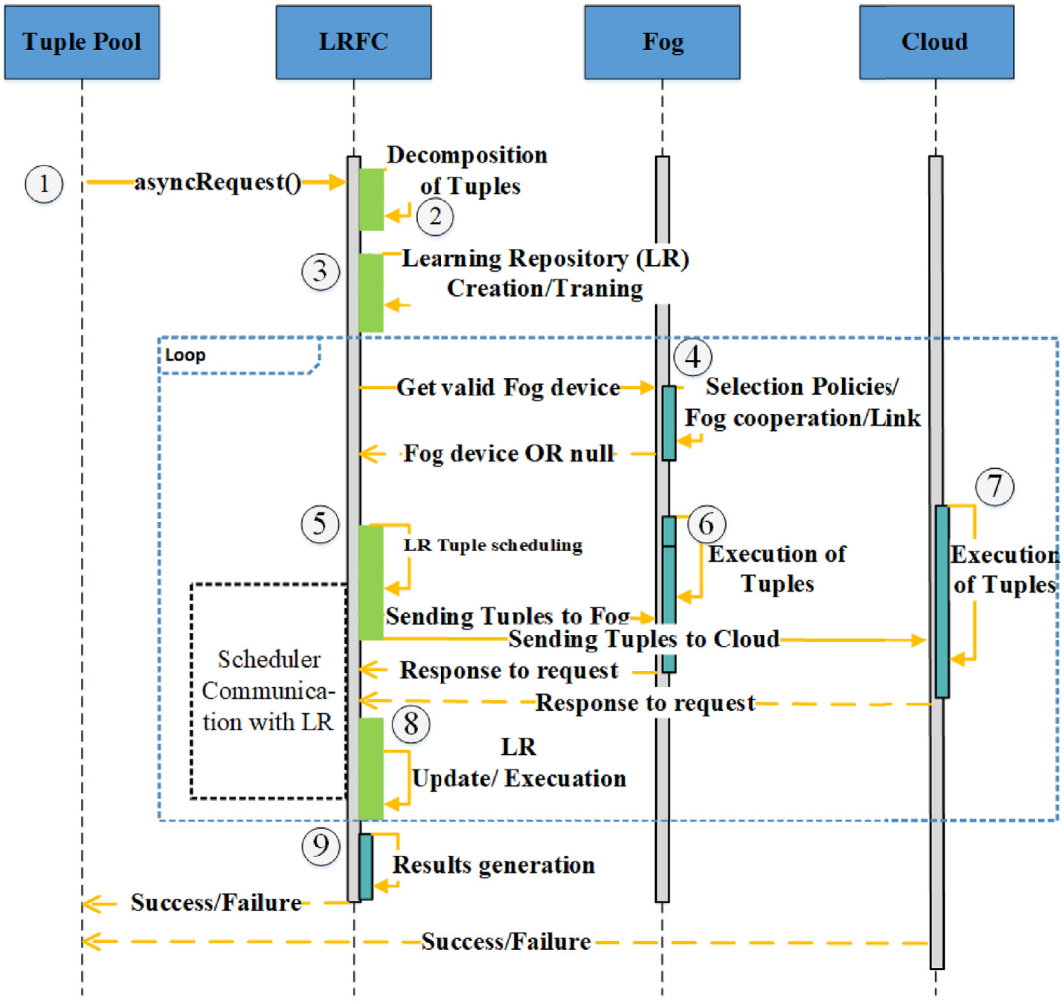


Fig. 3. Proposed system sequence diagram.

Algorithm 1 Learning Repository Fog-Cloud Approach

procedure LRFC(tp_{total}, fg_{total})
  C_T = [*true*, *false*]
  S_t = $\{S_1, S_2, S_3\}$
  D_c
  C_o = {*true*, *false*}
  G_w = {*true*, *false*}
  F_s, C_s ▷ storage for fog and cloud
  $tp_{ini} = (tp_{total} \times 5/100)$ ▷ getting 5% to initial execute
  $tp_{final} = (tp_{total} \times 95/100)$ ▷ 95% of remaining
  split = Splitting tp_{ini} *into* 5 *chunks*
  for ($j = 0$; *to* j = $split_{length}$) **do**
    for (t_i *in* split [*j* **do**
      Rd_{num} = [0, 1]
      if (Rd_{num} = 0) **then**
        fg_i = *exec FOGSIM*($fg_{total}, t_i, C_T, 0$)
      else

(*continued on next page*)

Algorithm 1 *(continued)*

        S_i = get service from S_t
        SendToCloud(t_i, *false*, S_i, D_c)
        temp = {$t_i.D_T, t_i.ID, t_i.ITP, t_i.PT, D_c.Name, t_i.Link$}
        C_s + = *temp*
      end if
    end for
    j + +
  end for
  for each: (t_i *in* tp_{final}) **do**
    F_{Fs} = getting t_i from F_s on the basis of D_T and order by $t_i.IPT \wedge t_i.PT_L$
    C_{Cs} = getting t_i from C_s on the basis of D_T and order by $t_i.IPT \wedge t_i.PT_L$
    Boolean f, c
    f = set true if F_{Fs} is null
    c = set true if C_{Cs} is null
    if (F_{Fs} is null *OR* C_{Cs} is null) **then**
      if (f) **then**
        exec *FOGSIM*($fg_{total}, t_i, C_T, 0$)
      end if

4. Formal verification of the proposed scheme

In this section, we analyze the efficiency of our scheduling algorithm in a formal way. To achieve that goal, we have used Uppaal

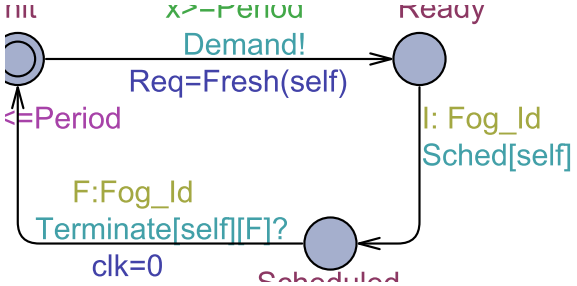


Fig. 4. Template of the IoT device.

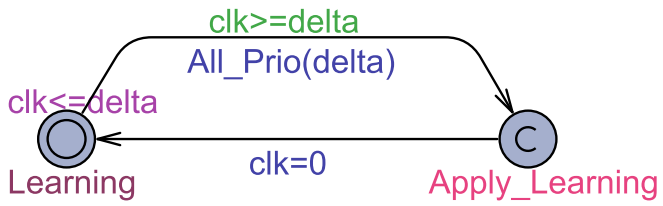


Fig. 5. Template of the learning process.

timed automata to describe the behavior or IoT devices and scheduling algorithm. Then used the underlying model checker to verify a set of derived properties. Given the rich expreviness of Uppaal formalism, the behavior of IoT devices, fogs and algorithm can be constrained with time attributes such as the response time of the scheduler to acquire a request, minimum/maximum time interval between two requests from the same IoT device, time to communicate with a fog, etc.

Request types describe the combination of the resource types that an IoT device can request. A request type can include a single resource (e.g computation resource), or a combination of different resources (e.g computation and storage resources). We use ReqT to denote a request type. A request R targets a set of resources each with a different amount, e.g R=(Compute = 500, RAM = 12). We use notation $|R^i|$ to refer to the amounts of individual resources of the request R. As example, $|R^1| = (Compute, 500)$.

Fig. 4 shows a parameterizable model of IoT devices. A device behavior is initially at location Init, waiting for certain time interval before issuing a request. The request type and budget are randomly generated using function *Fresh()*. While performing a request, the IoT device synchronizes with the scheduler using event *Demand!* The IoT device waits then to be scheduled to a fog at location *Ready*. Whenever scheduled, the device waits until the request is fully satisfied upon event *Terminate[]?* then it can start another request. Function *Expired(Req)* calculates when a request gets satisfied, given the performance characteristics of the fog and the budget requested by the IoT device.

To simulate the learning process, we use dynamic priorities for the allocation of fogs to different request types. The priority to allocate a fog F to a request type changes on the fly according to the dynamics of the system. Initially all priorities are set to zero (*Prio(ReqT,F,0)=0*), meaning that there is no preference in allocating a given fog to a given request type. For a given request type *ReqT*, if a fog F has recently been used many times to satisfy requests of type *ReqT* then the priority to allocate F to the given request type gets increased over time. Otherwise, the priority will be decreased on each time interval of length δ if F has not been allocated to any request of the concerned type during the last δ time interval.

$$Prio(ReqT, F, t)$$

$$= \begin{cases} \text{Prio(ReqT, F, t} - \delta) + + & \text{if } \exists\ (R_1, .., R_n) \in \text{ReqT}, \\ & \exists\ t' \in [\text{t} - \delta, \text{t}], \\ & \wedge\ \text{Allocate}(R_i, \text{F}, t') = \text{True} \\ \text{Prio(ReqT, F, t} - \delta) - & \text{Otherwise} \end{cases}$$

Fig. 5 shows the learning model where for each time interval delta, function *All_Prio(delta)* calls the aforementioned function *Prio()* for all potential fogs and request types as parameters.

When receiving a request from an IoT device, our scheduler considers first fogs having high priorities for the allocation to that request type. If two fogs have the same priority to serve a request and both have sufficient available resource budget, then the scheduler considers the neighborhood attribute where the closer fog gains the allocation. The allocation considers the resource availability at the fog level, the learning-based priority and the neighborhood. The neighborhood attribute is calculated based on the geo-location of the device and fog.

Algorithm 2 Learning Repository Fog-Cloud Approach - Part 2

```
        if (c) then
          SendToCloud(t_i, false, S_i, D_c)
          temp = {t_i.D_T, t_i.ID, t_i.ITP, t_i.PT, D_c.Name, t_i.Link}
          C_s + = temp
        else
          C_cost = C_Cs.IPT + C_Cs.PTL
          F_cost = F_Fs.IPT + F_Fs.PTL
          if (C_cost >= F_cost) then
            exec FOGSIM(fg_total, t_i, C_T, F_cost.fg_Id)
          else
            SendToCloud(t_i, false, S_i, D_c)
            temp = {t_i.D_T, t_i.ID, t_i.ITP, t_i.PT, D_c.Name, t_i.Link}
            C_s + = temp
          end if
        end if
      end if
    end for
  end procedure
  function FOGSIM (fg_total, tp_i, C_t, F_cost.fg_Id)
    for each: (fg in tp_final) do
      if (fghasPower() && hasResources()) then ▷ for requested
  tuple ti with fg_id
        CalcDis_MinProxity()
        fg_i = fg
        if (fg_i = null &&C_o = true) then
          CalcDis_MinProxity()
          fg_i = fg
        end if
      end if
    end for
    temp = {t_i.D_T, t_i.ID, t_i.ITP, t_i.PT, fg_i.ID, t_i.Link}
    F_c + = temp
  end function
```

The following function is used to show how to check whether a fog F satisfies a given request R issued at time instant t.

$$Satisfy(R, F, t)$$

$$= \begin{cases} \text{True} & \text{if } \forall - R^i - \in \text{R budget}(|R^i|, \text{t}) < \text{budget}(\text{F}, |R^i|, \text{t}) \\ \text{False} & \text{Otherwise} \end{cases}$$

Function $budget(X, t)$ returns the amount requested by R for a resource type X at time instant t. We overload this function to return the available amount of a given resource type ($|R^i|$) in a fog at time instant t,

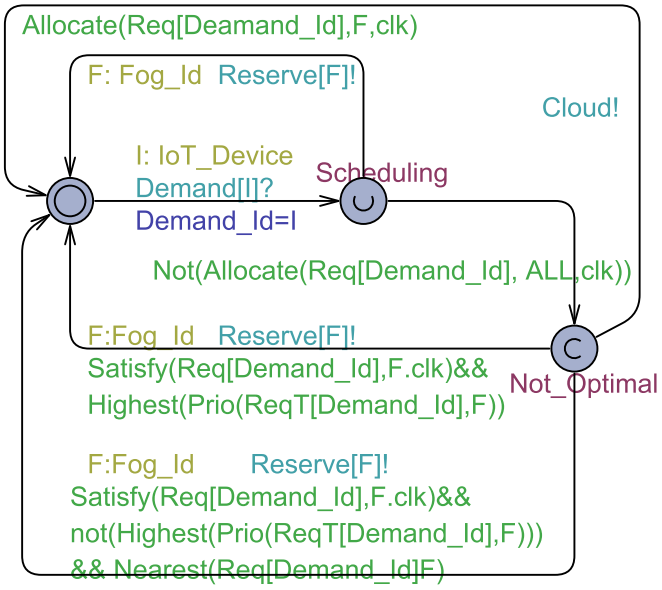


Fig. 6. Template of the scheduler model.

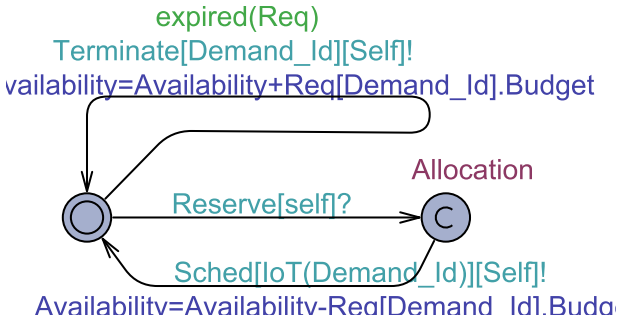


Fig. 7. Template of the fog model.

simply writing $budget(F, |R^i|, t)$.

The allocation of a fog to satisfy a request is performed via the following function:

$$Allocate(R,F,t) = \begin{cases} \text{True} & \text{if Satisfy(R, F, t)} \wedge \forall\ F_j\ \text{Satisfy}(R,F_j,\text{t}) \Rightarrow \\ & \text{not (Prio(R, F, t)} < \text{Prio(R}, F_i, t) \wedge \\ & \text{Near}(F_i, \text{IoT(R))} < \text{Near(F, IoT(R)))} \\ \text{False} & \text{Otherwise} \end{cases}$$

Function $IoT(R)$ returns the actual IoT device that performed request R. So that we calculate the neighborhood $Near(F_i, IoT(R))$ of the request issuer to the fog.

Fig. 6 shows our scheduler model. Basically, whenever a device issues a request it asks the scheduler to reserve a fog to it upon the event *Demand[]?* If there exists a fog satisfying function *Allocate()* for the given request, the fog will be immediately reserved using event *Reserve[]!* from location *Scheduling*. If such a fog does not exist, at location Not_Optimal, the scheduler searches a fog that satisfies the request while being the most used one to serve such a request type recently. Otherwise, the scheduler searches a fog which is the nearest one to the IoT device. If none of the cited options exists, the scheduler mediates the request to the cloud.

When a request processing terminates, the corresponding fog resources are released and become available. We omit describing the termination function as it is very trivial.

Fig. 7 shows the fog model. It is simple and consists in receiving a *Reserve[]* event from the scheduler, synchronizing with the requesting IoT device to start supplying resources. The fog updates the availability of its resources according to the current request budget, both when starts and terminates.

The efficiency property we have formally analyzed using model checking is the following:

Property 1. *Each request, from an IoT device, is always satisfied by the nearest fog having sufficient budget and an experience to satisfy such type of requests. Formally, we write:*

$$\forall R\ Allocate(R,F_j,t) \Rightarrow \neg(\exists\ F_k\ \ | Satisfy(R,F_k,t) \wedge Near(F_k, IoT(R)) < Near(F_j, IoT(R)) \wedge Prio(R,F_i,t) < Prio(R,F_k,t)$$

Table 2
Cloud services considered in the simulation setup.

bf Service	RAM (MB)	Storage (MB)	MIPS	Data Rate (kbps)
Service 1	8192	5000	9000	500
Service 2	16,384	10,000	18,000	500
Service 3	32,768	15,000	27,000	500

5. Performance evaluation

The simulation environment is set up using CloudSim (Calheiros et al., 2011) and iFogSim (Gupta et al., 2017). The arguments on why using CloudSim and iFogSim in fog/edge computing environments has been detailed in (Ficco et al., 2017). Modelling of environment is inspired by Azure Cloud Service (Chappellet al., 2008), and Amazon S3 service (Palankar et al., 2008). Analysis is performed based on relevant parameters such as data generation from IoTs (Chandio et al., 2014) data type, internal processing time, total processing time, queue delay, propagation delay, power consumption, available resources and resources required by a tuple, cooperation of fog nodes, and distance measurement in kilometers between nodes. The cloud services in simulation setup are presented in Table 2. Moreover, cloud data centers considered in simulation environment are shown in Table 3. Furthermore, the detail about dataset, deployment of fog and IoE devices, specifications of fog servers and performance evaluation metrics is given below:

5.1. Dataset

Fig. 8 presents the categories of the dataset used in the simulation (Iot-compute-dataset and http, 2019). It is a synthetic dataset. We have considered 30 and 50 thousand tuples to perform the experiments. The x-axis shows the different types of tuples and y-axis presents the number of tuples. This dataset is used to evaluate the performance of the proposed scheme.

5.1.1. Dataset characteristics

Dataset consists of multiple tuples. These tuples contains various properties such as - size, bandwidth, MIPS, and memory. Where, the tuple size refers to the required size of the tuple. It is important in terms of memory consumption at fog server and processing is performed based on the tuple size. The bandwidth of a tuple defines the bandwidth required by the tuple to reach its destination fog server. Every tuple has its specific bandwidth requirements and should be handled accordingly. Tuple MIPS refers to the processing requirements of a tuple to be executed at fog server. The memory of a tuple exhibits the memory required (in megabytes) by a tuple at a fog server. The dataset contains the job of heterogeneous nature having the information - name of the tuple, Tuple ID, tuple size, MIPS (required by a tuple), bandwidth, location (coordinates), type of IoE job (for instance, textual tuple of small

Table 3
Cloud data-centers in simulation setup.

DC	Geo Location	Memory (MB)	Storage (MB)	MIPS	BW (kbps)	Arch	OS	Status
USA Data-center	37.422421,-22.0866703	51,200	1000 k	500 k	50 k	x86	Linux	Live
Singapore Data-center	1.277911, 103.849662	51,200	1000 k	500 k	50 k	x86	Linux	Live

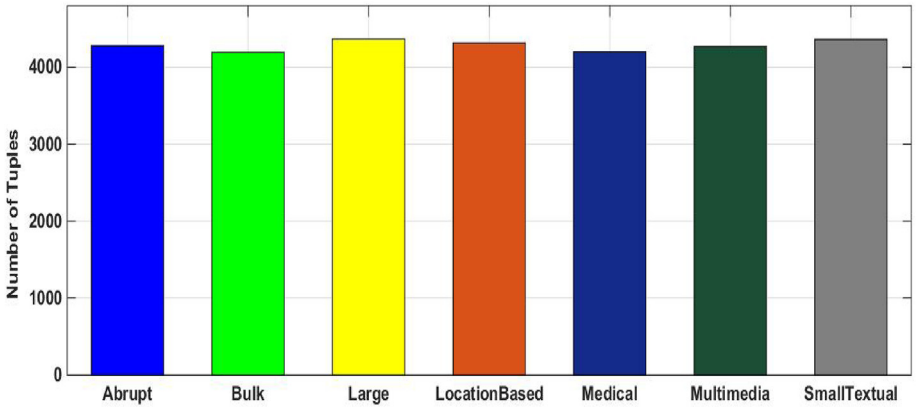


Fig. 8. IoE dataset used in simulation.

size, medical tuples and abrupt, etc.), IsReversed (if not served by any resource), IsServerd (it has a boolean value and defines whether the tuple is served ot not by any available resource - cloud/fog), IsserverbyCloud (exhibits if the tuple is served by the cloud resource), type of the device (sensor, mobile, actuator etc), queue delay (the time a task rests in queue), processing time etc. The dataset can be used in varied computing environments.

5.2. Geographic deployment

Fog servers and IoE devices are deployed randomly at varied latitude and longitude of Rawalpindi and Islamabad cities, in the simulation setup. The specifications of the fog servers are presented in Table 4. Whereas, Cloud data centers are deployed in Singapore and United States of America (USA).

5.3. Evaluation metrics

The following evaluation metrics have been considered to evaluate the performance of the proposed scheme. **Processing Time:** The time taken by a tuple/job for processing at a fog server is termed as processing time. We compute the processing time using Equation (1).

$$T_p = \frac{P_i(t)}{CP_i(fs)} \tag{1}$$

The T_p refers to the total processing time. Where, $P_i(t)$ is the power of the i^th and CP stands for the current power of a fog server fs. **Response time:** The round trip time of a tuple when it is generated from the source and returned back after completion. The following Equation (2) shows the computation of response time. Where, RT stands for response time, T_p is the propagation time, T_p shows processing time and T_q presents the queue delay.

$$RT = T_p + T_p + T_q \tag{2}$$

5.4. Policies

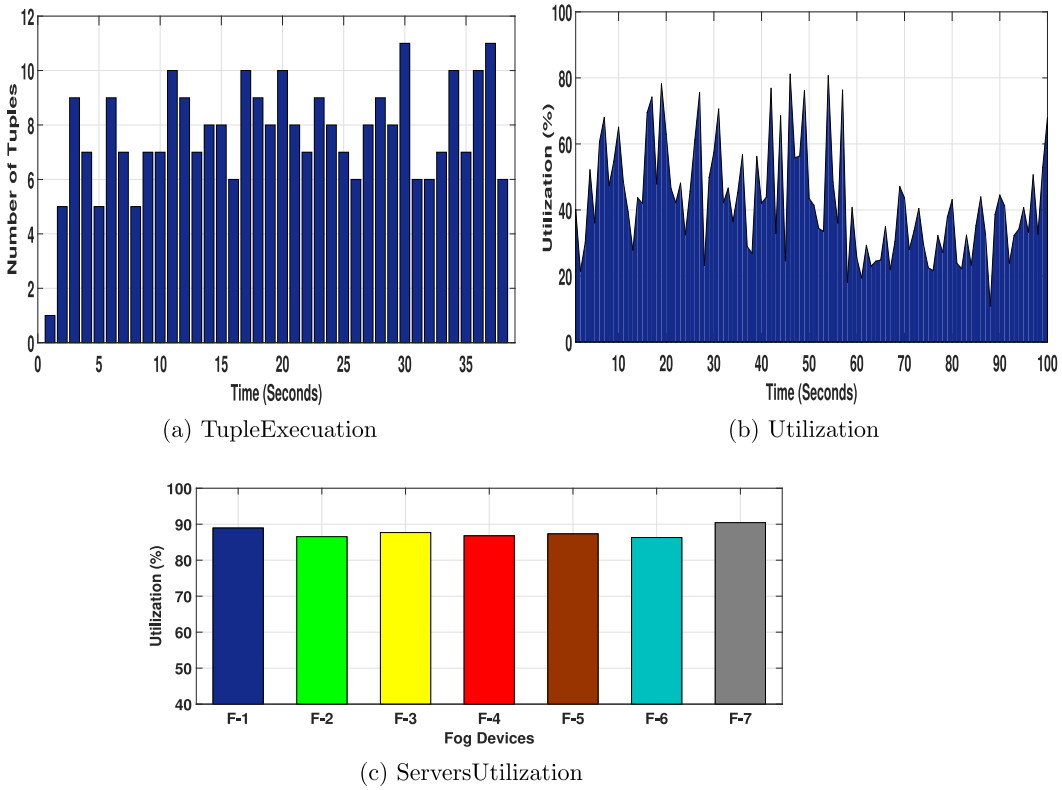
Here, following IoE data generation policies are considered to evaluate the proposed scheme. **Random policy:** This policy sends the tuples to fog servers arbitrarily without following their **First-Come-First-Served:** It refers to the synchronous forwarding of the tuples to fog servers (Bittencourt et al., 2017) according to their order of generation from IoE devices. **Shortest-Job-First:** SJF policy sends the small tuples on higher priority to fog servers in comparison to other tuples.

5.5. Results

Fog Servers Utilization: Fig. 9 shows the utilization of Fog resources. Fig. 9. a shows the simulation time (in seconds) at x-axis and y-axis represents the number of tuples a Fog server is serving. Each

Table 4
Fog servers specifications.

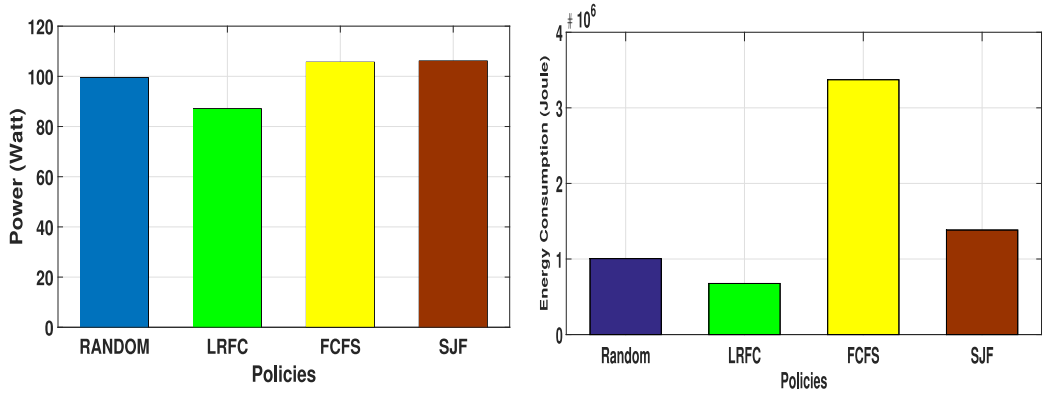
Fog_ID	Ram (MB)	MIPS	UpBw (kbps)	DownBw (kbps)	No. of Cores	Longitude	Latitude
F-0	10,240	85,000	1500	1200	2	72.94457212	33.74430192
F-1	8192	75,000	1200	1000	2	72.99649278	33.57278
F-2	8192	75,000	1200	1000	2	73.03918583	33.78799432
F-3	10,240	85,000	1500	1200	2	72.89806118	33.75918676
F-4	16,384	110,000	2500	1700	4	73.05820695	33.58553272
F-5	6144	50,000	1000	700	1	72.95355966	33.73420675
F-6	10,240	85,000	1500	1200	2	73.05820695	33.58553272
F-7	16,384	110,000	2500	1700	4	72.95355966	33.73420675
F-8	8192	75,000	1200	1000	2	73.01082938	33.70381249
F-9	12,288	95,000	2000	1500	3	72.89806118	33.75918676



(a) TupleExecuation (b) Utilization

(c) ServersUtilization

Fig. 9. Fog servers utilization.

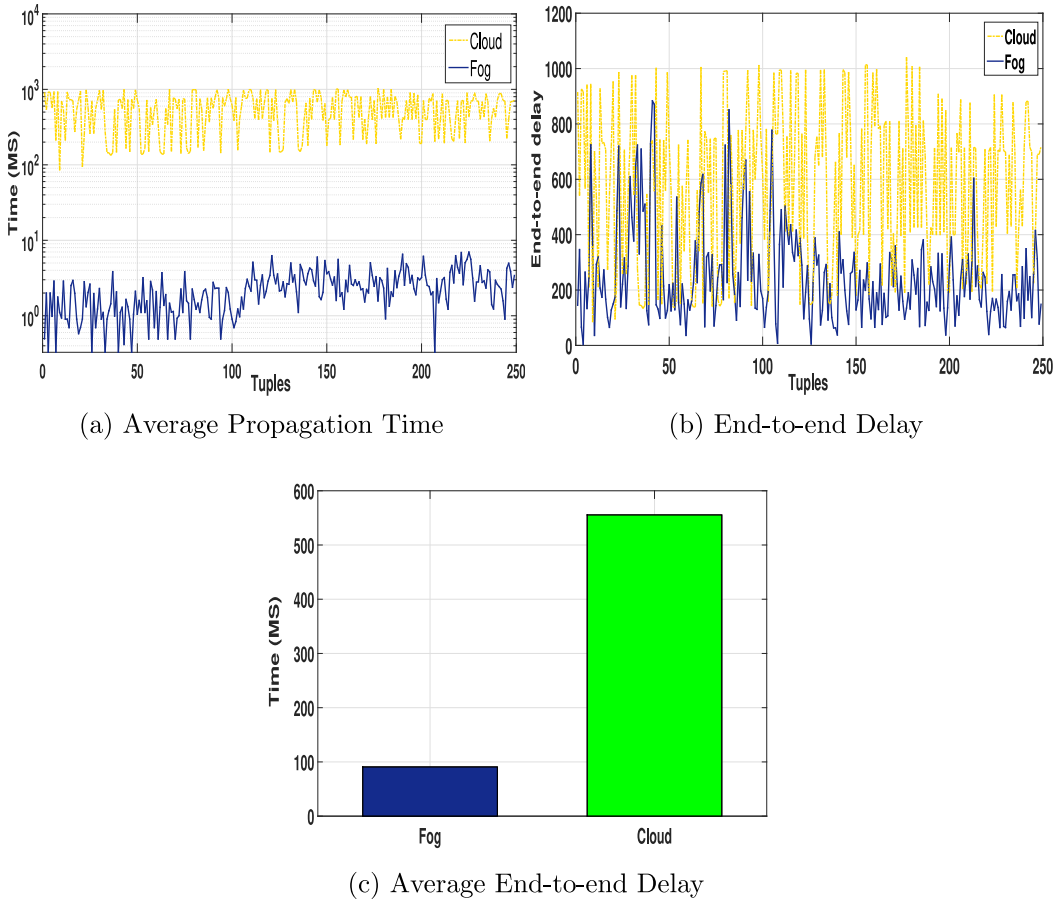


(a) GrandPowerConsumption (b) Energy Consumption

Fig. 10. Average Power and Energy Consumption of fog resources by different task scheduling policies.

Fog server has specific computing capacity in terms of MIPS. Similarly, the incoming job also has a specific size in terms of MIPS. The size, mean-inter-arrival time and the change in number of tuples directly affect the utilization of a Fog device. Fig. 9. b shows utilization of a Fog server in 100 s of simulation time. Fig. 9. c shows the utilization of 7 Fog servers. The x-axis presents fog devices and y-axis shows their percent utilization. In case of LRFC, the minimum utilization of Fog servers is 86.295 and it reaches the 90.435 at maximum. Servers utilization is used as a key metric to determine the resource utilization in Fog environment. Hence, it has a main role in resource management. Subsequently, the utilization of Fog resources has a direct impact on the their energy consumption. The under-utilization of Fog servers results in wastage of resources. Consequently, the efficient utilization of distributed resources is crucial to reduce the energy and to enhance the performance in fog environment.

Power and Energy Consumption: Fig. 10a and 10. b exhibits the power and energy consumption of fog resources respectively. Two types of power consumption occurs in fog servers - static power consumption and dynamic power consumption. When a fog server is powered-on and it has no load, it consumes a constant amount of power that is required by its hardware and software for basic functions is called static power. The rest of its power consumption is proportional to its utilization - known as dynamic power consumption. In the experiments, both types of power consumption is computed. Fig. 10. a presents the average power consumed by fog resources while applying different types of task scheduling policies considered in our simulation environment.



(a) Average Propagation Time (b) End-to-end Delay

(c) Average End-to-end Delay

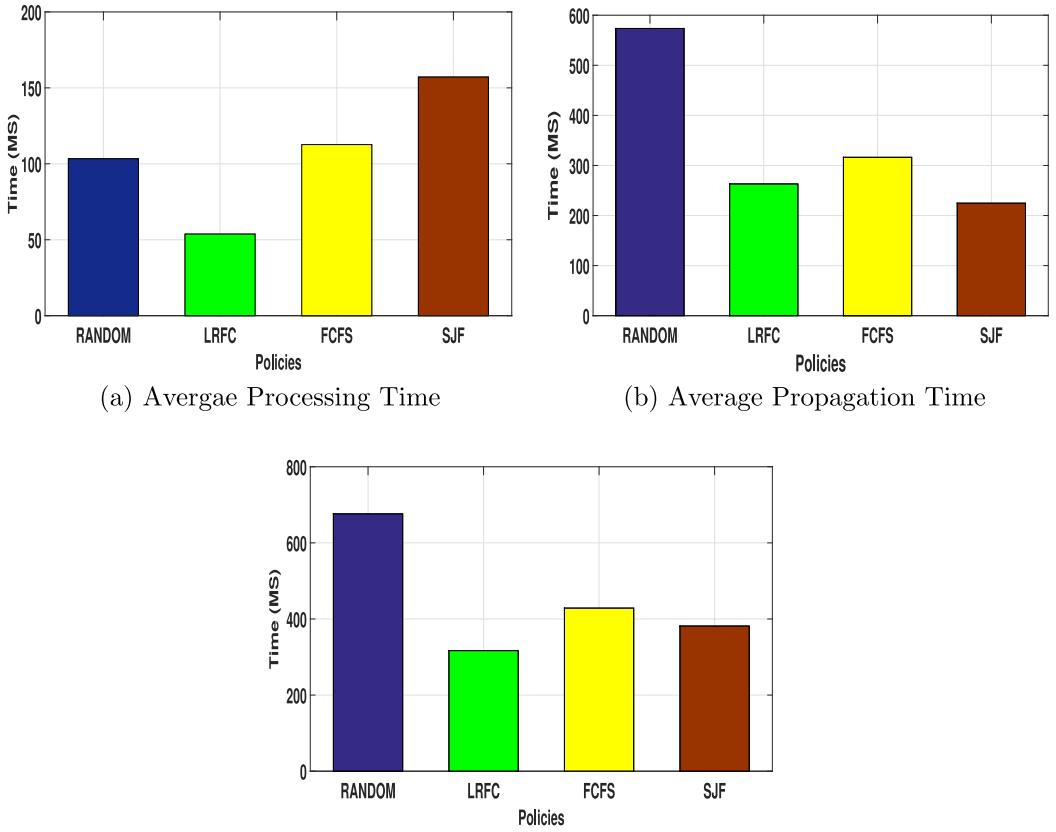
Fig. 11. A comparison of Cloud-only and Fog-cloud environments in terms of Propagation Time and End-to-end Delay.

It can be seen that the proposed policy exhibits the lowest power consumption in comparison to the rest of the policies. It is due to the fact that LRFC allocates the resources according to the requirements of the jobs that ultimately has an impact on the overall power consumption of fog resources. The efficient task scheduling results in reduced power consumption in distributed environment (Lee and Zomaya, 2010). Similarly, the energy consumption (that is the power consumed in a specific time period) is also considerately decreased while allocating resources efficiently. Most of the state-of-the art policies concentrate on the load-balancing that improves the quality of service (QoS); however, the energy consumption is somehow compromised. Moreover, allocating resources without considering nature of the jobs results in inefficient resource utilization that ultimately increases their power and energy consumption. Similarly, the overall computation time is also increased if the inappropriate resources are allocated. The LRFC policy selects the resources that perform best for specific types of jobs. This activity decreases the computation time that consequently minimizes the energy consumed by fog servers. The FCFS policy exhibits the highest energy consumption as it simply follows the flow of traffic and lacks in selecting the best resources. The shortest job first policy performs better than FCFS as it initially serves the smaller jobs. Subsequently, the smaller jobs do not wait and are served timely. In FCFS, smaller and bigger jobs are served at the same time and resources are occupied by bigger jobs that increase the overall computation time and hence the energy consumption. The random policy creates a load-balancing in the overall system and performs better than FCFS and SJF. The LRFC shows the optimal results because of the desirable task scheduling according to the job requirements.

Fig. 11. a shows the average propagation delay in cloud-only and cloud-fog environments. It is clear that Fog-cloud environment is clear winner in terms of propagation delay. It is due to the remote deployment of cloud resources that create a huge latency in flow of jobs generated far from the cloud infrastructure. Contrarily, Fog exists nearer to the IoE devices and significantly reduces the delay (Mahmud et al., 2018).

Whereas, Fig. 11b and 11. c shows a comparison of end-to-end delay of IoE jobs served on cloud-only and fog-cloud environments. The y-axis of both Figures present the end-to-end delay in milliseconds. It exhibits the usage of network and processing resources by IoE jobs. As, the cloud is remote from the end devices so the jobs traverse all the network and use all underlying network resources and bandwidth. Whereas, fog is closer to devices where jobs are generated and results in lower utilization of network resources. In the given scenario, majority of the requests are served by fog that not only results in lower latency but it also reduces the burden of traffic on cloud.

Fig. 12. a presents the processing delay occurred at the Fog servers while serving various IoE requests and applying different task scheduling policies. The x-axis presents the different task scheduling policies; whereas, y-axis shows the processing time in milliseconds. The strategy of allocating available Fog resources to incoming jobs has an important impact on the overall processing delay. If the jobs are not placed optimally at the fog resources, it causes the resource contention that increases the processing delay. Moreover, electing the best resource according to the job requirements improves the performance of the system by decreasing the delay. Fig. 12. a presents that when resources are allocated dynamically and intelligently, it reduces the server level pro-



(a) Avergae Processing Time

(b) Average Propagation Time

(c) Average End-to-end Delay

Fig. 12. Performance evaluation of proposed scheme.

cessing delay. Hence, LRFC outperforms rest of the policies considered in our simulations, in terms of overall average processing delay.

Fig. 12. b presents the average propagation time taken by all IoE jobs while traveling in the network. It includes the time taken in transmission of the data at links, at routers and switches and also in fog-to-fog communication in case of cooperation among fog servers. Fig. 12. b clearly depicts that LRFC policy shows the better results compared to Random and FCFS; however, a slight increase is noted compared to SJF policy. Initially, LRFC takes some time to be matured and jobs are assigned randomly to the fog resources. However, it gets smarter with the passage of time. The prematurity time may have a negative impact on the performance at initial stages.

The resultant end-to-end delay depends on the efficiency of task scheduling algorithm (Mahmud et al., 2018). Fig. 12. c shows the performance evaluation of different task scheduling algorithms in a fog-cloud environment, in terms of end-to-end delay. The x-axis shows the policies considered in our simulation and y-axis presents their corresponding end-to-end delay in milliseconds. The proposed LRFC policy exhibits the lowest end-to-end delay compared to FCFS, Random and SJF policies. As described earlier, the efficient task scheduling has a crucial role on the end-to-end delay. When the resources are allocated according to the job requirement keeping in view the other important factors like availability, capacity and proximity of resources (as done in LRFC), the average end-to-end time is reduced. Consequently, QoE of the system is improved. Additionally, assigning best resource for incoming job reduces the extra delays such as queuing and migration delays as well.

Finally, a comparative analysis of considered evaluation metrics is performed using 30*k* and 50*k* IoT jobs, as shown in Fig. 13. It can be observed that the proposed policy exhibits a slight difference among the varied performance evaluation metrics used in the paper even with a considerable amount of increase in IoT jobs. Consequently, the system shows a normal behaviour that confirms the scalability of the proposed approach in terms of number of number of IoE jobs.

6. Conclusion and future work

The paper explores task scheduling thoroughly in Fog computing environments. An adaptive and intelligent task scheduling technique, Learning Repository Fog-Cloud (LRFC), has been proposed to improve QoS (i.e., response time, and processing time of tuples) and energy consumption (i.e., power consumption of fog devices). The authors have proposed a smart soft layer between IoE/IoT-devices and Fog nodes that can be extended to implement various types of learning based policies. The proposed deployment model exhibits scalability and thus avoid performance bottlenecks. The proposed approach has been thoroughly evaluated using extensive simulations and verified formally. The verification of the proposed approach with current state-of-the-art shows promising results both in terms of energy efficiency and QoS. Our future work includes the utilization of our proposed smart layer with various experimentation of intelligent learning based techniques in combination with varied state-of-the-art scheduling policies to mainly access varied futuristic large scale distributed computational paradigms (i.e., Edge, Fog, and Cloud etc.)

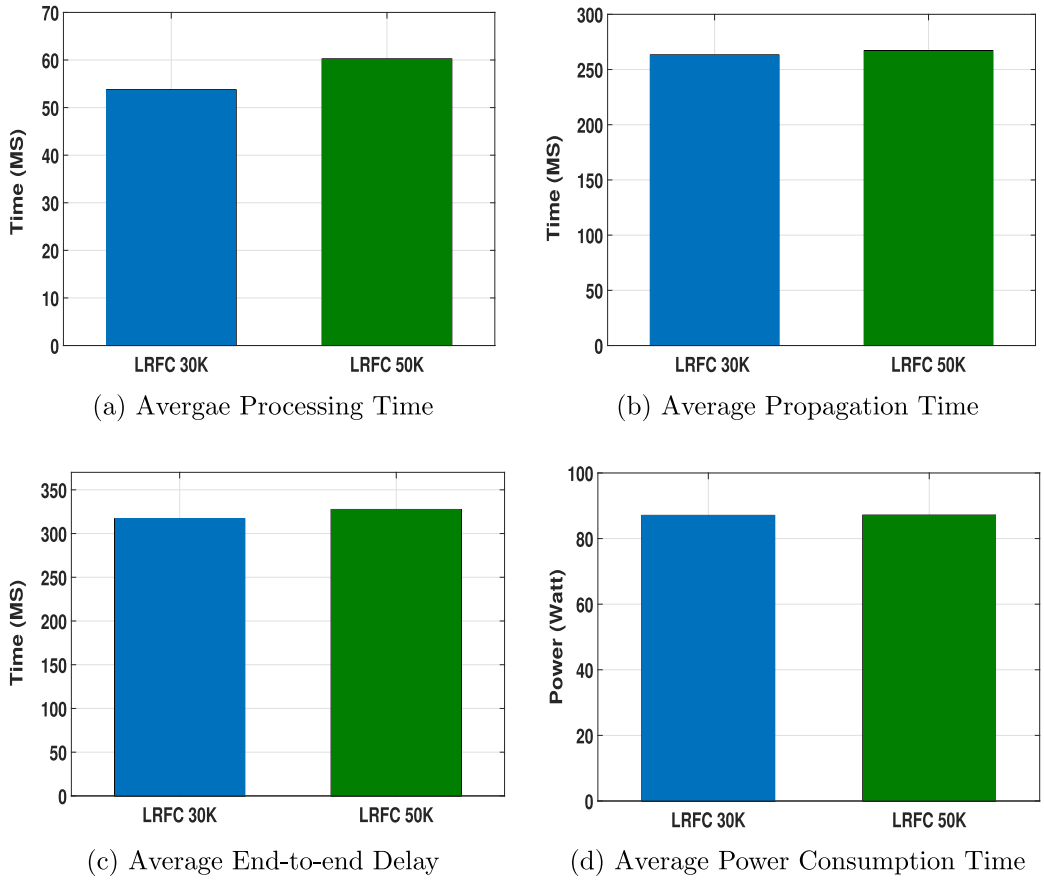


Fig. 13. Performance evaluation of LRFC with 30 k and 50 k.

Declaration of competing interest

We declare that there is no author's conflict of interest.

Acknowledgments

This work is supported by the European Commission, under the ASTRID and FutureTPM projects; Grant Agreements no. 786922 and 779391, respectively.

References

Aazam, M., 2015. Dynamic Resource Provisioning through Fog Micro Datacenter, Pervasive Computing and Communication Workshops. PerCom Workshops), pp. 105–110.
Aazam, M., Huh, E.-N., 2014. Fog Computing and Smart Gateway Based Communication for Cloud of Things, pp. 464–470.
Aazam, M., Huh, E.N., 2015. Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT. In: Proceedings - International Conference on Advanced Information Networking and Applications, AINA 2015-April, pp. 687–694.
Aha, D.W., 1991. Case-based learning algorithms. Database 1, 1–13.
Alrashed, S., A. A. J. o. I. A. &, U., 2017. User behaviour classification and prediction using fuzzy rule based system and linear regression. Res. Notes 12 (2017).
Amin, K., 2017. Building an Integrated CBR-Big Data Oriented Architecture for Case-Based Reasoning Systems.
Aslam, A., Ahmad, N., Saba, T., Almazyad, A.S., Rehman, A., Anjum, A., Khan, A., 2017. Decision support system for risk assessment and management strategies in distributed software development. IEEE Access 5, 20349–20373.
Bitam, S., Zeadally, S., Mellouk, A., 2017. Fog computing job scheduling optimization based on bees swarm. Enterprise Inf. Syst. 1–25 00.
Bitam, S., Zeadally, S., Mellouk, A., 2018. Fog computing job scheduling optimization based on bees swarm. Enterprise Inf. Syst. 12, 373–397.
Bittencourt, L.F., Diaz-Montes, J., Buyya, R., Rana, O.F., Parashar, M., 2017. Mobility-aware application scheduling in fog computing. IEEE Cloud Comput. 4, 26–35.
Brown, D., Aldea, A., Harrison, R., Martin, C., Bayley, I., 2017. Temporal Case-Based Reasoning for Type 1 Diabetes Mellitus Bolus Insulin Decision Support.
Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R., 2011. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software Pract. Ex. 41, 23–50.
Cardellini, V., Grassi, V., Presti, F.L., Nardelli, M., 2016. On QoS-Aware scheduling of data stream applications over fog computing infrastructures. In: Proceedings - IEEE Symposium on Computers and Communications 2016-Febru, pp. 271–276.
Chandio, A.A., Bilal, K., Tziritas, N., Yu, Z., Jiang, Q., Khan, S.U., Xu, C.-Z., 2014. A comparative study on resource allocation and energy efficient job scheduling strategies in large-scale parallel computing systems. Cluster Comput. 17, 1349–1367.
Chappell, D., et al., 2008. Introducing the Azure Services Platform. White paper. Oct 1364.
Choo, K.-K.R., Lu, R., Chen, L., Yi, X., 2018. A foggy research future: advances and future opportunities in fog computing research. Future Generat. Comput. Syst. 78, 677–679.
Dar, B.K., Shah, M.A., Islam, S.U., Maple, C., Mussadiq, S., Khan, S., 2019. Delay-aware accident detection and response system using fog computing. IEEE Access 7, 70975–70985.
Dastjerdi, A.V., Buyya, R., 2016. Fog computing: helping the internet of things realize its potential. Computer 49, 112–116.
Deng, R., Lu, R., Lai, C., Luan, T.H., Liang, H., 2016. Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. IEEE Intern. Thing. J. 3, 1171–1181.
Elkano, M., Galar, M., Sanz, J., Bustince, H., 2017. CHI-BD: A Fuzzy Rule-Based Classification System for Big Data Classification Problems, Fuzzy Sets and Systems.
Ficco, M., Esposito, C., Xiang, Y., Palmieri, F., 2017. Pseudo-dynamic testing of realistic edge-fog cloud ecosystems. IEEE Commun. Mag. 55, 98–104.
Gupta, H., Vahid Dastjerdi, A., Ghosh, S.K., Buyya, R., ifogsim, 2017. A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. Software Pract. Ex. 47, 1275–1296.
He, J., Cheng, P., Shi, L., Chen, J., Sun, Y., 2014. Time synchronization in WSNs: a maximum-value-based consensus approach. IEEE Trans. Automat. Contr. 59, 660–675.
Intharawijitr, K., Iida, K., Koga, H., 2016. Analysis of fog model considering computing and communication latency in 5G cellular networks, 2016 IEEE International Conference on Pervasive Computing and Communication Workshops. PerCom Workshops 2016, 5–8.
Iot-compute-dataset, 2019. Retrieved on (2019-10-28). https://github.com/saifulislamPhD/IoT-Compute-Dataset.

Keshtkar, A., Member, S., Arzanpour, S., 2014. Design and Implementation of a Rule-Based Learning Algorithm Using Zigbee Wireless Sensors for Energy Management, pp. 1–6.
Khan, M., Awais, M., Shamail, S., Awan, I., 2011. An empirical study of modeling self-management capabilities in autonomic systems using case-based reasoning. Simulat. Model. Pract. Theor. 19.
Kuhn, H.W., 2010. The Hungarian method for the assignment problem. In: 50 Years of Integer Programming 1958-2008: from the Early Years to the State-Of-The-Art, vol. 2, pp. 29–47.
Kumar, K.A., Singh, Y., Sanyal, S., 2009. Hybrid approach using case-based reasoning and rule-based reasoning for domain independent clinical decision support in ICU. Expert Syst. Appl. 36, 65–71.
Y. C. Lee, A. Y. Zomaya, Energy efficient resource allocation in large scale distributed systems, in: Distributed Computing and Applications to Business Engineering and Science (DCABES), 2010 Ninth International Symposium on, IEEE, pp. 580583.
Li, D., Sun, X., 2006. Nonlinear Integer Programming - Duan Li, Xiaoling Sun - Google Books. Springer Science \& Business Media.
Ligza, A., 2006. Logical Foundations for Rule-Based Systems. Logical Foundations for Rule-Based Systems, pp. 191–198.
Mahmud, R., Ramamohanarao, K., Buyya, R., 2018. Latency-aware application module management for fog computing environments. ACM Trans. Internet Technol. 19, 9.
Mahmud, R., Srirama, S.N., Ramamohanarao, K., Buyya, R., 2019. Quality of experience (qoe)-aware placement of applications in fog computing environments. J. Parallel Distr. Comput..
Mohan, N., Kangasharju, J., 2017. Edge-fog Cloud: A Distributed Cloud for Internet of Things Computations, 2016 Cloudification of the Internet of Things, CIoT 2016.
Mouradian, C., Naboulsi, D., Yangui, S., Glitho, R.H., Morrow, M.J., Polakos, P.A., 2017. A Comprehensive Survey on Fog Computing: State-Of-The-Art and Research Challenges. IEEE Communications Surveys and Tutorials.
Nez, H., Angulo, C., Catal, A., 2006. Rule-based learning systems for support vector machines. Neural Process. Lett. 24, 1–18.
Osanaiye, O.A., Chen, S., Yan, Z., Lu, R., Choo, K.-K.R., Dlodlo, M.E., 2017. From cloud to fog computing: a review and a conceptual live vm migration framework. IEEE Access 5, 8284–8300.
Oueis, J., Strinati, E.C., Barbarossa, S., 2015. The fog balancing: load distribution for small cell cloud computing. In: 2015 IEEE 81st Vehicular Technology Conference (VTC Spring), pp. 1–6.
M. R. Palankar, A. Iamnitchi, M. Ripeanu, S. Garfinkel, Amazon s3 for science grids: a viable solution?, in: Proceedings of the 2008 International Workshop on Data-Aware Distributed Computing, ACM, pp. 5564.
Prentzas, J., Hatzilygeroudis, I., 2003. Integrations of rule-based and case-based reasoning. In: Proceedings of the International Conference on Computer, Communication and Control Technologies, 4, pp. 81–85.
Sharaf-El-Deen, D.A., Moawad, I.F., Khalifa, M.E., 2014. A new hybrid case-based reasoning approach for medical diagnosis systems. J. Med. Syst. 38, 9.
Song, N., Gong, C., An, X., Zhan, Q., 2016. Fog computing dynamic load balancing mechanism based on graph repartitioning. China Commun. 13, 156–164.
Toor, A., ul Islam, S., Ahmed, G., Jabbar, S., Khalid, S., Sharif, A.M., 2019. Energy efficient edge-of-things. EURASIP J. Wirel. Commun. Netw. 2019, 82.
Toor, A., ul Islam, S., Sohail, N., Akhunzada, A., Boudjadar, J., Khattak, H.A., Din, I.U., Rodrigues, J.J., 2019. Energy and Performance Aware Fog Computing: A Case of Dvfs and Green Renewable Energy. Future Generation Computer Systems.
Tung, Y.H., Tseng, S.S., Weng, J.F., Lee, T.P., Liao, A.Y.H., Tsai, W.N., 2010. A rule-based CBR approach for expert finding and problem diagnosis. Expert Syst. Appl. 37, 2427–2438.
Van Den Bossche, R., Vanmechelen, K., Broeckhove, J., 2010. Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads, Proceedings - 2010 IEEE 3rd International Conference on Cloud Computing. CLOUD 2010, 228–235.
Wortmann, F., Flchter, K., 2015. Internet of things: Technology and value added. Business Inform. Syst. Eng. 57, 221–224.
Yan, A., Yu, H., Wang, D., 2017. Case-based reasoning classifier based on learning pseudo metric retrieval. Expert Syst. Appl. 89, 91–98.
Yuce, B., Rezgui, Y., 2017. An ANN-GA semantic rule-based system to reduce the gap between predicted and actual energy consumption in buildings. IEEE Trans. Autom. Sci. Eng. 14, 1351–1363.
Zeng, D., Gu, L., Guo, S., Cheng, Z., 2016. Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system. IEEE Trans. 65, 3702–3712.

Faizan Murtaza received his MS degree in Software Engineering from the Department of Computer Science, COMSATS University, Islamabad, Pakistan in 2018. He is a senior software developer at Enabling Technologies, Rawalpindi, Pakistan. His main research interest includes energy and performance aware resource management in Internet of Things (IoT) enabled Fog-Cloud environment.

Dr. Adnan Akhunzada is an enthusiastic and dedicated professional with extensive 12 years of R&D experience both in ICT industry and academia, with demonstrated history and a proven track record of high impact published research (i.e., Patents, Journals, Transactions, Commercial Products, Book chapters, Reputable Magazines, Conferences and Conference Proceedings). His experience as an educator & researcher is diverse. It includes work as a lecturer, a senior lecturer, a year tutor, occasional lecturer at other engineering departments, as an Assistant Professor at COMSATS University Islamabad (CUI), Senior Researcher at RISE SICs Vasteras AB, Sweden, as a Research Fellow & Scientific Lead at DTU Compute, The Technical University of Denmark (DTU), and visiting professor having mentorship of graduate students, and supervision of academic and R&D projects both at UG and PG level. He has also been involved in international accreditation such as Accreditation Board for Engineering and Technology (ABET), and curriculum development according to the guidelines of ACM/IEEE. He is currently involved in various EU and Swedish funded projects of cyber security. His main research capabilities and interest lies in the field of Cyber Security, Machine Learning, Deep Learning, Reinforcement learning, Artificial Intelligence, Blockchain and Data Mining, Information Systems, Large scale distributed systems (i.e., Edge, Fog, & Cloud, SDNs), IoT, Industry 4.0, and Internet of Everything (IoE)). He is a member of technical programme committee of varied reputable conferences and editorial boards. He is presently serving as an associate editor of IEEE Access.

Dr. Saif ul Islam received his PhD in Computer Science at the University Toulouse III Paul Sabatier, France in 2015. He is Assistant Professor at the Department of Computer Science, Dr. A. Q. Khan Institute of Computer Science and Information Technology, Rawalpindi, Pakistan. He served as a focal person of a research team at COMSATS working in O2 project in collaboration with CERN Switzerland. He has been part of the European Union funded research projects during his PhD. His research interests include resource and energy management in large scale distributed systems and in computer/wireless networks.

Dr. Jalil Boudjadar is an Assistant Professor in Software Engineering group, ECE division at the Department of Engineering at Aarhus University, Denmark. He is also a member of the DIGIT research Centre. He received his MS degree in June 2008 from Limoges University, and his PhD in December 2012 from Toulouse University France. His research interests include: Modeling and analysis of safety and security of software-intensive embedded systems, Performance and energy-consumption analysis of Embedded and cyber-physical systems, Real-time scheduling and resource sharing, Architecture and Architectural standards of software systems, Formal verification, semantics and refinement of real-time systems.

Prof. Rajkumar Buyya is a Fellow of IEEE, Professor of Computer Science and Software Engineering, Future Fellow of the Australian Research Council, and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft Pty Ltd., a spin-off company of the University, commercializing its innovations in Grid and Cloud Computing. Dr. Buyya has authored/co-authored over 450 publications. He is one of the highly cited authors in computer science and software engineering worldwide. Microsoft Academic Search Index ranked Dr. Buyya as one of the Top 5 Authors during the last 10 years (2001–2012) and #1 in the world during the last 5 years (2007–2012) in the area of Distributed and Parallel Computing. For further information on Dr. Buyya, please visit: http://www.buyya.com