Quality of Experience (QoE)-aware Placement of Applications in Fog Computing Environments

Redowan Mahmud[1], Satish Narayana Srirama[1,2], Kotagiri Ramamohanarao[1], Rajkumar Buyya[1]

Abstract

Fog computing aims at offering Cloud like services at the network edge for supporting Internet of Things (IoT) applications with low latency response requirements. Hierarchical, distributed and heterogeneous nature of computational instances make application placement in Fog a challenging task. Diversified user expectations and different features of IoT devices also intensify the application placement problem. Placement of applications to compatible Fog instances based on user expectations can enhance Quality of Experience (QoE) regarding the system services. In this paper, we propose a QoE-aware application placement policy that prioritizes different application placement requests according to user expectations and calculates the capabilities of Fog instances considering their current status. In Fog computing environment, it also facilitates placement of applications to suitable Fog instances so that user QoE is maximized in respect of utility access, resource consumption and service delivery. The proposed policy is evaluated by simulating a Fog environment using *iFogSim*. Experimental results indicate that the policy significantly improves data processing time, network congestion, resource affordability and service quality.

Keywords: Fog computing, Application placement, Quality of Experience, Fuzzy logic, User expectation.

1. Introduction

Modern computing and networking techniques are rapidly expanding Internet of Things (IoT) applications in many domains. Real-time interaction and stringent service delivery deadline resist IoT-applications to be placed in remote Cloud. Therefore, Fog computing paradigm is introduced to extend Cloud-based utilities for IoT-applications at the edge network [1]. In Fog, networking entities like gateway servers, routers, switches, etc. are known as Fog nodes and used for computational purposes. Fog can also support unified use of both Edge and Cloud resources. Fog computing facilitates IoT-application placement in proximity of the data source. Hence, it reduces network load and ensures in-time service delivery [2].

Unlike Cloud datacentres, Fog nodes are geographically distributed and resource constrained. Among Fog nodes, network round-trip time, data processing speed and resource availability also vary significantly. Therefore, application placement in Fog become quite challenging. To attain certain service level objectives in Fog, different application placement policies are required. Quality of Service (QoS) [3][4], resource [5], situation-aware [6] application placement in Fog have already been exploited. However, the impact of Quality of Experience (QoE) in Fog-based application placement is yet to be investigated extensively. In some cases, QoS and QoE can complement each other, although subtle differences between them often lead towards separate policy-based service management.

QoE is widely accepted as the user centric measurement of different service aspects. It observes user requirements, intentions and perceptions regarding a service in particular context [7]. Since QoE deals with user interests, QoE-aware policies can enhance user loyalty and decrease service relinquish rate. In Fog, QoE-aware policies have already been used for optimizing service coverage [8] and resource estimation [9]. Apart from resource and service provisioning, application placement in Fog reckoning user QoE can improve data processing time, resource consumption and network quality. However, in real-time environment like Fog, user interests regarding different system services vary from one to another and QoE dominating factors change very frequently. Therefore, developing efficient QoE-aware policies for Fog is a challenging task.

Currently different techniques are applied to identify and measure QoE. Feedback-based approaches such as Mean Opinion Scores (MOS), Standard deviation of Opinion Scores (SOS) and Net Promoter Score (NPS) [9][10] are commonly used to define user QoE. In IoT, where human interventions are limited and real-time interactions happen very often, giving feedback after every certain interval to notify QoE, is not feasible. Similarly, prediction-based QoE models [11] [12] also fail when QoE dominating factors vary significantly. Evaluation of QoE after placing applications creates complexities, if any placement modification based on the evaluation is required to be made. In this sense, prior to application placement, it is more viable to identify QoE dominating factors and their combined impact on user QoE. Later, applications can be placed to suitable computing instances by meeting the factors so that user

[1]Cloud Computing and Distributed Systems (CLOUDS) Laboratory
School of Computing and Information Systems
The University of Melbourne, Australia.
[2]Mobile and Cloud Laboratory, Institute of Computer Science
University of Tartu, Estonia
Email: mahmudm@student.unimelb.edu.au

QoE does not degrade. Thus, discrepancy between user feedback and QoE on specific service attribute can be monitored.

In this paper, several user expectation parameters are identified that can influence the QoE. The user *Expectation Metric* includes parameters regarding service access rate of the application user, required resources to run the application and expected data processing time. Based on user Expectation Metric, each application placement request is prioritized. Fog computing instances are also classified according to their *Status Metric* parameters (proximity, resource availability and processing speed). Finally, prioritized application placement requests are mapped to competent computing instances so that user QoE regarding the system services gets maximized.

The main **contributions** of the paper are:

- A QoE-aware application placement policy comprising of separate Fuzzy logic based approaches that prioritizes different application placement requests and classifies Fog computational instances based on the user expectations and current status of the instances respectively.
- A linearly optimized mapping of application placement requests to Fog computing instances that ensures maximized QoE-gain of the user.
- The proposed policy is evaluated through simulation using *iFogSim* [13]. The experimental results show significant improvement in QoE enhancement compared to other QoE and QoS-aware policies.

The rest of the paper is organized as follows. In Section 2, relevant research works are reviewed. In Section 3 and 4, the motivation and the addressed problem of this research are discussed. Section 5 represents the system overview and assumptions. The proposed QoE-aware application placement policy and an illustrative example are described in Section 6 and 7 respectively. Section 8 enlightens the simulation environment and the performance evaluation. Finally Section 9 concludes the paper and offers direction for future work.
The list of acronyms used in this paper is shown in Table 1.

Table 1: List of Acronyms

CCS	Capacity Class Score
CoAP	Constrained Application Protocol
EEG	Electroencephalogram
FCN	Fog Computational Node
FGN	Fog Gateway Node
IoT	Internet of Things
ITU	International Telecommunication Union
MCI	Micro Computing Instance
MeFoRE	MEdia FOg Resource Estimation
NPS	Net Promoter Score
NRR	Network Relaxation Ratio
PTRR	Processing Time Reduction Ratio
QoE	Quality of Experience
QoS	Quality of Service
REST	Representational State Transfer
RG	Resource Gain
RoE	Rating of Expectation
SCIP	Solving Constraint Integer Programs
SNMP	Simple Network Management Protocol
TIPS	Thousand Instructions Per Second

2. Related Work

A summary of several QoS/QoE-aware application management policies in different computing paradigms is shown in Table 2. Mahmud et al. in [14] propose a context-aware application scheduling policy in Mobile Cloud Computing (MCC) to enhance user's QoE. The policy runs in a centralized Cloudlet and prioritizes users requests based on battery level of the requesting device and signal to noise ratio of the network. It ensures users to get response of their requests before terminating access to the service due to poor connectivity or device failure. It also focuses on QoE gain through differences between service delivery deadline and actual service delivery time.

Zhou et al. in [15] propose a MCC-based QoE-aware cache management policy for multi-media applications. The policy finds the best data streaming bit rate in different scenarios. It ranks the user's video streaming requests based on the access rate and then allocates available resources at the caching server according to the rank of the requests. The relationship between user provided feedback and server response rate determines the enhanced QoE of the users. End device, base stations and cache servers participate simultaneously to conduct the policy.

Peng et al. in [16] propose a QoE-aware application management framework for Mobile Edge Computing (MEC) by applying network function virtualization and software defined networking. Due to proximity of MEC instances, the proposed framework inherently meets user's expectation regarding service access. Besides, it takes user's resource requirements and the global view of the available resources into account while managing the applications through a centralized orchestrator. The developed MEC eco-system is capable of enhancing user's QoE in both uplink and downlink directions.

Dutta et al. discuss a QoE-aware transcoding policy for MEC in [17]. According to the policy, a centralized edge orchestrator assesses the user's expected service processing time (tolerable buffering delay) and adjusts the video processing speed (encoding rate) so that user's QoE in respect of service responsiveness does not degrade. After a fixed time interval, the policy checks whether the encoding rate is acceptable for the user or further operations are required. The policy enforces edge content customization based on user expectations.

A QoE-aware bandwidth scheduling policy for wireless communication is discussed by X. Lin et al. in [18]. It takes user's service access rate and tolerance towards packet processing delay into account while defining the user QoE indicator for the network. The policy operates in a decentralized manner over the gateway, core network and traditional wireless networking equipment. The policy enhances the QoE in terms of attained and committed ratio of the networking resources.

Anand et al. in [19] propose a QoE-optimized scheduler for multi-class system (e.g. web interactive, file downloads, etc.) in wireless networks. The QoE of end user is modeled through a cost function based on mean flow delay and prioritizes the service requests accordingly. In addition, the policy addresses resource allocation among different classes considering the sensitivity towards flow delay. The proposed scheduler is an extension of Gittin index scheduler.

Table 2: A Summary of Related Work and Their Features Comparison

Work	Observes User Expectations in			Meets Instances Status regarding			Decentralized Management	Prioritized Placement	Compound QoE Gain
	Service Access	Resource Requirement	Processing Time	Proximity/ Response Rate	Resource Availability	Processing Speed			
Mahmud et al. [14]	✓		✓			✓		✓	✓
Zhou et al. [15]	✓			✓	✓		✓	✓	
Peng et al. [16]	✓	✓		✓	✓				✓
Dutta et al. [17]			✓	✓		✓			
X. Lin et al. [18]	✓		✓	✓			✓		
Anand et al. [19]			✓		✓	✓		✓	
Skarlat et al. [3]		✓	✓		✓	✓	✓	✓	
Brogi et al. [4]		✓		✓		✓	✓		
Y. Lin et al. [8]	✓		✓	✓		✓	✓	✓	
Aazam et al. [9]		✓			✓		✓	✓	
Iotti et al. [20]	✓			✓	✓			✓	
This work (QoE-aware)	✓	✓	✓	✓	✓	✓	✓	✓	✓

In Fog computing paradigm, different QoS and QoE-aware application management policies are also studied. Skarlat et al. in [3], formulate a Fog Service Placement Problem that targets QoS-aware application placement on virtualized Fog resources. They consider deadline satisfaction of the applications as QoS-metric and follows the earliest deadline prioritization while executing the applications. The proposed policy runs through a colony based orchestration among the Fog nodes and conciliates resource requirements of the applications with available resources of the system. Each colony connects Cloud through a middleware for additional resources.

Another QoS-aware application placement policy is developed by Brogi et al. in [4]. The policy deals with responsiveness and processing speed of the infrastructure in association with monetary issues. It is used to place multi-component IoT applications in hierarchical Fog environment. Driven by the policy, a Java based tool named FogTorch is developed. The tool can be applied at any level of a application's life-cycle.

In [8], Y. Lin et al. identify three factors (response time, network congestion, service coverage) that dominate user's QoE while playing interactive games. They propose a lightweight system named *Cloud-Fog* to extend service coverage for the end users. It prioritizes service requests according to the tolerance towards latency. To maintain the games continuity even in congested network, a video encoding rate adaptation strategy is applied in that system. Besides, deadline-satisfied game state scheduling improves the service response delay. In both the approaches, data packets are dropped to a certain extent.

A QoE-based Fog resource estimation policy, named *MEdia FOg Resource Estimation (MeFoRE)*, is discussed by Aazam et al. in [9]. The policy considers user's history of service giving up (Relinquish Rate) and QoE (NPS) while prioritizing service requests and estimating Fog resources. It aims at maximizing resource utilization and QoS. Service Level Agreement (SLA) violations are tracked though poor NPS values given by a user. Number of resources is increased based on the degree of SLA violations so that the user's loyalty can be re-gained.

In [20], Iotti et al. have developed a model for Fog-based Internet access networks that assist dynamic placement of Cloud or Web content at the edge networks. The model facilitates proactive caching and enforcement of traffic policies so that network infrastructures can interact with external applications smoothly. According to the authors, the model bears great potentiality in optimizing network usage, latency and QoE and in some cases preserve resources for authorized users.

Our proposed QoE-aware application placement policy for Fog differs from the aforementioned works since we have considered multiple user expectation parameters such as service access rate, amount of required resources and sensitivity towards data processing delay simultaneously. The policy prioritizes application placement requests based on user expectations. In addition, the policy investigates resource availability, proximity and processing capabilities of Fog computational instances concurrently to identify their competency for meeting expectations of the users. The policy aims at maximizing compound QoE gain of the users in respect of less congested network, adequate resource allocation and reduced application processing time. Besides, we have developed the policy in decentralized manner so that it gets less prone to single point failure and management overhead. In fact, for Fog computing, the proposed policy can encapsulate and deal with those multidimensional aspects of QoE-aware application placement which the existing solutions can not address individually.

3. Motivation and Requirements

3.1. Scope of Quality of Experience

In Fog-enabled IoT system, the scope of QoE can be very diverse and complex. To understand the scope of QoE, it can be compared with QoS. According to International Telecommunication Union (ITU), QoS refers to the overall features of system services which help to meet the stated and implied needs of the end users [21]. Conversely, ITU defines QoE as the total acceptability of a service that is determined by subjective perception of the end users [22]. Moreover, QoE encapsulates user's requirement, intentions and perceptions while provisioning system services (network, application execution platform) whereas, QoS drives through an agreement between user and provider that strongly monitors technical attributes (cost, service delivery deadline, packet loss ratio, jitter, throughput, etc.) of system services. In addition, QoE is the subjective measurement of system services that can be expressed through both qualitative and quantitative parameters; on the contrary, QoS

is more focused on objective parameters of the underlying network and application execution platform [23].

The definitions of QoS and QoE highlights that they are fundamentally different to each other. However, sometimes user's expectation for enhanced QoE can help system services to improve their QoS [24]. For example, in an Internet-enabled system, on fixed charge, a user can expect less buffering while viewing multimedia contents. In order to enhance the user's QoE regarding that system, the network service providers can allocate sufficient bandwidth and maintain acceptable jitter that can significantly improve the QoS of the corresponding service. Conversely, end users perceived QoE can degrade the acceptability of a service greatly even when the proper QoS regarding the service is maintained [25]. It actually happens due to diversified characteristics of the users. Extending the aforementioned example, let's assume, on fixed charge, the Internet-enabled system provisions the network service in such way so that the QoS guarantees downloading of a particular file in maximum 5 minutes time. Now, two users require that file in 3 and 7 minutes respectively. If the system downloads the file in exact 5 minutes time, the expectation of the second user will be met; however, for the first user it will be failed. As a consequence, QoE of both users will not be the same for that system although the system maintains the QoS. Moreover, in QoS-assured 5 minute time it may happen that a particular user's QoE becomes higher when the file is downloaded within 1 minute and gets lower when the file is downloaded in 4 minutes. Therefore, it is often very difficult to apply the same technique to meet both QoS and QoE regarding a system service. Since the inclusion of QoE makes service response more stringent and complicated, it requires separate treatment in comparison to QoS.

3.2. Application Scenario

In real-world, users interact with different Fog-enabled IoT applications in diversified ways. For example, to play *Electroencephalogram (EEG) Tractor Beam* game [26], a user requires wearing a MINDO-4S wireless EEG headset and connecting the smart phone to a local Fog node. The game initiates as a mobile application at each user's smart phone and with the connected Fog nodes; users exchange information with each other. During the game, wearable IoT-device sensed EEG data streams are sent to the Fog nodes through the user's smart phones. For each user, real-time EEG signal analysis and brain state (concentration) prediction are conducted at the Fog nodes. On the display, the multi-user game shows that all the players are on a ring surrounding a target object and exerting an attractive force onto the target in proportion to the level of their concentration. The user, who pulls the object towards him/her by exercising concentration, finally wins the game. In this game, for real-time interactions, Fog service access rate of the users is required to be fast and timely. Since it is a multi-user game, the amount of required resources to run the service in Fog will be large. Moreover, in such competitive scenario, the expected service delivery time for each user can become stringent.

Unlike the multi-player virtual reality game applications, there also exist comparatively less interactive IoT applications. Fog Computing based face identification [27] can be mentioned here as an example that captures facial image of a user by vision sensors or cameras. Later, the images are sent to the Fog nodes from end devices with a view to extract the facial region by using efficient face detection algorithms. Image pre-processing algorithms are also applied to improve the quality and reduce the noises of the extracted image segment. Through specific feature extraction algorithms or pattern recognition techniques, feature vector of the facial image segment is then identified. Finally, the feature vector is either sent to system database for storage or stored data is used to compare the feature vector for identifying a registered user. This kind of application is event driven that often does not require consistent access to Fog services. As, the application does not deal with multi-user simultaneously, associate services consume fewer amounts of Fog resources compared to the multi-user applications. In addition, the expected service delivery time for the application will not get stringent until any emergency situation arises.

The aforementioned examples represent that in a Fog environment, multiple applications with different user interests and requirements can run together. In such case, a general placement policy for every application, can not guarantee the enhanced QoE for all the users. It is also very difficult to ensure the convergence of user's multiple expectations to the system's affordability for the higher gain of QoE. Therefore, an efficient QoE-aware application placement strategy for Fog computing is required to develop that can meet the diversified user expectations and the system status for enhanced QoE of all the application users.

4. Problem Description

4.1. Exploration of Expectation and Status Metrics

From the discussion of Section 3, it is realized that user's expectations can vary from application to application. Different expectation parameters have individual impact on user's overall QoE and can drive the QoS of multiple system components e.g. network, application execution platform etc. simultaneously. In this work, user expectations while accessing the services, requiring computational resources and processing data signals through the applications, are investigated. The target of meeting user's expectation for faster service access can help improving the responsiveness of network. The performance of application execution platform in on-demand resource provisioning and low-latency service delivery can get enhanced if it aims at satisfying user expectations of high computational resources and processing data signals within rigid time-frame. However, the expectation parameters are subjective and can be expanded to multiple levels. For example, user service access rate for different applications can be slower, normal or faster. Besides, the priority of different applications based on multiple expectation parameters is difficult to determine while developing a QoE-aware application placement policy for Fog computing.

In addition, Fog is a distributed computing paradigm closer to the edge network. In this environment, heterogeneous and resource constraint Fog nodes are deployed in hierarchical order with a view to execute IoT applications in real-time. Moreover, it is considered that the lower-level Fog nodes are more

resource constraint compared to the higher-level nodes [28]. Extending the aforementioned characteristics of Fog environment, we have considered three different status parameters of Fog instances while identifying their capacity towards satisfying user expectations. Different round-trip-time status of the Fog instances meets the hierarchical and distributed orientation of the Fog environment along with the networking capabilities. Besides, diversified resource availability and processing speed status signify the heterogeneity among Fog instances in respect of resource capacity and application run-time environment. Since, different status parameters of Fog instances facilitate different aspects of user expectations, it is required to calculate the QoE-enhancement capabilities of Fog instances based on all the status parameters. Besides, the calculation should be more generalized so that it can cope up with any computational improvement of the Fog instances.

4.2. *Enhancement of Quality of Experience*

The main challenge of QoE-aware application placement is to determine which applications will be placed to which Fog instances. This mapping of applications and instances should be done in such a way that user's QoE gain in all aspects get maximized and system QoS in respect of packet loss rate, service cost and deadline satisfaction get observed. In this case, the priority of user expectations regarding the applications and capability of instances in meeting user expectations can be considered actively. For simplicity of the mapping, their calculation can be aligned through a generalized approach. However, in real-time and resource constraint Fog environment, mapping of applications and instances along with associated calculations should not take significant amount of time and computation effort that can obstruct the ultimate goal of the proposed policy.

5. System Overview

5.1. *Application Model*

Fog-enabled IoT applications are usually divided into multiple interconnected Application Modules [13]. The module, running at end user's proximate devices (e.g. smart phone, set top boxes, bed-side monitors, etc.), initiates the system and offers interfaces for authentication, sensing frequency calibration, data aggregation, local data storage and outcome representation. Among the Fog nodes, the subsequent Application Modules are either extended from Cloud to meet the latency issues [29] or offloaded from the user's proximate devices due to resource constraints [30]. For simplicity, here we assume that, Fog-enabled IoT applications are composed of two Application Modules; *Client Module* and *Main Application Module*.

The Client Module runs at the user's proximate devices. It grasps the user's preferences and the contextual information; and delivers output (acknowledgement/instruction) of the Main Application Module to the users. The Main Application Module conducts all data operations of the application and output of the Main Application Module is regarded as final product of the Fog-enabled IoT systems. The data operations within Main Application Module can include data filtration, data analysis, event processing, etc. Besides, execution of the Main Application Module can be ended with notification and storage operation based on the results of overall data operations. Since, the placement of Client Module is predefined, here we mainly focus on placing Main Application Module over Fog node. For simplicity, in rest of the paper, by the term "application" we refer to the Main Application Module of Fog-enabled IoT systems.

5.2. *Organization of Fog Layer*

In the system model, Cloud datacentres are the superior computational platform and IoT devices exclusively generate data signals. IoT devices do not process data due to resource and energy constraints. Fog operates as an intermediate computing paradigm between Cloud datacentres and IoT devices. In Fog, nodes are organized in hierarchical order as shown in Fig. 1. Here, Fog nodes are classified into two categories; *Fog Gateway Nodes (FGN)* and *Fog Computational Nodes (FCN)* [31]

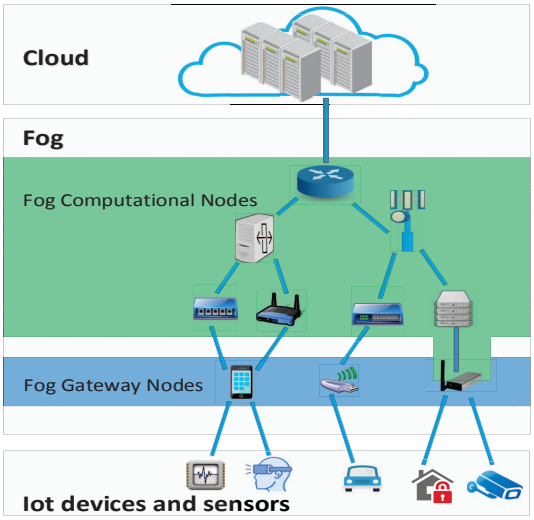

Figure 1: Organization of Fog

The lowest level of Fog nodes, known as Fog Gateway Nodes (FGNs), reside closer to the users. Through FGN, IoT devices and associate applications get subscribed to Fog environment for being monitored, placed and executed. The upper level Fog nodes, called as Fog Computational Nodes (FCNs), provide resources to the applications for processing and analyzing the data signals. According to OpenFog Consortium [28], there exists differences in computation intensity and resource capacity between FGNs and FCNs. However, we assume, each FGN has minimum ability to perfrom its assigned operations during QoE-aware application placement. Moreover, by applying existing Fog computing standards [32][33], each Fog node can offer RESTful services or Application Program Interface to the applications for querying and provisioning computation facilities. Fog infrastructure providers can apply port knocking, privileged port authentication, attribute-based encryption and other techniques to secure the communication daemon running on different Fog nodes for receiving requests and responses. Due to such security concerns, each node gets accessible to only a

set of Fog nodes. We assume that a Fog node maintains rapid and dynamic communication with all of its accessible nodes through efficient protocols such as CoAP and SNMP [34].

However, distance between Fog node and IoT devices in hierarchical setting is reflected through the round-trip delay of the data signals. Besides, computation capability and resource availability of lower level Fog nodes are less compared to that of upper level Fog nodes. There also exist diversity among Fog nodes of the same level. Thus, in the system, heterogeneity of the Fog nodes in capacity and efficiency always gets intensified.

5.3. Architecture of Fog nodes

5.3.1. Fog Computational Nodes (FCNs)

Extending the OpenFog Consortium reference architecture [28], we assume that a FCN is composed of three components; *Controller Component*, *Computational Component* and *Communication Component* as depicted in Fig. 2.

Computational Component is equipped with resources (e.g. CPU, memory, bandwidth, etc.) to run different applications. In Computational Component, resources are virtualized among *Micro Computing Instances, (MCI)*, where the applications are assigned for execution [2]. Additional resources for an MCI can be dynamically provisioned from either un-allocated resources or other MCIs without degrading the service quality. All configured MCIs in a FCN operate independently. Communication Component serves traditional networking functionalities like routing, packet forwarding, etc. Controller Component is responsible for monitoring and managing the overall activities of Computational Component and Communication Component. In Controller Component, there is data container that stores meta-data regarding the running applications and Status Metric parameters of the MCIs. In Controller Component, we propose a *Capacity Scoring Unit* to determine a capacity index for each MCI based on associate Status Metric parameters so that MCIs can be ranked according to their competence.

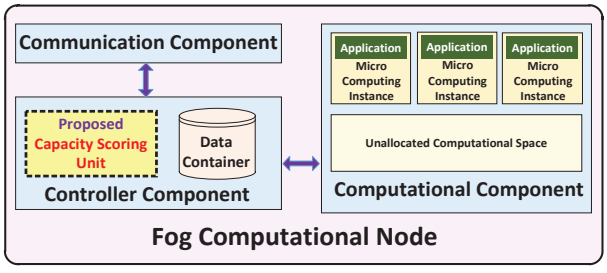

Figure 2: Architecture of a Fog Computational Node

5.3.2. Fog Gateway Nodes (FGNs)

User's premises equipment (set top boxes, cable modems) and hand-held devices (tablets, smart phones) are well suited to be used as FGNs. Extending the concept of IoT-gateway [35], a general architecture of FGNs is represented in Fig. 3. Sometimes, like FCNs, FGNs facilitate the computation of incoming data signals from IoT devices. For a particular Fog-enabled IoT system, we assume corresponding FGNs run the

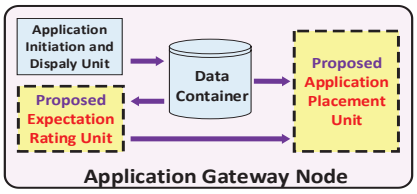

Figure 3: Architecture of a Fog Gateway Node

Client Module and assists to place the subsequent module to upper level FCNs. In this approach, at first connections between IoT devices and FGNs are established. The *Application Initiation Unit* of FGNs initiates the Client Module, through which a user conveys expectations regarding the application to FGNs. At FGN, the capacity index of MCIs at upper level FCNs are obtained through RESTful services and kept in a data container. In addition, the data container stores QoS attributes and user Expectation Metric regarding the applications for further processing. However, in FGN, we propose inclusion of two separate units named *Expectation Rating Unit* and *Application Placement Unit*. For each application placement request, Expectation Rating Unit calculates a priority value by taking user Expectation Metric into account. Besides, the Application Placement Unit of FGN conducts mapping of applications to suitable Fog instances based on the priority value of application placement requests and the capacity index of MCIs respectively.

Relevant notations and definitions used in system model and problem formulation are represented in Table 3.

Table 3: Notations

Symbol	Definition
M	Set of all FGNs.
N	Set of all FCNs.
A_m	Set of all application placement requests in FGN $m \in M$.
I_n	Set of all MCIs in FCN $n \in N$.
ω	Access rate parameter in Expectation Metric.
γ	Resource requirement parameter in Expectation Metric.
λ	Processing time parameter in Expectation Metric.
Ω	Round trip time parameter in Status Metric.
Γ	Resource availability parameter in Status Metric.
Λ	Processing speed parameter in Status Metric.
E_{a_m}	Expectation Metric for application $a \in A_m$.
S_{i_n}	Status Metric for instance $i \in I_n$.
η_{a_m}	RoE of application $a \in A_m$.
υ_{a_m}	Data signal size for $a \in A_m$.
τ_{i_n}	CCS of instance $i \in I_n$.
$U_x^{a_m}$	Expectation (value) of parameter x for application $a \in A_m$; $x \in \{\omega, \gamma, \lambda\}$
$V_y^{i_n}$	Status (value) of parameter y for instance $i \in I_n$; $y \in \{\Omega, \Gamma, \Lambda\}$
μ_x	Fuzzy membership function for any E_{a_m} parameter x.
μ'_y	Fuzzy membership function for any S_{i_n} parameter y.
F_r	Fuzzy output set for RoE calculation.
F'_c	Fuzzy output set for CCS calculation.
$\phi^{f_{a_m}}$	Singleton value for a Fuzzy output (RoE) $f_{a_m} \in F_r$ of $a \in A_m$,
$\Phi^{f'_{i_n}}$	Singleton value for a Fuzzy output (CCS) $f'_{i_n} \in F'_c$ of $i \in I_n$,
μ_r	Membership function for any Fuzzy output in RoE calculation.
μ'_c	Membership function for any Fuzzy output in CCS calculation
$z_{i_n}^{a_m} \in \{0, 1\}$	Equals to 1 if $a \in A_m$ mapped to $i \in I_n$, 0 otherwise.
$Q_{\delta,\zeta,\rho}$	QoS parameter for service delivery time, service cost, data signal loss rate respectively.
Ar, Rr, Pt	Fuzzy set for service access rate, resource requirement, processing time respectively.
Rtt, Ra, Ps	Fuzzy set for round-trip time, resource availability, processing speed respectively.

6. QoE-aware Application Placement

The flowchart of the proposed QoE-aware application placement policy is depicted in Fig. 4. The basic steps of the policy are to calculate a priority value named *Rating of Expectation (RoE)* of each application placement request based on the user expectation parameters, identify a capacity index named *Capacity Class Score (CCS)* of MCIs in FCNs according to the status parameters and ensure QoE maximized placement of the applications to competent MCIs using associate RoE and CCS values. In order to conduct the steps, Expectation Rating Unit, Application Placement Unit of FGNs and Capacity Scoring Unit of FCNs actively participate. Details of the steps are discussed in the following subsections.

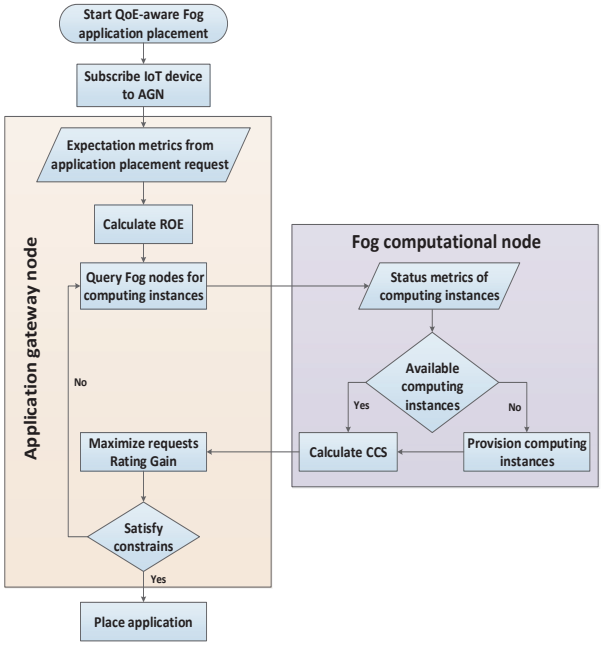

Figure 4: Flowchart of QoE-aware Application Placement

6.1. Calculation of Rating of Expectation (RoE)

After subscribing to a FGN m, the IoT device user apprises the $E_{a_m} \in \{U_\omega^{a_m}, U_\gamma^{a_m}, U_\lambda^{a_m}\}$ regarding an application a_m to the system through Application Initiation Unit. The E_{a_m} is stored in data container and forwarded to Expectation Rating Unit of FGN m. For every parameter in E_{a_m}, the range and unit of the numerical values are not the same. In order to simplify further calculation, the numerical value of each parameters in E_{a_m} is normalized to fall in the interval $[-1, 1]$ using the Eq. 1:

$$\overline{U_x^{a_m}} = 2\left(\frac{U_x^{a_m} - \alpha_x}{\beta_x - \alpha_x}\right) - 1 \tag{1}$$

Here, $U_x^{a_m}$ is the exact numerical value of parameter x within the range $[\alpha_x, \beta_x]$. For each parameter, $[\alpha_x, \beta_x]$ is set according to the scope for that parameter offered in the Fog environment. If numerical value of any Expectation Metric parameter does not fit within the associate range, the application will be discarded from placing in Fog. In this case, Cloud or other computing facilities can be pursued for placing the application. However, in Expectation Rating Unit, to calculate the η_{a_m} of the application from the normalized parameters in E_{a_m}, a Fuzzy logic based approach is followed. Fuzzy logic usually includes three phases; fuzzification, fuzzy inference and defuzzification.

In fuzzification, the normalized value $\overline{U_x^{a_m}}$ of any E_{a_m} parameter x is converted into equivalent fuzzy dimension by using associate membership function μ_x. In this work, membership functions of different Expectation Metric parameters form three distinct fuzzy sets over the normalized range $[-1, 1]$. The fuzzy sets are listed as:

- Access rate: $Ar \in \{Slow, Normal, Fast\}$
- Required resources: $Rr \in \{Small, Regular, Large\}$
- Processing time: $Pt \in \{Stringent, Moderate, Flexible\}$

Based on observations, the membership degree, $\mu_x(\overline{U_x^{a_m}})$ for any normalized value of parameter x on the respective fuzzy set is shown in Fig. 5.

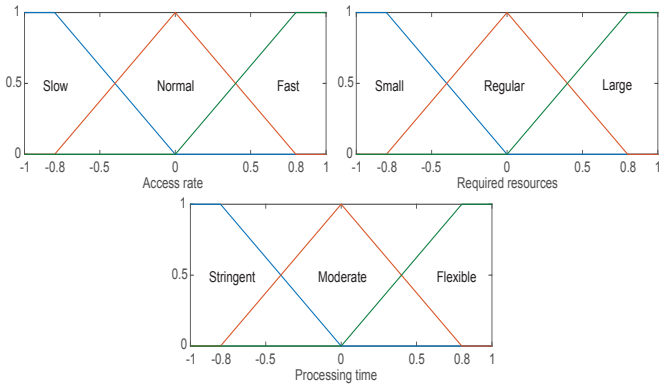

Figure 5: Membership Function of Expectation Metric Parameters

During fuzzy inference, fuzzy inputs are mutually compared to determine the corresponding fuzzy output. A set of fuzzy rules assist in this case. Here, the fuzzy output set for RoE is listed as; $F_r \in \{High, Medium, Low\}$ and the applied fuzzy rules are represented in Fig. 6. For instance, the rule to determine the fuzzy output $f_{a_m} \in F_r$ for application a_m with normal access rate, large resource requirements and moderate processing time expectation is interpreted as:

> *If access rate (ω) is normal or resource requirement (γ) is large or processing time expectation (λ) is moderate then RoE (f_{a_m}) is high.*

While setting the fuzzy rules, comparatively rigid expectation parameters (e.g. large resource requirements) are given higher weight. As a consequence, exact value of RoE for the requests become more aligned with the rigid expectation parameters compared to the relaxed parameters (e.g. normal service access rate, moderate processing time). Such characteristics of fuzzy rules ensure that even having two relaxed expectation parameters, the RoE of an application placement request can get increased due to a single rigid expectation parameter. Since the Expectation Metric parameters are independent and not closely coupled, the logical *or* operator is used in associate

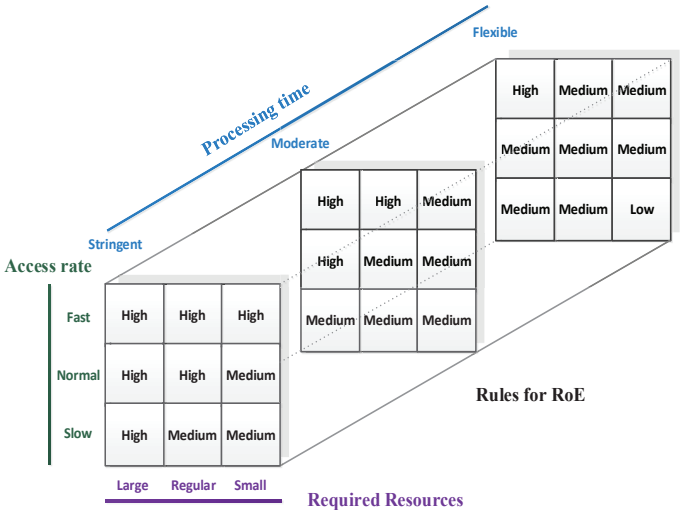

Figure 6: Fuzzy Rules for RoE Calculation

fuzzy rules to compare the Expectation Metric parameters and determine the fuzzy output. Generally, in logical *or* operation, the membership degree of fuzzy output is set according to the maximum membership degree of the compared parameters. For application a_m, the membership degree of fuzzy output, $\mu_r(f_{a_m})$ is determined using Eq. 2:

$$\mu_r(f_{a_m}) = max(\mu_\omega(\overline{U_\omega^{a_m}}), \mu_\gamma(\overline{U_\gamma^{a_m}}), \mu_\lambda(\overline{U_\lambda^{a_m}})) \tag{2}$$

In fuzzy inference, based on the Expectation Metric parameters, any j number of fuzzy rules can be triggered. In this case, membership degrees of associate fuzzy output are required to be combined together. Through defuzzification, the exact RoE of an application placement request is calculated from such combined membership degrees of the fuzzy output. A set of singleton values assist in this calculation. For each fuzzy output, f_{a_m} of application a_m, there is a singleton value $\phi^{f_{a_m}}$ that refers to the maximum rating of the application for that fuzzy output. The singleton values are determined in such a way so that logical distinction of different fuzzy output becomes clearly visible. For defuzzification, we have used the discrete center of gravity equation as shown in Eq. 3.

$$\eta_{a_m} = \frac{\sum_{k=1}^{k=j} \mu_r(f_{a_m}^k) \times \phi_k^{f_{a_m}}}{\sum_{k=1}^{k=j} \mu_r(f_{a_m}^k)} \tag{3}$$

η_{a_m} is the exact RoE for application a_m obtained by applying Fuzzy logic on different parameters of E_{a_m}. Later, η_{a_m} is used by Application Placement Unit to place the application in a suitable Fog computing instance.

6.2. Calculation of Capacity Class Score (CCS)

After calculating RoE of different application placement requests, FGN m queries accessible FCNs about available MCIs and associated CCS values. For each MCI i_n in a FCN n, the CCS is calculated in Capacity Class Scoring unit from the corresponding Status Metric, $S_{i_n} \in \{V_\Omega^{i_n}, V_\Gamma^{i_n}, V_\Lambda^{i_n}\}$. Like Expectation Metric parameters of application placement request, Status Metric parameters are heterogeneous in numeric range and unit. Therefore, using Eq. 4 different parameter y of S_{i_n} have been normalized to $[-1, 1]$.

$$\overline{V_y^{i_n}} = 2\left(\frac{V_y^{i_n} - \alpha'_y}{\beta'_y - \alpha'_y}\right) - 1 \tag{4}$$

The exact numeric value $V_y^{i_n}$ of parameter y remains with in the range of $[\alpha'_y, \beta'_y]$. The range is set according to the capacity of Fog environment for that parameter. In Capacity Scoring Unit, to calculate the CCS of instances based on multiple status parameters, another Fuzzy logic based approach is applied.

For parameter y, the membership degree of normalized $\overline{V_y^{i_n}}$ value to associate fuzzy sets is determined by the membership function μ'_y. The fuzzy input sets for different parameters of Status Metric are listed as:

- Round trip time: $Rtt \in \{Short, Typical, Lengthy\}$
- Resource availability: $Ra \in \{Poor, Standard, Rich\}$
- Processing speed: $Ps \in \{Least, Average, Intense\}$

Based on observations, the membership degree, $\mu'_y(\overline{V_y^{i_n}})$ for any normalized value of parameter y on the respective fuzzy set is shown in Fig. 7.

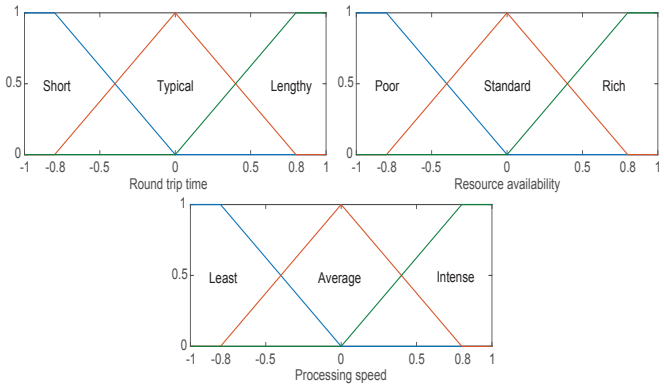

Figure 7: Membership Function of Status Metrics Parameters

The fuzzy rules applied for determining the fuzzy output for CCS calculation is represented in Fig. 8. Here, the associate fuzzy output set is listed as; $F'_c \in \{Higher, Medial, Lower\}$. For the instance with lengthy round trip delay (experienced from the FGN), standard resource (number of processing cores) availability and average per core processing speed, the rule to determine the fuzzy output, $f'_{i_n} \in F'_c$ is interpreted as:

> *If round-trip time (Ω) is lengthy and resource availability (Γ) is standard and processing speed (Λ) is average then CCS (f'_{i_n}) is lower.*

In fuzzy rules for calculating CCS, comparatively impediment status parameters (e.g. lengthy round trip delay) are given higher weight. As a consequence, exact CSS value of the instances highlight the limitations more, rather than the convenience (e.g. standard resource availability, average processing speed). Besides, in hierarchical orchestration, location of FCNs influences different parameters of Status Metric. Generally, MCIs of lower level FCNs support shorter amount of round

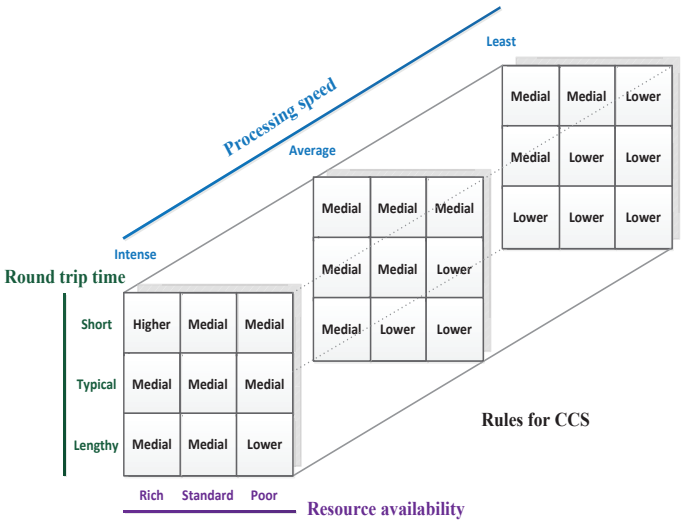

Figure 8: Fuzzy Rules for CCS Calculation

trip delay compared to the upper level FCNs. Conversely, MCIs of upper level FCNs are well-stuffed in processing capabilities than the lower FCNs. On the basis of location, the Status Metric parameters can be coupled with each other. Therefore, in fuzzy rules while comparing different Status Metric parameters, logical *and* operator has been used. In logical *and* operation, the membership degree of fuzzy output is set according to the minimum membership degree of the compared parameters. For computing instance i_n, the membership degree of fuzzy output, $\mu'_c(f'_{i_n})$ is determined using Eq. 5:

$$\mu'_c(f'_{i_n}) = min(\mu'_\Omega(\overline{V^{i_n}_\Omega}), \mu'_\Gamma(\overline{V^{i_n}_\Gamma}), \mu'_\Lambda(\overline{V^{i_n}_\Lambda})) \tag{5}$$

In order to calculate the exact CCS τ_{i_n} of the instance i_n, the membership degrees of the output, generated by triggering any j number of fuzzy rules, are combined together using the discrete center of gravity equation as shown in Eq. 6.

$$\tau_{i_n} = \frac{\sum_{k=1}^{k=j} \mu'_c(f'^k_{i_n}) \times \Phi_k^{f'_{i_n}}}{\sum_{k=1}^{k=j} \mu'_c(f'^k_{i_n})} \tag{6}$$

Here, the singleton value $\Phi^{f'_{i_n}}$ refers to the maximum score of an instance for the fuzzy output f'_{i_n}. The exact CCS of the instance τ_{i_n}, obtained through the aforementioned fuzzy approach, is forwarded to the querying FGN to conduct the following application placement steps.

6.3. Mapping of applications to Fog instances

The product of RoE of an application and CCS of a computing instance is called *Rating Gain* for placing the application on that instance. The mapping of applications to computational instance is done in such a way so that total Rating Gain of the applications get maximized. The maximum Rating Gain promotes the QoE-aware placement of the applications. The high RoE of the applications denotes the high combined intensity of associate Expectation Metric parameters. Similarly, the higher CCS refers to the higher capability of the instances to meet different user expectations even within the limitations. Both RoE and CCS are calculated under identical environment variables that enhances resemblance among the values. Since RoE of an application is the representative parameter for all of its expectation parameters, maximized Rating Gain of that application ensures the best possible convergence of the expectation parameters to corresponding status parameters of the instances. As a consequence, the possibility to manage Fog facilities (service accessibility, computational resources, application runtime), without degrading the user expectations, increases and the QoE regarding the application gets optimized.

In a FGN m, the mapping of applications to computing instances is conducted in the Application placement unit through the following multi-constraint objective function:

$$\max \sum_{\forall a_m \in A_m} \sum_{\forall n \in N} \sum_{\forall i_n \in I_n} z_{i_n}^{a_m} (\eta_{a_m} \times \tau_{i_n}) \tag{7}$$

subject to,

$$\sum_{a_m \in A_m} z_{i_n}^{a_m} = 1; \forall n \in N, \forall i_n \in I_n \tag{8}$$

$$\delta_{a_m} \leq Q_\delta \tag{9}$$

$$\zeta_{a_m} \leq Q_\zeta \tag{10}$$

$$\rho_{a_m} \leq Q_\rho \tag{11}$$

The objective function in Eq. 7 maximizes the Rating Gain for all application placement requests received by the FGN that subsequently enhances the overall user QoE. The constraint in Eq. 8 ensures one to one mapping between applications and instances. Besides, constraints in Eq. 9, Eq. 10 and Eq. 11 maintains the QoS of the application in terms of service delivery time, service cost and packet loss rate respectively. If a FGN fails to arrange constraint-satisfied placement of the applications, it re-queries the nodes for further instances.

The formulated objective function is a decentralized optimization problem. When application placement requests are submitted to a FGN, the optimization problem is solved and placement of the applications are conducted. By using any integer programming solver e.g. SCIP [36], the FGN can solve this multi-constraint optimization problem. To solve the optimization problem, FGN considers a local view of the Fog system. Due to location, the probability of receiving large number of application placement requests by a FGN on a particular time is low. Therefore, the optimization problem is less prone to be an NP-hard problem.

6.4. Rationality of the Applied Techniques

In this paper, we have used Fuzzy logic to determine the Rating of Expectation of application placement requests based on multiple user expectation parameters and Capacity Class Score of Fog instances according to their different status parameters. In a real-time system, where the dominance of multiple parameters is significant, Fuzzy logic based reasoning is considered among the best possible solutions. Fuzzy logic and its mathematical implication are simple and easy to understand. It has a

great potential to manage uncertain and linguistic information and can assist efficiently in converting qualitative data to quantitative data [37]. By tuning associated Fuzzy sets and rules, Fuzzy logic based solutions can be scalable according to context of the system. It requires less amount of data to train the system for future operations. Besides, in a Fuzzy logic-enabled system, stable results can be determined very quickly [38].

After determing RoE and CCS of application plcement request and Fog instances respectively, we have applied a multi-constraint single objective optimization technique on them to maximize the QoE Gain and deploy the applications according to the solution. Single-objective multi-constraint optimization problem often acts linearly and can be solved using any light-weight optimization solver within a shorter period of time [36].

However, rather than using Fuzzy logic and single objective optimization, in the proposed policy, multi-objective optimization can be applied. To conduct multi-objective optimization, often huge computational effort is required [39]. In most of the cases, multi-objective optimization problem is designed to meet a particular scenario which makes them less adaptive and scalable. Besides, solving a multi-objective optimization problem is time consuming and complex. Therefore, in real-time environment like Fog, where computation is done in resource constrained nodes, solving a multi-objective optimization problem often gets obstructed and affect the stringent service requirements [40]. Considering these challenges of multi-objective optimization, we rather preferred to apply Fuzzy logic and single objective optimization in our proposed policy.

7. Illustrative Example

In order to numerically illustrate the basic steps of proposed QoE-aware application placement policy, we have considered a Fog environment as depicted in Fig. 9.

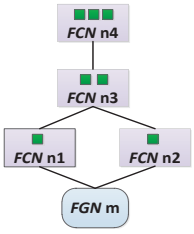

Figure 9: Illustrative Fog Environment

In this Fog environment, at any time t, the FGN m receives five application placement requests with $\upsilon_{a_m} = 1000 \sim 2000$ instructions ($\forall a_m \in A_m$). The scope in the Fog for different expectation parameters of application placement requests is represented in Table. 4

Table 4: Scope of Expectation Parameters

Parameter	Value
$[\alpha_\omega, \beta_\omega]$	[2, 10] per sec
$[\alpha_\gamma, \beta_\gamma]$	[1, 8] CPU cores
$[\alpha_\lambda, \beta_\lambda]$	[30, 120] ms

The exact expectation parameters of the requests along with normalized values, degree of membership to different fuzzy sets and RoE are shown in Table. 5. Here, the singleton values are set as; $\phi^{High} = 10$, $\phi^{Medium} = 5$, $\phi^{Low} = 2$ and the degree of membership to a particular fuzzy set is represented in the similar order of the set elements as listed in Section 6.1.

Table 5: Parameters of Application Placement Requests

Id	ω	γ	λ	η
app#1	$U^1_\omega = 2$ $\overline{U^1_\omega} = -1.0$ $\mu_\omega(\overline{U^1_\omega}) \rightarrow Ar$: {1.0, 0.0, 0.0}	$U^1_\gamma = 2$ $\overline{U^1_\gamma} = -0.71$ $\mu_\gamma(\overline{U^1_\gamma}) \rightarrow Rr$: {0.89, 0.11, 0.0}	$U^1_\lambda = 120$ $\overline{U^1_\lambda} = 1.0$ $\mu_\lambda(\overline{U^1_\lambda}) \rightarrow Pt$: {0.0, 0.0, 1.0}	5.69
app#2	$U^2_\omega = 5$ $\overline{U^2_\omega} = 0.0$ $\mu_\omega(\overline{U^2_\omega}) \rightarrow Ar$: {0.0, 1.0, 0.0}	$U^2_\gamma = 5$ $\overline{U^2_\gamma} = 0.14$ $\mu_\gamma(\overline{U^2_\gamma}) \rightarrow Rr$: {0.0, 0.83, 0.18}	$U^2_\lambda = 70$ $\overline{U^2_\lambda} = -0.11$ $\mu_\lambda(\overline{U^2_\lambda}) \rightarrow Pt$: {0.14, 0.86, 0.0}	6.69
app#3	$U^3_\omega = 3$ $\overline{U^3_\omega} = -0.75$ $\mu_\omega(\overline{U^3_\omega}) \rightarrow Ar$: {0.94, 0.06, 0.0}	$U^3_\gamma = 3$ $\overline{U^3_\gamma} = -0.43$ $\mu_\gamma(\overline{U^3_\gamma}) \rightarrow Rr$: {0.54, 0.46, 0.0}	$U^3_\lambda = 90$ $\overline{U^3_\lambda} = 0.33$ $\mu_\lambda(\overline{U^3_\lambda}) \rightarrow Pt$: {0.0, 0.59, 0.41}	6.21
app#4	$U^4_\omega = 7$ $\overline{U^4_\omega} = 0.25$ $\mu_\omega(\overline{U^4_\omega}) \rightarrow Ar$: {0.0, 0.69, 0.31}	$U^4_\gamma = 8$ $\overline{U^4_\gamma} = 1.0$ $\mu_\gamma(\overline{U^4_\gamma}) \rightarrow Rr$: {0.0, 0.0, 1.0}	$U^4_\lambda = 60$ $\overline{U^4_\lambda} = -0.33$ $\mu_\lambda(\overline{U^4_\lambda}) \rightarrow Pt$: {0.41, 0.59, 0.0}	7.28
app#5	$U^5_\omega = 8$ $\overline{U^5_\omega} = 0.5$ $\mu_\omega(\overline{U^5_\omega}) \rightarrow Ar$: {0.0, 0.38, 0.63}	$U^5_\gamma = 3$ $\overline{U^5_\gamma} = -0.43$ $\mu_\gamma(\overline{U^5_\gamma}) \rightarrow Rr$: {0.54, 0.46, 0.0}	$U^5_\lambda = 50$ $\overline{U^5_\lambda} = -0.56$ $\mu_\lambda(\overline{U^5_\lambda}) \rightarrow Pt$: {0.7, 0.3, 0.0}	7.03

The FGN m can query each FCNs of the system about MCIs. There are seven instances in the system (two at lower level, two at mid level and three at upper level). Different Status Metric parameters of the instances remains within the range shown in Table. 6.

Table 6: Scope of Status Parameters

Parameter	Value
$[\alpha_\Omega, \beta_\Omega]$	[100, 600] ms
$[\alpha_\Gamma, \beta_\Gamma]$	[1, 10] CPU cores
$[\alpha_\Lambda, \beta_\Lambda]$	[10, 70] TIPS

On time t, the exact status parameters of the instances along with normalized values, degree of membership to different fuzzy sets and CCS are shown in Table. 7. Here, the singleton values are set as; $\Phi^{Higher} = 10$, $\Phi^{Medial} = 5$, $\Phi^{Lower} = 2$ and the degree of membership to a particular fuzzy set is represented in the similar order of the set elements as listed in Section 6.2.

By applying Eq. 7 on RoE of the requests and CCS of the instances, the FGN m calculates the maximized Rating Gain of the applications. It also provides the optimal mapping of applications and instances. In this illustrative example we use SCIP solver to solve the optimization problem. The solution is represented in Table. 8. Here the constraints (Q_δ = 250~750 ms, Q_ζ = 0.1~0.15 $ per min, Q_ρ = 3~5 % data signals) are assumed to be met. Based on the optimization solution, after exploring the expectation and the status parameters (from Table. 5, Table. 7) of mapped applications and instances, it is found that, for almost every parameters, user expectations have been satisfied.

In the illustrative example, numeric values of Fog instances are extended from the literature [41][42]. The values explicitly

Table 7: Parameters of Computing Instances

Id	Ω	Γ	Λ	τ
ins#1	$\overline{V^1_\Omega} = 100$ $V^1_\Omega = -1.0$ $\mu'_\Omega(V^1_\Omega) \rightarrow Rtt$: $\{1.0, 0.0, 0.0\}$	$\overline{V^1_\Gamma} = 3$ $V^1_\Gamma = -0.56$ $\mu'_\Gamma(V^1_\Gamma) \rightarrow Ra$: $\{0.70, 0.30, 0.0\}$	$\overline{V^1_\Lambda} = 20$ $V^1_\Lambda = -0.67$ $\mu'_\Lambda(V^1_\Lambda) \rightarrow Ps$: $\{0.84, 0.16, 0.0\}$	3.41
ins#2	$\overline{V^2_\Omega} = 100$ $V^2_\Omega = -1.0$ $\mu'_\Omega(V^2_\Omega) \rightarrow Rtt$: $\{1.0, 0.0, 0.0\}$	$\overline{V^2_\Gamma} = 2$ $V^2_\Gamma = -0.78$ $\mu'_\Gamma(V^2_\Gamma) \rightarrow Ra$: $\{0.97, 0.03, 0.0\}$	$\overline{V^2_\Lambda} = 20$ $V^2_\Lambda = -0.67$ $\mu'_\Lambda(V^2_\Lambda) \rightarrow Ps$: $\{0.84, 0.16, 0.0\}$	2.62
ins#3	$\overline{V^3_\Omega} = 200$ $V^3_\Omega = -0.6$ $\mu'_\Omega(V^3_\Omega) \rightarrow Rtt$: $\{0.75, 0.25, 0.0\}$	$\overline{V^3_\Gamma} = 4$ $V^3_\Gamma = -0.33$ $\mu'_\Gamma(V^3_\Gamma) \rightarrow Ra$: $\{0.41, 0.59, 0.0\}$	$\overline{V^3_\Lambda} = 40$ $V^3_\Lambda = 0.0$ $\mu'_\Lambda(V^3_\Lambda) \rightarrow Ps$: $\{0.0, 1.0, 0.0\}$	4.50
ins#4	$\overline{V^4_\Omega} = 300$ $V^4_\Omega = -0.20$ $\mu'_\Omega(V^4_\Omega) \rightarrow Rtt$: $\{0.25, 0.75, 0.0\}$	$\overline{V^4_\Gamma} = 5$ $V^4_\Gamma = -0.11$ $\mu'_\Gamma(V^4_\Gamma) \rightarrow Ra$: $\{0.14, 0.86, 0.0\}$	$\overline{V^4_\Lambda} = 30$ $V^4_\Lambda = -0.33$ $\mu'_\Lambda(V^4_\Lambda) \rightarrow Ps$: $\{0.41, 0.59, 0.0\}$	3.79
ins#5	$\overline{V^5_\Omega} = 400$ $V^5_\Omega = 0.20$ $\mu'_\Omega(V^5_\Omega) \rightarrow Rtt$: $\{0.0, 0.75, 0.25\}$	$\overline{V^5_\Gamma} = 6$ $V^5_\Gamma = 0.11$ $\mu'_\Gamma(V^5_\Gamma) \rightarrow Ra$: $\{0.0, 0.86, 0.14\}$	$\overline{V^5_\Lambda} = 50$ $V^5_\Lambda = 0.33$ $\mu'_\Lambda(V^5_\Lambda) \rightarrow Ps$: $\{0.0, 0.59, 0.41\}$	4.64
ins#6	$\overline{V^6_\Omega} = 500$ $V^6_\Omega = 0.60$ $\mu'_\Omega(V^6_\Omega) \rightarrow Rtt$: $\{0.0, 0.25, 0.75\}$	$\overline{V^6_\Gamma} = 8$ $V^6_\Gamma = 0.56$ $\mu'_\Gamma(V^6_\Gamma) \rightarrow Ra$: $\{0.0, 0.30, 0.70\}$	$\overline{V^6_\Lambda} = 70$ $V^6_\Lambda = 1.0$ $\mu'_\Lambda(V^6_\Lambda) \rightarrow Ps$: $\{0.0, 0.0, 1.0\}$	5.00
ins#7	$\overline{V^7_\Omega} = 500$ $\overline{V^7_\Omega} = 0.60$ $\mu'_\Omega(V^7_\Omega) \rightarrow Rtt$: $\{0.0, 0.25, 0.75\}$	$\overline{V^7_\Gamma} = 6$ $\overline{V^7_\Gamma} = 0.11$ $\mu'_\Gamma(V^7_\Gamma) \rightarrow Ra$: $\{0.0, 0.86, 0.14\}$	$\overline{V^7_\Lambda} = 60$ $\overline{V^7_\Lambda} = 0.67$ $\mu'_\Lambda(V^7_\Lambda) \rightarrow Ps$: $\{0.0, 0.16, 0.84\}$	4.74

represent the computational limitations of Fog instances compared to the Cloud instances. The illustrative example also exhibits how our proposed policy/model can deal with the lower computational capabilities of different Fog instances and distinguish them through CCS while meeting the well-known features of the Fog environment. In addition, the configuration of FGN m used in this example is *Intel(R) Core(TM)2 Duo CPU E6550 @ 2.33GHz 2GB DDR2 RAM*. On this configuration, FGN m takes $20ms$ to calculate RoE of the application placement requests and $8ms$ to solve the optimization problem.

Table 8: Solution of the Optimization Problem

application	instance	Rating Gain
app#1	ins#4	21.57
app#2	ins#5	31.04
app#3	ins#3	27.95
app#4	ins#6	36.40
app#5	ins#7	33.32

8. Performance Evaluation

The proposed QoE-aware application palcement policy is compared with different QoS and QoE-aware policies. The QoS-aware application placement policy in [3] meets execution deadline of the applications. Among the QoE-aware policies, Cloud-Fog [8] optimizes service coverage, response time and network congestion whereas MeFoRE [9] ensures efficient resource estimation based on user's feedback. However, while comparing the proposed policy with the aforementioned policies; network congestion, amount of allocated resources, reduced processing time and percentage of QoS-satisfied data signals are considered as performance metrics.

8.1. Simulation Environment

To evaluate the proposed policy, a Fog environment is simulated using iFogSim [13]. iFogSim is an extention of *CloudSim* [43] framework which has been widely used for simulating different computing paradigms. In iFogSim, varying configuration and count of FCNs, different number of applications have been placed. In simulation, we consider synthetic workload as compatible real workload for the proposed application placement policy is not currently available. For each application, the workload includes computation for different tasks such as data filtration, analysis and event processing. Table 9 represents the details of workload and system parameters.

Table 9: Simulation Parameters

Parameter	Value
Expectation Metrics:	
Access rate	2-10 per sec
Resource requirement	1-8 CPU cores
Processing time	30-120 ms
Status Metrics:	
Round trip time	100-600 ms
Resource availability	1-10 CPU cores
Processing speed	10-70 TIPS
Applications service delivery deadline	250 - 750 ms
Data signal loss rate of the network	3-5 %
Service cost	0.1-0.15 $ per min
Number of accessible FCN per FGN	4-10
Data signal size	1000-2000 instructions

8.2. Experiment and Discussion

The applicability of the proposed QoE-aware application placement policy has been validated through simulation experiments on network congestion, resource allocation, processing time, application placement time and QoS satisfaction rate. To demonstrate the potentiality of the proposed QoE-aware policy in handling network congestion, we have calculated average *Network Relaxation Ratio (NRR)* for the applications placed by any FGN m at time t using Eq. 12:

$$avg(NRR_m) = \frac{1}{|A^t_m|} \sum_{\forall a_m \in A_m} \frac{2}{U^{a_m}_\omega \times V^{i_n}_\Omega} \tag{12}$$

subject to, $z^{a_m}_{i_n} = 1$ and $V^{i_n}_\Omega$ in second.

Any value of $NRR > 1$ for an application refers to less possibility of network congestion. For example, an application with $U^{a_m}_\omega = 2$ per second, in every service access receives a data signal to process. Intermediate delay between receiving two data signals for that application will be 0.5 sec. Let us assume the application has been placed in an instance where $V^{i_n}_\Omega = 0.3$ sec. In that case, roughly even after propagating a data signal to the application, the network will remain free upto 0.35 sec and the *NRR* for that application will be 3.33. As a consequence, there will be lesser possibility of network congestion. The proposed QoE-aware policy actively participates in relaxing network (Fig. 10). As the number of applications increase, the data transmission load over the network increases and $avg(NRR)$ declines, which is natural. However, compared

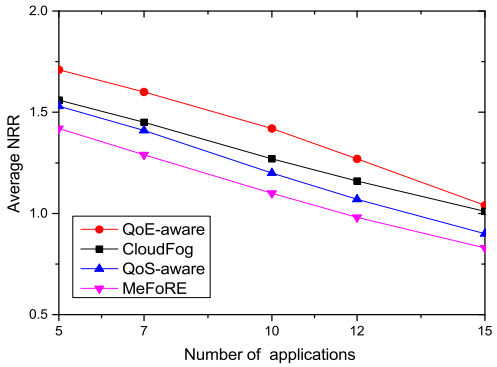

Figure 10: Network Relaxation Ratio vs Number of Applications

to other approaches the declining rate of $avg(NRR)$ in the QoE-aware policy is lower. Among other approaches Cloud-Fog performs well as it discards data signals of increasing applications to mitigate network congestion. MeFoRE prefers application placement to upper level FCNs to meet increasing user demand based on the feedback that eventually increases data transmission time and chance of network congestion. Similarly, in the QoS-aware placement, additional time is required to maintain intra-inter communication within the colonies that adversely affect network flexibility.

Resource Gain (RG) of applications refers to the QoE of users in respect of resource consumption. Here, the average *RG* for the applications placed by FGN m at time t has been calculated using Eq. 13:

$$avg(RG_m) = \frac{1}{|A_m^t|} \sum_{\forall a_m \in A_m} \frac{V_\Gamma^{i_n}}{U_\gamma^{a_m}} \tag{13}$$

subject to, $z_{i_n}^{a_m} = 1$.

If an application with $U_\gamma^{a_m} = 2$ processing cores, is placed to a computing instance having $V_\Gamma^{i_n} = 3$ cores, the *RG* for the application will be 1.5. Any value of $RG > 1$ refers that by paying almost same cost, the user is consuming additional resources. The proposed QoE-aware policy ensures higher $avg(RG)$ for the application, although with the increasing number of applications, $avg(RG)$ declines (Fig. 11). Re-provisioning of the resources for additional applications contribute in this case. However, $avg(RG)$ in other policies are competitively low. Since Cloud-Fog changes operations on data according to the load, it can run the applications even with less resources compared to expectation. MeFoRE resists resource under-utilization that ultimately reduces $avg(RG)$. In QoS-aware policy, additional resources are only allocated when users ask for them, that affects the fixed-cost *RG* of the applications.

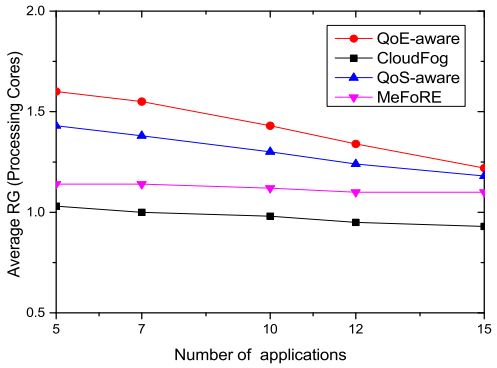

Figure 11: Resource Gain vs Number of Applications

In the experiments, reduction in processing time of the applications has been represented through *Processing Time Reduction Ratio (PTRR)*. average *PTRR* of the applications placed by FGN m at time t has been calculated using Eq. 14:

$$avg(PTRR_m) = \frac{1}{|A_m^t|} \sum_{\forall a_m \in A_m} \frac{U_\lambda^{a_m} \times V_\Lambda^{i_n}}{\upsilon_{a_m}} \tag{14}$$

subject to, $z_{i_n}^{a_m} = 1$ and $U_\lambda^{a_m}$ in second.

If an application ,with $U_\lambda^{a_m} = 0.12$ sec and $\upsilon_{a_m} = 1000$ instructions, is placed to a computing instance having $V_\Lambda^{i_n} = 30$ Thousand Instructions Per Second (TIPS), the *PTRR* for the application will be 3.6. Any value of $PTRR > 1$ ensures faster data process than the expectation. The QoE-aware policy increases $avg(PTRR)$ for the applications although it declines with the number of applications (Fig. 12). Compared to other policies, $avg(PTRR)$ in the proposed policy is much higher. Cloud-Fog maintains better $avg(PTRR)$ by discarding data signals although incentive based resource sharing during high computational load fails to retain the higher $avg(PTRR)$. MeFoRE increases $avg(PTRR)$ by iteratively placing applications in upper level FCN. It happens only when user feedback in respect of application processing time gets poor. The QoS-aware policy concentrates more on optimizing processing time in priority basis rather than maintaining higher $avg(PTRR)$ for all the applications.

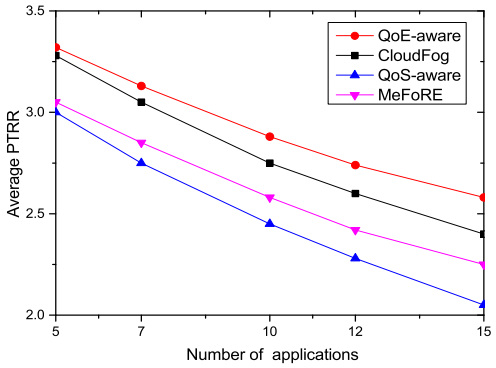

Figure 12: Processing Time Reduction Ratio vs Number of Applications

Fig. 13 represents the average application placement time in different approaches. In the proposed QoE-aware policy placement decision is taken at the FGN. Since necessary calculations in other entities can be done in parallel, the required time to place applications gets lower. With the increasing number of applications, placement time increases. Time to find the solution of the optimization problem for increasing number of

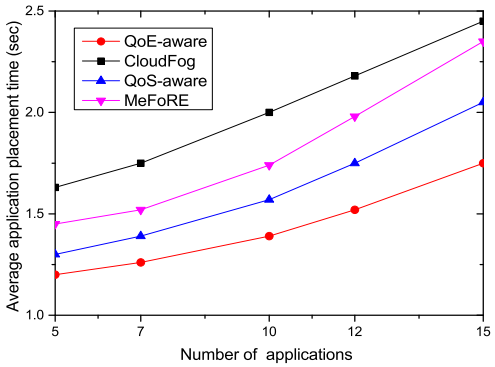

Figure 13: Application Placement Time vs Number of Applications

applications contributes in this case. However, in Cloud-Fog remote Cloud interferes in application placement, in MeFoRE iterative approach is used for placing application and in QoS-aware policy a controller node places the application over the Fog cells. All these facts adversely affect the application placement time.

The percentage of QoS -satisfied data signals in the proposed QoE-aware policy is higher as it considers multiple QoS parameters (cost, deadline, packet loss rate) while placing the applications (Fig. 14). In QoS-aware policy only deadline has been considered as the QoS metric. In that policy, for many data signals the deadline satisfied QoS can not be ensured when the overhead in controller node increases significantly. To maintain congestion free network and tolerable response delay, Cloud-Fog discards large number of data packets that eventually degrades QoS. Since MeFoRE prefers to place application in upper level FCNs, it often fails to maintain cost and deadline satisfied QoS requirement of the data signals.

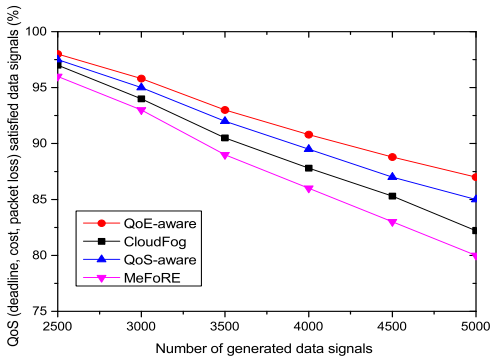

Figure 14: Percentage of QoS Satisfied Data Signals

In the QoE-aware policy three different aspects of QoE (service accessibility, resource affordability and service processing time) have been maintained simultaneously. Since, intensity of expectations for different applications are diversified, it is not possible to ensure maximized QoE gain (*NRR*, *RG*, *PTRR*) for every applications. Fig.15 represents that among the placed applications on different experiments, how many applications achieved the QoE gain in every aspect. According to the results almost 92% applications gets higher *PTRR*. In respect of *RG*

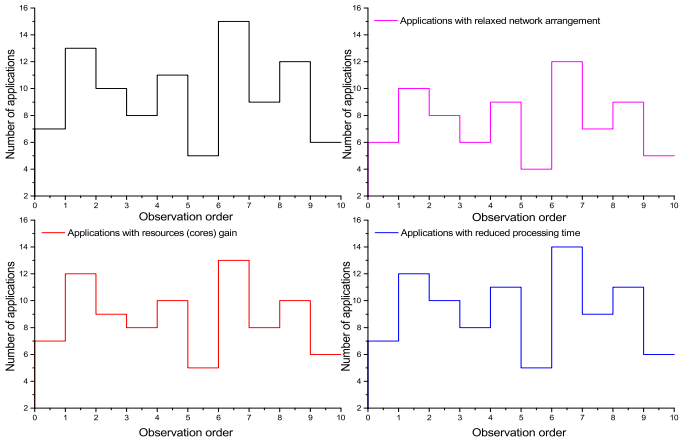

Figure 15: QoE Gain of Applications from Different Service Aspects

and *NRR*, this percentage belongs to 88% and 80% respectively.

9. Conclusions and Future Work

As a computing paradigm, Fog has a significant potential to support IoT applications. To exploit benefits of Fog, different application placement policies are required to be investigated. In this work, we have discussed about QoE-aware application placement in Fog that considers both expectations of user regarding the applications and status of the Fog computing instances while placing the applications. We have applied two separate fuzzy logic models to simplify the mapping of applications to compatible instances by calculating application's Rating of Expectations and Capacity Class Score of the instances. The developed linear optimization problem ensures the best convergence between user expectations and scope within the Fog environment that eventually maximizes the QoE. The simulation results demonstrate that our proposed policy performs well in attaining the objective compared to the other policies.

In future, we aim to evaluate performance of the proposed policy in real Fog environment. In this case, Raspberry Pi/personal computer/Cisco IOx equipment can be used as Fog Computational Nodes and hand-held smartphones can be employed as Fog Gateway Nodes. Due to recent technological advancements, these devices are now capable of running Fuzzy inference engines, lightweight optimization solvers, RESTful services and virtualizing resources through Docker (Containers) or VMWare (Virtual Machines). With the help of these features, we can expect to translate the proposed policy in real-environment. Besides, we intent to explore different application models and their cost-based placement in Fog.

References

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

[2] R. Mahmud, K. Ramamohanarao, R. Buyya, Fog Computing: A Taxonomy, Survey and Future Directions, in: D. M. Beniamino, Y. Laurence, K.-C. Li, E. Antonio (Eds.), Internet of Everything: Algorithms, Methodologies, Technologies and Perspectives, Springer, 2017.

[3] O. Skarlat, M. Nardelli, S. Schulte, S. Dustdar, Towards QoS-aware Fog Service Placement, in: Proc. of the First IEEE International Conference on Fog and Edge Computing, ICFEC '17, IEEE, 2017.

[4] A. Brogi, S. Forti, QoS-aware Deployment of IoT Applications through the Fog, IEEE Internet of Things Journal PP (99) (2017) 1–10.

[5] M. Taneja, A. Davy, Resource Aware Placement of Data Analytics Platform in Fog Computing, Procedia Computer Science 97 (2016) 153 – 156, 2nd International Conference on Cloud Forward: From Distributed to Complete Computing.

[6] E. Saurez, K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, Incremental Deployment and Migration of Geo-distributed Situation Awareness Applications in the Fog, in: Proc. of the 10th ACM International Conference on Distributed and Event-based Systems, DEBS '16, ACM, 2016, pp. 258–269.

[7] K. u. R. Laghari, N. Crespi, B. Molina, C. E. Palau, QoE Aware Service Delivery in Distributed Environment, in: Proc. of the IEEE Workshops of International Conference on Advanced Information Networking and Applications, WAINA '11, IEEE, 2011, pp. 837–842.

[8] Y. Lin, H. Shen, Cloud Fog: Towards High Quality of Experience in Cloud Gaming, in: Proc. of the 44th International Conference on Parallel Processing, ICPP '15, IEEE, 2015, pp. 500–509.

[9] M. Aazam, M. St-Hilaire, C. H. Lung, I. Lambadaris, MeFoRE: QoE-based Resource Estimation at Fog to Enhance QoS in IoT, in: Proc. of the 23rd International Conference on Telecommunications, ICT '16, IEEE, 2016, pp. 1–5.

[10] T. Hofeld, R. Schatz, S. Egger, SOS: The MOS is not Enough!, in: Proc. of the Third International Workshop on Quality of Multimedia Experience, QoMEX '11, IEEE, 2011, pp. 131–136.

[11] L. Li, M. Rong, G. Zhang, An Internet of Things QoE Evaluation Method-based on Multiple Linear Regression Analysis, in: Proc. of the 10th International Conference on Computer Science Education, ICCSE '15, IEEE, 2015, pp. 925–928.

[12] M. Fiedler, T. Hossfeld, P. Tran-Gia, A Generic Quantitative Relationship between Quality of Experience and Quality of Service, IEEE Network 24 (2) (2010) 36–41.

[13] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, 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, 2017.

[14] M. R. Mahmud, M. Afrin, M. A. Razzaque, M. M. Hassan, A. Alelaiwi, M. Alrubaian, Maximizing Quality of Experience through Context-aware Mobile Application Scheduling in Cloudlet Infrastructure, Software: Practice and Experience 46 (11) (2016) 1525–1545, spe.2392.

[15] X. Zhou, M. Sun, Y. Wang, X. Wu, A New QoE-driven Video Cache Allocation Scheme for Mobile Cloud Server, in: Proc. of the 11th International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness, QSHINE '15, IEEE, 2015, pp. 122–126.

[16] S. Peng, J. O. Fajardo, P. S. Khodashenas, B. Blanco, F. Liberal, C. Ruiz, C. Turyagyenda, M. Wilson, S. Vadgama, QoE-Oriented Mobile Edge Service Management Leveraging SDN and NFV, Mobile Information Systems 2017.

[17] S. Dutta, T. Taleb, P. A. Frangoudis, A. Ksentini, On-the-fly Qoe-aware Transcoding in the Mobile Edge, in: Proc. of the Global Communications Conference, GLOBECOM '16, IEEE, 2016, pp. 1–6.

[18] X. Lin, W. Wu, Y. Zhu, T. Qiu, Z. Mi, SARS: A Novel QoE-based Service-aware Resource Scheduling Scheme in Wireless Network, in: Proc. of the IEEE International Conference on Ubiquitous Wireless Broadband, ICUWB '16, IEEE, 2016, pp. 1–4.

[19] A. Anand, G. de Veciana, Measurement-based Scheduler for Multi-class QoE optimization in wireless networks, in: Proc. of the 36th IEEE Conference on Computer Communications, INFOCOM '17, IEEE, 2017, pp. 1–9.

[20] N. Iotti, M. Picone, S. Cirani, G. Ferrari, Improving Quality of Experience in Future Wireless Access Networks through Fog Computing, IEEE Internet Computing 21 (2) (2017) 26–33.

[21] E. Recommendation, Definitions of Terms Related to Quality of Service (2008).

[22] F. ITU-T IPTV, Definition of Quality of Experience (QoE) (2007).

[23] M. Varela, L. Skorin-Kapov, T. Ebrahimi, Quality of Service versus Quality of Experience, in: Quality of experience, Springer, 2014, pp. 85–96.

[24] P. Merino, M. G. Martini, R. Schatz, L. Skorin-Kapov, M. Varela, Improving QoS and QoE for Mobile Communications, Journal of Computer Networks and Communications 2013 (2013) 1–2.

[25] H. Nam, K.-H. Kim, H. Schulzrinne, Qoe Matters More than QoS: Why People Stop Watching Cat Videos, in: Proc. of the 35th IEEE Conference on Computer Communications, INFOCOM '16, IEEE, 2016, pp. 1–9.

[26] J. K. Zao, T. T. Gan, C. K. You, S. J. R. Méndez, C. E. Chung, Y. Te Wang, T. Mullen, T. P. Jung, Augmented Brain Computer Interaction-based on Fog Computing and Linked Data, in: Proc. of the International Conference on Intelligent Environments, IE '14, IEEE, 2014, pp. 374–377.

[27] P. Hu, H. Ning, T. Qiu, Y. Zhang, X. Luo, Fog Computing-based Face Identification and Resolution Scheme in Internet of Things, IEEE Transactions on Industrial Informatics 13 (2017) 1910–1920.

[28] O. C. Architecture Working Group, OpenFog Reference Architecture for Fog Computing, OPFRA001 20817 (2017) 162.

[29] S. Yangui, P. Ravindran, O. Bibani, R. H. Glitho, N. B. Hadj-Alouane, M. J. Morrow, P. A. Polakos, A Platform-as-a-Service for Hybrid Cloud/Fog Environments, in: Proc. of the IEEE International Symposium on Local and Metropolitan Area Networks, LANMAN '16, IEEE, 2016, pp. 1–7.

[30] M. Taneja, A. Davy, Resource-aware Placement of IoT Application Modules in Fog-Cloud Computing Paradigm, in: Proc. of the IFIP/IEEE Symposium on Integrated Network and Service Management, IM '15, IEEE, 2017, pp. 1222–1228.

[31] E. M. Tordera, X. Masip-Bruin, J. García-Almiñana, A. Jukan, G.-J. Ren, J. Zhu, Do We All Really Know What a Fog Node is? Current Trends Towards an Open Definition, Computer Communications 109 (2017) 117 – 130.

[32] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, B. Koldehofe, Mobile fog: A Programming Model for Large-scale Applications on the Internet of Things, in: Proc. of the 2nd ACM SIGCOMM workshop on Mobile cloud computing, ACM, 2013, pp. 15–20.

[33] C. Chang, S. N. Srirama, R. Buyya, Indie Fog: An Efficient Fog-Computing Infrastructure for the Internet of Things, Computer 50 (9) (2017) 92–98.

[34] M. Slabicki, K. Grochla, Performance Evaluation of CoAP, SNMP and NETCONF Protocols in Fog Computing Architecture, in: Proc. of the IEEE/IFIP Network Operations and Management Symposium, NOMS '16, IEEE, 2016, pp. 1315–1319.

[35] T. Zachariah, N. Klugman, B. Campbell, J. Adkins, N. Jackson, P. Dutta, The Internet of Things has a Gateway Problem, in: Proc. of the 16th International Workshop on Mobile Computing Systems and Applications, HotMobile '15, ACM, 2015, pp. 27–32.

[36] T. Achterberg, SCIP: Solving Constraint Integer Programs, Mathematical Programming Computation 1 (1) (2009) 1–41.

[37] E. Mamdani, S. Assilian, An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller, International Journal of Man-Machine Studies 7 (1) (1975) 1 – 13.

[38] A. Ansari, A. A. Bakar, A Comparative Study of Three Artificial Intelligence Techniques: Genetic Algorithm, Neural Network, and Fuzzy Logic, on Scheduling Problem, in: Proc. of the 4th International Conference on Artificial Intelligence with Applications in Engineering and Technology, ICAIET '14, IEEE, 2014, pp. 31–36.

[39] G. Chiandussi, M. Codegone, S. Ferrero, F. E. Varesio, Comparison of Multi-objective Optimization Methodologies for Engineering Applications, Computers & Mathematics with Applications 63 (5) (2012) 912–942.

[40] M. Helbig, K. Deb, A. Engelbrecht, Key Challenges andFuture Directions of Dynamic Multi-objective Optimisation, in: Proc. of the IEEE Congress on Evolutionary Computation, CEC '16, IEEE, 2016, pp. 1256–1261.

[41] Y. Elkhatib, B. Porter, H. B. Ribeiro, M. F. Zhani, J. Qadir, E. Rivière, On Using Micro-clouds to Deliver the Fog, IEEE Internet Computing 21 (2) (2017) 8–15.

[42] P. Bellavista, A. Zanni, Feasibility of Fog Computing Deployment Based on Docker Containerization over Raspberrypi, in: Proceedings of the 18th International Conference on Distributed Computing and Networking, ACM, 2017, p. 16.

[43] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, R. Buyya, CloudSim: a Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms, Software: Practice and experience 41 (1) (2011) 23–50.