Quality of Service (QoS)-aware Application Management in Fog Computing Environments

Md Redowan Mahmud

Submitted in total fulfilment of the requirements of the degree of

Doctor of Philosophy

School of Computing and Information Systems
THE UNIVERSITY OF MELBOURNE

January 2020

ORCID: 0000-0003-0785-0457

Quality of Service (QoS)-aware Application Management in Fog Computing Environments

Md Redowan Mahmud
Principal Supervisor: Prof. Rajkumar Buyya
Co-Supervisor: Prof. Ramamohanarao Kotagiri

Abstract

In recent years, the Internet of Things (IoT) paradigm is being rapidly adopted in the creation of applications for various smart-city, healthcare, Industry 4.0, and Agtech-based Cyber-Physical Systems (CPS). Usually, the IoT-enabled CPSs reside at a multi-hop distance from the Cloud datacentres. As a consequence, the Cloud-centric execution of IoT applications often fails to meet their Quality of Service (QoS) requirements in real-time. *Fog computing*, an extension of Cloud at the edge network, can execute the IoT applications closer to the data sources. Thus, it can improve the application service delivery time and reduce network congestion. However, the Fog computing nodes are highly distributed, heterogeneous, and most of them are constrained in resources and spatial sharing. Therefore, without efficient management of applications, it is complicated to harness the capabilities of Fog computing for different IoT-driven use cases.

Application management is as an integral part of computing resource management. It can be ensured by finding suitable placement options for the applications within the computing infrastructure. In IoT-enabled CPSs, different entities including, applications, Fog nodes, IoT devices, users, and service providers, continuously interact with each other. This thesis focuses on application placement in Fog environments considering *a.* the characteristics of the applications, *b.* the communication delay among the Fog nodes, *c.* the context of the IoT devices, *d.* the service expectations of the users, and *e.* the operational cost of the providers. It demonstrates how the placement of applications from the perspectives of different system entities can improve the application's QoS, the user's Quality of Experience (QoE), and the provider's profit. This thesis advances the state-of-the-art by making the following contributions:

1. A comprehensive taxonomy and literature review on application management approaches in Fog computing environments.

2. An application characteristics-driven model that facilitates application classification and selection for Fog-based placement at the gateway level.

3. A latency-aware application management policy that deals with the service delivery deadline and the inter-nodal communication delay simultaneously while placing the applications over distributed Fog nodes.

4. A context-aware application management policy that optimizes the service time of applications by coordinating the sensing frequency and data size of IoT devices with the capacity of Fog nodes.

5. A QoE-aware application management policy that prioritizes the placement of applications in Fog environments based on user expectations.

6. A pricing model for integrated Fog-Cloud environments that enhances the profit of providers for executing the applications in the proximity of end-users.

Declaration

This is to certify that

1. the thesis comprises only my original work towards the PhD,
2. due acknowledgement has been made in the text to all other material used,
3. the thesis is less than 100,000 words in length, exclusive of tables, maps, bibliographies and appendices.

Md Redowan Mahmud, January 2020

Preface

Main Contributions

This thesis research has been carried out in the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, The University of Melbourne. The main contributions of the thesis are discussed in Chapters 2-7 and are based on the following publications:

- **Redowan Mahmud**, Kotagiri Ramamohanarao, and Rajkumar Buyya, "Fog Computing: A Taxonomy, Survey and Future Directions", *Internet of Everything: Algorithms, Methodologies, Technologies and Perspectives*, B. DiMartino, K. Li, L. Yang, A. Esposito (eds), Pages: 103-130, ISBN 978-981-10-5860-8, Springer, Singapore, 2018.

- **Redowan Mahmud**, Kotagiri Ramamohanarao, and Rajkumar Buyya, "Application Management in Fog Computing Environments: A Taxonomy, Review and Future Directions", *ACM Computing Surveys* (in second revision).

- **Redowan Mahmud**, Kotagiri Ramamohanarao, and Rajkumar Buyya, "Edge Affinity-based Management of Applications in Fog Computing Environments", *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing (UCC'19)*, IEEE Press, Pages: 61-70, Auckland, New Zealand, December 2-5, 2019.

- **Redowan Mahmud**, Kotagiri Ramamohanarao, and Rajkumar Buyya, "Latency-aware Application Module Management for Fog Computing Environments", *ACM Transactions on Internet Technology (TOIT)*, Volume 19, No. 1, Article 9, Pages: 1-21, ISSN:1533-5399, ACM Press, New York, USA, January 2019.

- **Redowan Mahmud**, Adel Nadjaran Toosi, Kotagiri Ramamohanarao, and Rajkumar Buyya, "Context-aware Application Placement for Industry 4.0 in Fog Computing Environments", *IEEE Transactions on Industrial Informatics* (in press, DOI: 10.1109/TII.2019.2952412, accepted on November 5, 2019).

- **Redowan Mahmud**, Satish Narayana Srirama, Kotagiri Ramamohanarao, and Rajkumar Buyya, "Quality of Experience (QoE)-aware Placement of Applications in Fog Computing Environments", *Journal of Parallel and Distributed Computing (JPDC)*, Volume 132, Pages: 190-203, ISSN: 0743-7315, Elsevier Press, Amsterdam, The Netherlands, October 2019.

- **Redowan Mahmud**, Satish Narayana Srirama, Kotagiri Ramamohanarao, and Rajkumar Buyya, "Profit-aware Application Placement for Integrated Fog-Cloud Computing Environments", *Journal of Parallel and Distributed Computing (JPDC)*, Volume 135, Pages: 177-190, ISSN: 0743-7315, Elsevier Press, Amsterdam, The Netherlands, January 2020.

Supplementary Contributions

During the PhD candidature, I have also contributed to the following collaborative works (this thesis does not claim them as its contributions):

- **Redowan Mahmud** and Rajkumar Buyya, "Modelling and Simulation of Fog and Edge Computing Environments using iFogSim Toolkit", *Fog and Edge Computing: Principles and Paradigms*, R. Buyya and S. Srirama (eds), Pages: 433-466, ISBN: 978-111-95-2498-4, Wiley Press, New York, USA, 2019.

- **Redowan Mahmud**, Fernando Luiz Koch, and Rajkumar Buyya, "Cloud-Fog Interoperability in IoT-enabled Healthcare Solutions", *Proceedings of the 19th International Conference on Distributed Computing and Networking (ICDCN 2018)*, ACM Press, Article: 32, Pages: 1–10, Varanasi, India, January 4-7, 2018.

- Shreshth Tuli, **Redowan Mahmud**, Shikhar Tuli, and Rajkumar Buyya, "FogBus: A Blockchain-based Lightweight Framework for Edge and Fog Computing", *Journal*

of Systems and Software (JSS), Volume 154, Pages: 22-36, ISSN: 0164-1212, Elsevier Press, Amsterdam, The Netherlands, August 2019.

- Wendy Yanez, **Redowan Mahmud**, Rami Bahsoon, Yuqun Zhang, and Rajkumar Buyya, "Data Allocation Mechanism for Internet-of-Things Systems With Blockchain", *IEEE Internet of Things Journal*, Volume 7, Number 4, Pages: 3509-3522, ISSN: 2327-4662, IEEE Computer Society Press, USA, April 2020.

- Adel Nadjaran Toosi, **Redowan Mahmud**, Qinghua Chi, and Rajkumar Buyya, "Management and Orchestration of Network Slices in 5G, Fog, Edge and Clouds", *Fog and Edge Computing: Principles and Paradigms*, R. Buyya and S. Srirama (eds), Pages: 79-102, ISBN: 978-111-95-2498-4, Wiley Press, New York, USA, 2019.

Acknowledgements

All praise to Almighty ALLAH who blessed me to complete my PhD studies and I convey my humblest gratitude to the Holy Prophet Muhammad (S.A.W).

I would like to thank my supervisors, Professor Rajkumar Buyya and Professor Ramamohanarao Kotagiri, for giving me the opportunity to undertake this PhD. I am truly grateful for their invaluable support, guidance and motivation throughout my candidature. I would also like to express my gratitude to the chair of my PhD advisory committee, Dr Aaron Harwood, for his comments and suggestions.

I am thankful to Dr. Rodrigo N. Calheiros and Dr. Amir Vahid Dastjerdi for their constructive comments and technical advices in the beginning of my PhD journey.

Special thanks to Dr. Md. Abdur Razzaque for helping me to develop the research skills during my undergraduate studies. I would also like to thank all the past and current members of the CLOUDS Laboratory, at the University of Melbourne. In particular, I thank Dr. Satish Narayana Srirama, Dr. Adel Nadjaran Toosi, Dr. Md Akter Hussain, Dr. Fernando Koch, Dr. Rami Bahsoon, Dr. Chenhao Qu, Dr. Jungmin Jay Son, Dr. Safiollah Heidari, Dr. Minxian Xu, Dr. Kyong Hoon Kim, Dr. Maria Salama, Dr. Sara Kardani, Shashikant Ilager, Muhammad Hilman, Muhammed Tawfiqul Islam, TianZhang He, Mohammad Goudarzi, Zhiheng Zhong, Samodha Pallewatta, Amanda Jayanetti and Mohammad Reza Razian for their support.

I acknowledge the University of Melbourne and Australian Research Council (Discovery Project) for providing me with scholarships to pursue my doctoral studies.

At this phase of my PhD journey, I am remembering my father Matiar Rahman, who has always been the centre of my inspiration. My deepest gratitude goes to my mother Asrafi Rahman, sister Rumana Rahman and Nurjahan Lisha, brother Rabiul Hasan and Niaz Mahmud, nephew Rayan and niece Rizvana, uncle Noyon Chowdhury, and my in-laws for their continuous support, affection and encouragement.

Last but not least, I am thankful to my wife Mahbuba Afrin for her endless love, devotion, appreciations and understandings.

Md Redowan Mahmud
Melbourne, Australia
January 2020

Contents

List of Figures

List of Tables

List of Acronyms

IoT Internet of Things

CPS Cyber-Physical Systems

SLA Service Level Agreement

QoS Quality of Service

QoE Quality of Experience

LAN Local Area Network

MCC Mobile Cloud computing

C-RAN Cloud Radio Access Network

SDN Software-Defined Networking

URLLC Ultra Reliability and Low latency Communication

ETSI European Telecommunications Standards Institute

MEC Mobile Edge Computing

NFV Network Function Virtualization

FRAN Fog computing-enabled Radio Access Network

BLE Bluetooth Low Energy

DAG Directed Acyclic Graph

DoS Denial of Service

GL Gateway Level

IL Infrastructure Level

IAB IoT Application Broker

FRM Fog Resource Manager

CRM Cloud Resource Manager

FCs Fog Clusters

FGs Fog Gateways

ILP Integer Linear Program

Avg. ADH Average Amount of Data Handled

Avg. ML Average Management Load

Avg. ADRP Average Delay from Request to Placement

Per. QSA Percentage of QoS Satisfied Applications

Avg. NRT Average Network Relaxation Time

Avg. RUR Average Resource Utilization Ratio

MigCEP Mobile Complex Event Processing

MMC Mobile Micro Cloud

CoAP Constrained Application Protocol

SNMP Simple Network Management Protocol

MSB Module Sleeping Block

PL Placement List

RL Routing List

SORS Service Oriented Resource Sharing

PVA Peer VMs Aggregation

SCIP Solving Constraint Integer Programs

I4OAs Industry 4.0-Oriented Applications

FCNs Fog Computing Nodes

Avg. SDT Average Service Delivery Time

Avg. CRO Average Computing Resource Overhead

Avg. NRR Average Network Relaxation Ratio

Per. DSI Percentage of Deadline Satisfied Inputs

TIPM Time to Identify the Placement Map

MOS Mean Opinion Scores

SOS Standard deviation of Opinion Scores

NPS Net Promoter Score

MeFoRE MEdia FOg Resource Estimation

ITU International Telecommunication Union

EEG Electroencephalogram

FGNs Fog Gateway Nodes

MCIs Micro Computing Instances

RoE Rating of Expectation

CCS Capacity Class Score

RG Resource Gain

PTRR Processing Time Reduction Ratio

TIPS Thousand Instructions Per Second

SFCs Service Function Chains

PM Profit Merit

SLOs Service Level Objectives

Chapter 1

Introduction

The Internet of Things (IoT) paradigm has changed the structure of physical environments by connecting numerous computing components, digital machines and animals with the Internet. It enables them to perceive the external ambiance as sensors and trigger any action based on the given commands using actuators [1]. Thus, it creates a novel type of interaction among different real-world entities in ingenious ways. The ongoing advancement in the field of hardware and communication technology is consistently improving and expanding the applicability of IoT. It consequently helps in realizing the theory of smart city, remote healthcare, Industry 4.0 and smart agriculture [2]. Recently, various Cyber-Physical Systems (CPS) for smart environments such as indoor locators, digital health recorder and robot-assisted supply chain manager have been developed through the widespread deployment of IoT devices. Moreover, according to the current trend of practising IoT, it is expected that by 2030, there will be 1.2 trillion active IoT devices with potential economic impact of $15 trillion per year [3].

IoT devices can generate data incessantly or periodically. Different types of applications are used to process these data [4]. As most of the IoT devices are equipped with limited energy, computing and networking capabilities, they are considered ill-suited to execute heavyweight applications [5]. Moreover, based on the working environment of the CPSs, applications are often forced to process data within a defined time frame. Their data-driven interactions with IoT devices also demand a less-congested network. The computing paradigm executing the applications for IoT-enabled CPSs needs to observe these issues so that the desired responsiveness of the CPSs can be ensured.

Figure 1.1: Geographical coverage of different Cloud service providers

1.1 Fog: A New Computing Paradigm

Cloud computing has been the backbone for hosting subscription-oriented resources and application services. It is also used to execute the applications for different IoT-enabled CPSs [6]. Cloud datacentres consist of data and computing servers to facilitate users with storage and virtualized computing instances [7]. As shown in Figure 1.1, these datacentres are located at a multi-hop distance from the IoT devices [1]. Therefore, an extended period is required to transfer data and command between the IoT devices and the applications executing on the Cloud instances. It degrades the application service delivery time. When a large number of IoT devices initiate data-driven interactions with remote applications, it adds substantial load to the network and creates severe congestion. It also increases the computational overhead on Cloud datacentres [8]. Consequently, the Cloud-centric application execution model fail to meet the performance threshold for different IoT-enabled CPSs, especially in latency-sensitive use cases. To address such limitations, an extension of Cloud computing named *Fog computing* was introduced by CISCO in 2012 [9].

[1] Source: https://www.atomia.com/2016/11/24/comparing-the-geographical-coverage-of-aws-azure-and-google-cloud/

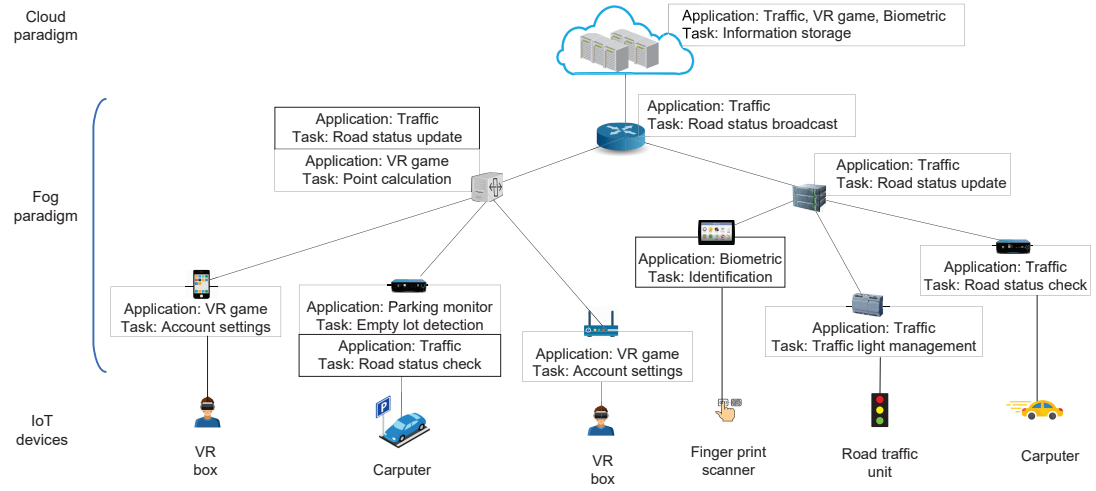

Figure 1.2: An illustration of application execution in Fog environments

1.1.1 Benefits of Fog Computing

Fog manages an intermediate layer between end-user devices and Cloud datacentres by utilizing the computing components within the edge network [10]. In Fog environments, these computing components; for example personal computers, gateways, Raspberry PIs, nano-servers and micro-datacentres, are commonly known as *Fog nodes*. As shown in Figure 1.2, Fog nodes can execute various IoT applications in the proximity of data sources. Hence, Fog computing resists end-user devices to send a huge amount of data towards Cloud datacentres that decreases the data propagation delay. As a consequence, the service latency of different applications improves [11]. Moreover, it conserves network bandwidth that reduces the scope of network congestion. Furthermore, providers can migrate the computational load from Cloud datacentres to network edge through Fog computing. As Fog nodes are less expensive, it lowers the operational cost of providers and saves the energy in Cloud datacentres. Additionally, Fog computing supports robust location-awareness to simplify the communication with mobile and energy-constrained end-user devices [12]. Because of the features mentioned above, Fog computing is considered very promising to meet the application service requirements for different IoT-enabled CPSs.

1.1.2 Initiatives for Realizing Fog Computing

Taking the benefits of Fog computing into cognizance, several technology giants such as Amazon, Alphabet and Microsoft have already started integrating Fog services with their Cloud infrastructure [13]. To standardize the theory of Fog computing, an association of academic and industrial bodies, named Industrial Internet Consortium, has been formed recently [14]. Moreover, Cisco has developed IOx-enabled networking devices and begun to market them as multi-purpose Fog nodes across the globe [15]. Following Cisco, different hardware manufacturers such as Intel and Dell have also approached to build compatible machines for Fog computing [16].

There exists other companies like SONM and NEC Laboratories which are engaged in making dedicated software systems for Fog computing environments. The development of FogFlow framework is regarded as a successful attempt to this direction [17]. Similarly, start-up initiatives like FogHorn Systems and Drofika Labs, have launched different Fog computing-based IoT solutions to simplify the day to day life of users [18]. The Fog computing-based real-time remote patient monitoring and critical care management offered by Tata Consultancy Services is one of the recent inclusions in this domain [19]. With such a pace of advancement, Fog computing is expected to add $5.7 billion more to the global market of utility computing by 2025 [20].

1.1.3 Challenges of Fog Computing

The challenges of application management in Fog computing environments are discussed below.

• *Resource and energy-constrained, distributed and heterogeneous Fog nodes*: Most of the Fog nodes are constrained in processing power, networking capability, storage and energy capacity. Their resource architecture, communication standards and operating systems also vary from one to another. Additionally, they are deployed in distributed order at the edge network. As a consequence, time-optimized and platform-independent application execution becomes tedious to ensure in Fog.

• *Absence of business model*: Fog environments is a relatively new computing paradigm that lacks a unanimous business model. This limitation of Fog computing significantly

affects the cost and budget management of providers and users respectively. Additionally, Fog infrastructure is less flexible than Cloud in terms of sharing resources. For example, the Fog resources located in California cannot be harnessed for the CPSs in Melbourne due to the geographical distribution. Due to localized supply and uneven demand, subscription-oriented business models are difficult to develop for Fog computing environments.

• *Subjected to uncertain failures*: Although Fog computing reduces application service delivery time, Fog nodes are highly prone to get affected by anomalies, power failures and out of capacity faults. It obstructs the execution of applications assigned to them. Due to the latency constraints, it also becomes difficult to recover the system from fault without degrading the system performance.

• *Standard-less integration*: In some cases, the applications executing in Fog environments need the services offered by different computing paradigms. For example, the Fog-based health data analytic applications require the Cloud-based storage service to facilitate location-independent medical report sharing. To manage applications during such scenarios, integration of Fog infrastructure with other services is necessary. Nevertheless, the absence of efficient frameworks and standards often resist the Fog environments to provide this assistance to the applications.

• *Lack of interoperability*: Due to various benefits of Fog computing, it is expected to execute most of the IoT applications in Fog environments. Regrettably, it requires an extensive programming effort to customize the existing Cloud-based IoT applications so that they can execute in Fog environments. It happens due to lack of application interoperability. Moreover, it is hard to solve because of the differences in resource architecture, management operations, service orchestration and security measures of Fog and Cloud computing environments.

• *Inefficient task distribution*: Fog computing environment is decentralized in nature. Therefore, it is tough to orchestrate the services offered by Fog environments and synchronize the Fog nodes while assigning applications to them for execution. Furthermore, the coexistence of multiple decision-making entities increase the application management complexity in Fog environments that ultimately results in poor distribution of application tasks over the Fog nodes.

• *Less secured*: The outcomes of Fog-based applications can be used by different parties simultaneously. For example, the services of a Fog-based healthcare application are relevant to hospitals, insurance companies and employer organization. In such cases, Fog environments require to ensure on-demand and secure access to application outcomes as a part of application management. However, due to the resource scarcity and dynamics of Fog environments, it is hard to apply compute-intensive and complex security measures on Fog nodes.

In this thesis, we address the challenges of executing applications through resource-constrained, heterogeneous and distributed Fog nodes by identifying their suitable placement options in Fog environments. Here, we propose a taxonomy on application management in Fog computing and review the existing application management strategies and their limitations. Furthermore, we develop a set of application placement policies to enable the efficient management of applications in Fog environments.

1.2 Research Questions and Objectives

In smart systems, numerous IoT devices, applications and Fog nodes continuously interact with each other. These interactions are mostly driven by the service exceptions of the users and the monetary aspects of the infrastructure providers. This thesis investigates the placement of applications in Fog environments from the perspectives of different entities interacting with the IoT-enabled smart systems. The objective of this thesis is to enhance the application's Quality of Service (QoS), user's Quality of Experience (QoE) and provider's profit by harnessing the Fog nodes. We formulate the application placement problem and attain the objective by exploiting the following research questions:

- Q1. *What are the QoS dominating factors for the applications in Fog environments?* : To answer this question, we first analyse the characteristics of the applications such as their latency-sensitivity, data load and frequency of external interactions. The intensity of these characteristics depends on the service delivery deadline, the size of the input and the sensing frequency of IoT devices and they play vital roles in defining the QoS requirements of the applications. Moreover, if the application is distributed across multiple Fog nodes, the inter-nodal communication delay

among them becomes crucial to meet the application QoS. Similarly, for monolithic applications, per unit time computing and networking resource occupancy help in determining the overhead of Fog nodes which consequently denotes their suitability for meeting the QoS of the applications.

- Q2. *Why the enhancement of user's QoE through Fog computing is essential?* : The user expectations in terms of application responsiveness, resource requirements and service access can vary from one to another. If Fog computing is unable to deal with them, the relinquish rate of users will increase. Additionally, the acceptability and QoE for the application will degrade even when the Fog computing meets its QoS. For example, an application's QoS can guarantee downloading of a file in maximum 5 minutes. On a particular scenario, two users may require that file within 3 and 7 minutes respectively. If the application downloads the file in exact 5 minutes, the expectation of the second user will be met; however, it will be failed for the first user. As a consequence, despite of meeting the QoS, QoE of both users will not be the same for that application. Hence, it is required to manage the applications in Fog environments according to user expectations.

- Q3. *How should the pricing of Fog resources be modelled to increase the provider's profit?* : From the perspective of infrastructure providers, the successful realization of Fog computing largely depends on its monetary aspects. In most of the cases, the explicit intention of maximizing revenue leads the providers to overlook the stringent QoS requirements of the applications and the budget constraints of the users. As a result, the Service Level Agreement (SLA) violates. In such cases, the commitment of providing compensation to users for SLA violation can backfire and affect the profit accumulation of providers. The uneven expenses of operating heterogeneous Fog nodes can make it even more complicated for the providers. An efficient pricing model for the Fog resources can solve this issue to a great extent. It should be set according to the scale of performance improvement for executing the applications in Fog environments. Hence, it will arouse the necessity of satisfying the SLA that will consequently increase the provider's profit.

1.3 Thesis Contributions

Based on the research questions mentioned above, the contributions of this thesis are:

1. Proposes a taxonomy on application management in Fog computing and reviews the existing application architecture, placement and maintenance approaches.
2. Investigates efficient application management policies that ensure QoS satisfied service delivery for different types of applications in Fog environments (addresses the Q1).
    - A framework that distributes the application management tasks across the gateway and the infrastructure level of Fog environments.
    - A procedure to select applications for Fog-based placement according to their QoS requirements.
    - A latency-aware approach for placing distributed applications over Fog nodes.
    - A module forwarding strategy that re-locates applications to optimize the number of Fog nodes without degrading their QoS.
    - A context-aware approach that optimizes application service delivery time.
    - A strategy to manage the network congestion and computation overhead of Fog nodes while executing the applications.
3. Exploits an expeditious technique that enhances the QoE of users by managing the applications according to their expectations (addresses the Q2).
    - An innovative approach to prioritize service expectations of different users in Fog environments.
    - A policy to classify the Fog nodes based on their capabilities of satisfying user demand.
    - An optimization model that ensures maximum QoE-gain of the users through resource-constrained Fog nodes.
4. Develops a novel scheme that increases the profit of providers in integrated Fog-Cloud computing environments (addresses the Q3)

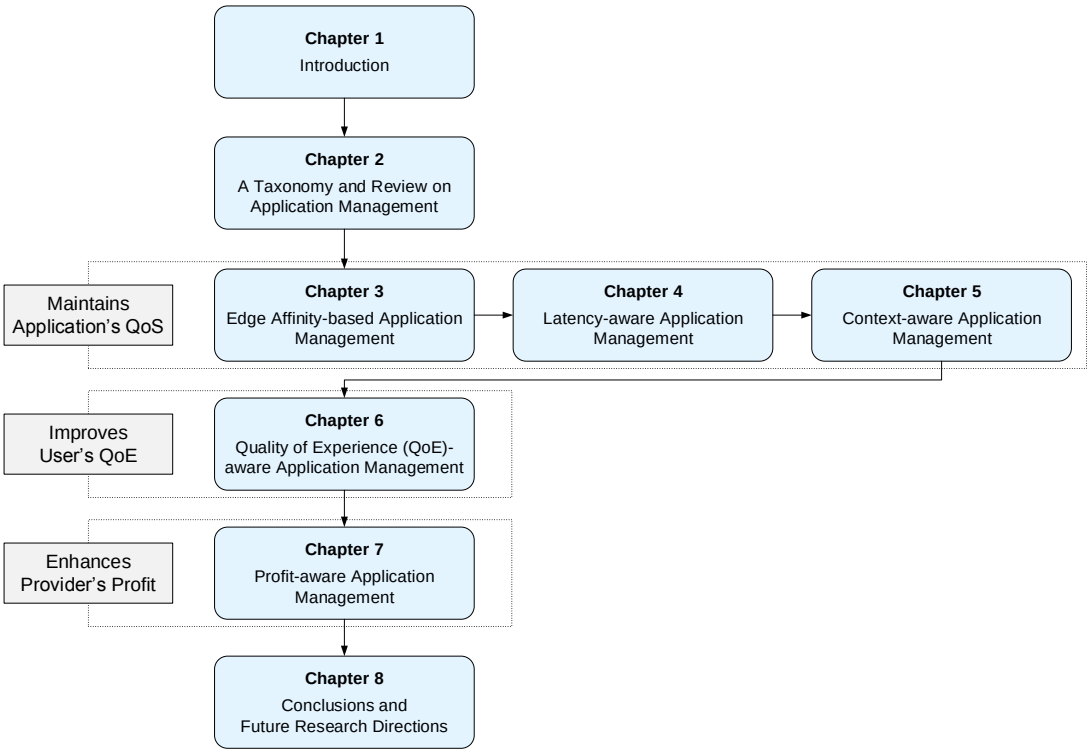

Figure 1.3: The thesis structure

- A placement policy that enhances the provider's profit and meets application's QoS simultaneously.
- A pricing model for Fog resources that increases revenue according to the level of performance improvement.
- A compensation method that supports both the user's and the provider's economic interests through an inverse relationship between compensation amount and QoS satisfaction rate.

1.4 Thesis Organization

The structure of this thesis is shown in Figure 1.3. The remaining part of this thesis is organized as follows:

- Chapter 2 presents a taxonomy and literature review on application management in Fog computing environments. This chapter is derived from:

  - **Redowan Mahmud**, Kotagiri Ramamohanarao, and Rajkumar Buyya, "Fog Computing: A Taxonomy, Survey and Future Directions", *Internet of Everything: Algorithms, Methodologies, Technologies and Perspectives*, B. DiMartino, K. Li, L. Yang, A. Esposito (eds), Pages: 103-130, ISBN 978-981-10-5860-8, Springer, Singapore, 2018.

  - **Redowan Mahmud**, Kotagiri Ramamohanarao, and Rajkumar Buyya, "Application Management in Fog Computing Environments: A Taxonomy, Review and Future Directions", ACM Computing Surveys (in second revision).

- Chapter 3 presents an application characteristics-driven model that facilitates application classification and selection at the gateway level of IoT-enabled smart systems for Fog-based placement. This chapter is derived from:

  - **Redowan Mahmud**, Kotagiri Ramamohanarao, and Rajkumar Buyya, "Edge Affinity-based Management of Applications in Fog Computing Environments", *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing (UCC'19)*, IEEE Press, Pages: 61-70, Auckland, New Zealand, December 2-5, 2019.

- Chapter 4 presents a latency-aware application management policy that deals with the service delivery deadline and inter-nodal communication delay for placing the applications over distributed Fog nodes. This chapter is derived from:

  - **Redowan Mahmud**, Kotagiri Ramamohanarao, and Rajkumar Buyya, "Latency-aware Application Module Management for Fog Computing Environments", *ACM Transactions on Internet Technology (TOIT)*, Volume 19, No. 1, Article 9, Pages: 1-21, ISSN:1533-5399, ACM Press, New York, USA, January 2019.

- Chapter 5 presents a context-aware application management policy that optimizes the service time of applications based on the sensing frequency and the data size of the IoT devices. This chapter is derived from:

  - **Redowan Mahmud**, Adel Nadjaran Toosi, Kotagiri Ramamohanarao, and Rajkumar Buyya, "Context-aware Application Placement for Industry 4.0 in Fog Com-

puting Environments", *IEEE Transactions on Industrial Informatics* (in press, DOI: 10.1109/TII.2019.2952412, accepted on November 5, 2019).

- Chapter 6 proposes a QoE-aware application management policy that prioritizes the placement of applications in Fog environments based on the user expectations and the capacity of the Fog nodes. This chapter is derived from:

  - **Redowan Mahmud**, Satish Narayana Srirama, Ramamohanarao Kotagiri, and Rajkumar Buyya, "Quality of Experience (QoE)-aware Placement of Applications in Fog Computing Environments", *Journal of Parallel and Distributed Computing (JPDC)*, Volume 132, Pages: 190-203, ISSN: 0743-7315, Elsevier Press, Amsterdam, The Netherlands, October 2019.

- Chapter 7 proposes a pricing model for integrated Fog-Cloud environments that enhances the profit of service providers for placing and executing the applications in the proximity of users. This chapter is derived from:

  - **Redowan Mahmud**, Satish Narayana Srirama, Kotagiri Ramamohanarao, and Rajkumar Buyya, "Profit-aware Application Placement for Integrated Fog-Cloud Computing Environments", *Journal of Parallel and Distributed Computing (JPDC)*, Volume 135, Pages: 177-190, ISSN: 0743-7315, Elsevier Press, Amsterdam, The Netherlands, January 2020.

- Chapter 8 concludes the thesis by summarizing the findings and offers directions for future research.

Chapter 2

A Taxonomy and Review on Application Management

This chapter investigate the existing application management approaches in Fog computing and review them in terms of architecture, placement and maintenance. Based on in-depth analysis of the literature, a taxonomy on application management in Fog computing environments is proposed. The detailed survey of existing approaches is conducted according to the taxonomy. A perspective model for managing applications in Fog environments is also presented. Finally, the research gaps for further improvement of Fog computing concept is highlighted.

2.1 Introduction

Fog computing creates a wide distribution of infrastructure and platform services . Infrastructure services include on-demand exploitation of computing, networking (bandwidth and firewalls) and storage resources, whereas platform services facilitate application runtime environments, operating systems and programming interfaces [21]. Fog resource management denotes the administrative operations such as deployment, virtualization and monitoring of Fog nodes that foster the Fog-based infrastructure and platform services [22]. Fog resource management also functions load balancing, dynamic

This chapter is derived from:

- **Redowan Mahmud**, Kotagiri Ramamohanarao, and Rajkumar Buyya, "Fog Computing: A Taxonomy, Survey and Future Directions", *Internet of Everything: Algorithms, Methodologies, Technologies and Perspectives*, B. DiMartino, K. Li, L. Yang, A. Esposito (eds), Pages: 103-130, ISBN 978-981-10-5860-8, Springer, Singapore, 2018.
- **Redowan Mahmud**, Kotagiri Ramamohanarao, and Rajkumar Buyya, "Application Management in Fog Computing Environments: A Taxonomy, Review and Future Directions", ACM Computing Surveys (in second revision).

provisioning and auto-scaling to ensure service availability and multi-tenancy [23].

Efficient Fog resource management assists IoT-enabled CPSs to operate multiple applications simultaneously. However, the characteristics of these applications vary from one CPS to another. For example, the expected application service delivery time for a CPS that remotely monitors the respiratory functions of critical patients is quite stringent compared to a CPS, which measures the environmental parameters [24]. Moreover, an application that assists a CPS to perform virtual reality operations handles huge amount of data in per unit time compared to an application which helps in tracking the empty parking slots. Such diversified characteristics play vital roles in defining the Quality of Service (QoS) requirements of the applications that cannot be met only through Fog resource management. This perception also urges to develop different application management strategies according to the preferences of the applications. Usually, an application management strategy refers to a collection of algorithms, mathematical models, empirical analysis and recommendations that regulate the implementation, installation and execution of applications in a computing environment. Moreover, application management strategies practice admission control, location transparency, data maintenance and service resiliency as per the demands of the corresponding system [25]. Nevertheless, there are three research questions that become crucial while developing application management strategies for Fog computing environments. They are listed as:

- *How should the applications be composed?*

  To address this question, an application management strategy requires to specify the features of applications such as their programming model, functional layout, service type, workload type so that they can be aligned with the Fog-based infrastructure and platform services.

- *How should the applications be placed?*

  To address this question, an application management strategy requires to find suitable placement options for the applications in Fog environments. At the same time, the strategy needs to make a balance between application-centric QoS requirements.

- *How should the applications be maintained?*

To address this question, an application management strategy requires to facilitate security and resiliency support during application execution in Fog environments. Moreover, the strategy needs to monitor the performance of the applications in consistent manner.

Investigating the operational responses to these questions, a notable number of application management strategies has already been developed for Fog computing environments. They predominantly focus on the modularization of applications to deal with the resource constraints of Fog nodes [26] [27] [28]. These strategies also adopt the web service-based communication techniques to simplify the interactions between different components of modular applications hosted on distributed Fog nodes [29] [30]. While assigning the applications to the Fog nodes, the existing application management strategies give much emphasis on meeting the service delivery deadline and optimizing the cost and energy consumptions [31] [32] [33]. Most of them operate discretely and apply strict synchronization measures over the Fog nodes to mitigate the effect of interference [34]. The application management strategies also incorporate both proactive and reactive fault tolerance techniques to support the reliable execution of the applications in Fog environments [35] [36] [37]. However, in the literature, very few initiatives have been found that categorize the application management strategies in a systematic way [10] [38]. Therefore, in this chapter, we identify three important aspects of application management in Fog computing environments namely application architecture, application placement and application maintenance, as shown in Fig. 2.1 and present separate taxonomy for each of them. Based on the proposed taxonomy, we also review the existing application management strategies and denote the associated research gaps. We also discuss a framework that is logically distributed and helps adaptive and holistic management of applications in Fog computing environments.

The rest of the chapter is organized as follows: The the differences between Fog computing and other contemporary computing paradigms along with the description of related surveys are illustrated in Section 2.2. In Section 2.3, a discussion on application architecture in Fog environments is presented. Section 2.4 highlights the existing techniques to place applications in Fog computing. Section 2.5 explores application maintenance operations. Section 2.6 demonstrates a perspective framework for Fog-based

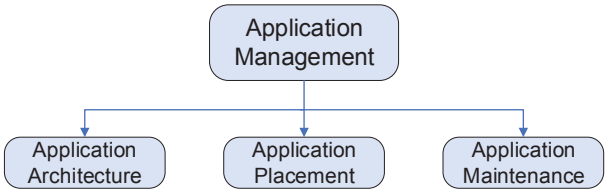

Figure 2.1: Aspects of application management

application management. Finally, Section 2.7 concludes the chapter.

2.2 Background Study

Fog computing is a very recent inclusion in the domain of computing paradigms. In this section, the evolution of Fog computing and its differences with the contemporary computing paradigms are discussed. Additionally, the existing literature survey and the various aspects of application management in Fog computing are highlighted.

2.2.1 Evolution of Fog Computing

The perception regarding distributed computation has been continuously updating since its origin. As a consequence, different computing paradigms based on distributed computation have emerged from time to time as shown in Figure 2.2. In 1967, Cluster computing was first introduced where a set of computers, tightly coupled with each other through Local Area Network (LAN), work together like a single system to perform the same tasks [39]. In most of the cases, these cluster computers are homogeneous, and they are controlled and managed by a software running on a specific entity within the cluster. The further expansion of Cluster computing is made in form of Grid computing during early 1990s [40]. It connects computers and clusters from different administrative domains to process non-interactive workloads. The computing components in Grid environments are loosely coupled, geographically dispersed and heterogeneous. They are managed in decentralized manner and set to perform different tasks. At the beginning of 2000s, Grid computing is improvised to Cloud computing that offers not only infrastructure services but also platform and software services as utility over the Internet

[7]. In Cloud environments, remote datacentres host virtualized computing resources, and a centralized system manages their operations and ensures their on-demand access to users. Unlike Grid and Cluster computing, Cloud computing is widely adopted in numerous domains including industry, healthcare, education and research due to its service oriented architecture, high resource availability, scalability, guaranteed services and location independence.

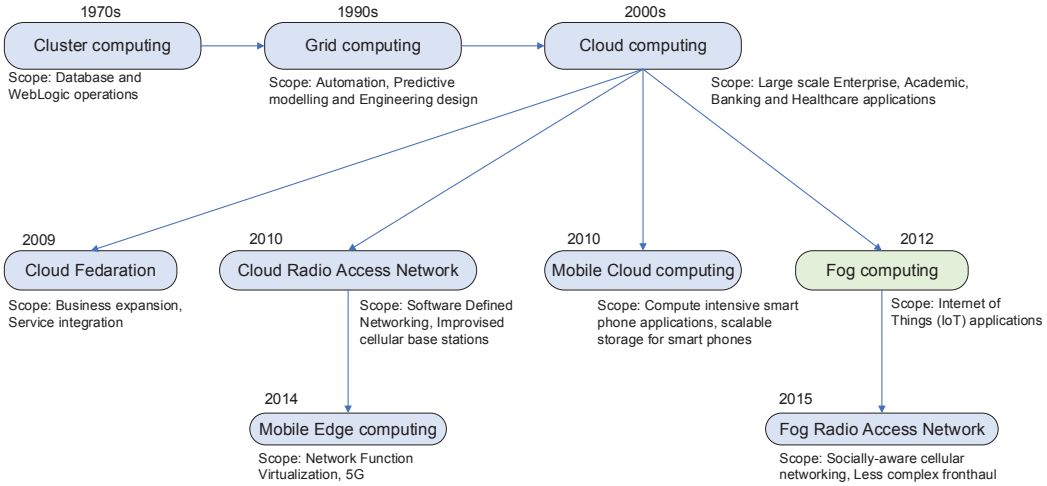

Figure 2.2: Evolution of Fog computing

Recently, different extensions of Cloud computing have been developed. In Federated Cloud computing, Cloud services from multiple providers are inter-connected to balance the computational load among them and accommodate the spikes of user's demand [41]. It supports providers to expand their geographic footprints and business, and assists users to access different Cloud environments using single credential. Moreover, during 2009-2012, the number of smart phone users, their application and cellular service requirements, and the practice of IoT technology began to proliferate significantly. To handle this situation, Cloud services are augmented with traditional cellular and sensor network architecture, and the concept of Mobile Cloud computing (MCC), Cloud Radio Access Network (C-RAN) and Fog computing are proposed [42] [43].

Through MCC, users are facilitated to offload compute and data intensive smart phone-based applications to Cloud for execution. Thus, it overcomes the limitations of smart phones in terms of energy, storage and computation, and improves user experi-

ence. Sometimes, MCC is designed with an additional computing layer constructed with Cloudlets in between smart phones and Cloud datacentres. Hence, it offers a three-tier computing environment and reduces application offloading delay to meet their latency constraints [44]. Conversely, C-RAN moves the radio signal processing from cellular base stations to Cloud datacentres and helps to reduce cost of the providers. It improves resource utilization and simplifies the coordination among cellular base stations. C-RAN is also considered as the foundation for Software-Defined Networking (SDN) [45]. SDN separates control (IP routing table, Routing protocol) planes from distributed networking devices and places them to an centralized entity so that holistic administration can be enabled in the networking architecture.

Although C-RAN promotes the realization of 2G,3G and 4G, it has limitations in assisting Ultra Reliability and Low latency Communication (URLLC) service for the fifth generation cellular network standard, commonly known as 5G. To complement C-RAN in dealing with 5G related issues, European Telecommunications Standards Institute (ETSI) brought the idea of Mobile Edge Computing (MEC) [46]. In MEC, a virtualized server is set at the cellular base station to ensure flexible and rapid deployment of new cellular services for the users. It offers real-time access to radio network information to endorse Tactile Internet, interactive gaming and virtual reality applications through 5G. Moreover, MEC plays an important role in Network Function Virtualization (NFV) [47]. NFV technique shifts the operations of forwarding and management plane such as packet transfer, security maintenance, address translation and mobility management from purpose build hardware to virtualized instances that consequently improves network response time.

Among the available extensions of Cloud computing, Fog is more focused on serving IoT applications [9]. By extending the computation facilities closer to users, it ensures reduced application service delivery time for different IoT-enabled CPSs with less network congestion. Similar to MCC, Fog computing creates multi-tier application execution platform between IoT devices and Cloud datacentres through deployment of Fog nodes in different networking level and assists data processing within the communication channel. It is highly scalable and convenient for device to device interactions. Moreover, the Fog computing-enabled Radio Access Network (FRAN) enhances

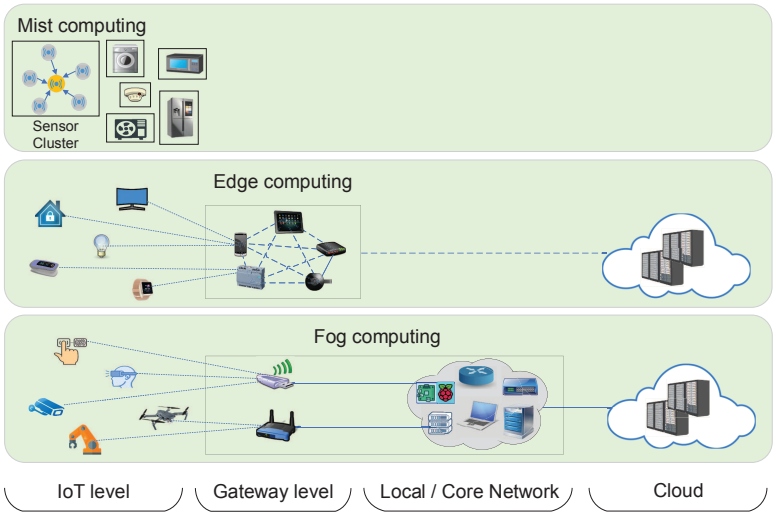

Figure 2.3: Domain of Mist, Edge and Fog computing

the responsiveness of socially-aware networking and simplifies the complexity in cellular communication [48]. Additionally, to harness the capabilities of Fog computing, various specialized networking components namely Cisco IOx-enabled devices have already been launched. They can support NFV and promote the realization of MEC. Hence, it is widely accepted that Fog computing not only complements IoT-enabled CPSs but also endorses the basic operations of different Cloud-extensions. Several association of academic and industrial bodies such as Industrial Internet Consortium and FogHorn Systems are currently working to standardize such concepts of Fog computing across the world. Apart from them, some initiatives are also taken to engage other computing paradigms for solving various IoT-related issues. For example, MEC has recently been redefined as Multi-access Edge Computing so that it can assist IoT-enabled CPSs as well. This sort of activities are going to make the research and marketplace of Fog computing quite challenging in the upcoming years.

2.2.2 Comparison among Mist, Edge and Fog computing

Likewise Fog computing, Edge and Mist computing support application execution in proximity of data sources as shown in Fig. 2.3. More precisely, Mist computing enables the IoT devices to process data within themselves whereas Edge computing performs

the processing operations at the gateways of IoT devices [49] [50]. For instance, smart watches can be considered as IoT devices. Users connect these smart watches to their smart phones via Bluetooth Low Energy (BLE) networking so that they can receive mobile notifications while walking or driving. Here, the smart phones act as the IoT gateways for the smart watches. At the same time, the smart watches sense blood pressure, heart beat and oxygen saturation rate of the users. If a watch executes the application to process the generated data, then it is regarded as Mist computing [51]. Conversely, when the watch forwards the data to a smart phone-based application for processing, then it becomes Edge computing [52]. However, compared to them, Fog computing not only harnesses the IoT gateways but also engage other computing components within the edge network such as smart routers, personal computers, Raspberry Pi devices and even micro-datacentres to process the IoT data [8].

Although Mist and Edge computing can solve many IoT-related issues, they have certain limitations. The computing components in Mist are not abundant in processing, networking and energy capacity. They are less capable of executing large-scale and complex applications for a longer period of time [53]. On the other hand, the management of Edge nodes are very much user-centric with only reactive fault-tolerant facilities. In Edge environments, it is also tedious to ensure the fairness among multiple users [54]. Fog computing overcomes these limitations of Mist and Edge by leveraging comparatively powerful resources at the user premises level and lessening the burdens of resource and application service management from the users. Moreover, Fog computing maintains a seamless communication with Cloud datacentres that eventually offers an extensive execution platform for the IoT applications [55]. The notable differences between Mist, Edge and Fog computing are listed in Table 2.1.

However, Mist, Edge and Fog are relatively new computing paradigms and their evolution processes are still ongoing. Therefore, many researchers and industries adopt different approaches to define them. For instance, there are several research works in the literature that consider Edge computing as a subset of Fog computing [24]. Oppositely, in other works, Edge computing is regarded as a superset embracing all paradigms where the computation is moved to the edge network, including Fog computing, Mobile Cloud computing and Mobile Edge computing [49]. There are also some examples

Facts	Mist	Edge	Fog
Place of operation	IoT devices, sensors and actuators	Gateway devices and set-top boxes	Specialized networking and general-purpose computing machines
Elementary Hardware	Microcontroller	Programmable logic controller	Single-board computer
Wireless standards	Zigbee, Bluetooth LE and Zwave	Bluetooth and WiFi	WiFi and LTE
Policy manager	Manufacturer	Users	Service providers
Application deployment	Programmed	Installed by user	Requested by user to service providers
Resource assignment	Dedicated	Shared	Shared or virtualized
Application-user mapping	Single application, single user	Multiple application, single user	Multiple application, multiple user
Resource orientation	Standalone, homogeneous cluster	Peer to Peer, Ad-hoc	Cloud of Things
Cloud communication	Incoherent or through mediator	Event-driven	Seamless
Fault tolerance techniques	Replacement	User-defined exception handling	Proactive and reactive
Extended from	Wireless sensor network, embedded systems	Personalized computing environments	Cloud computing

Table 2.1: Summary of Mist, Edge and Fog computing

where Fog and Edge computing are used interchangeably [56]. Moreover, in certain cases, Edge computation is regarded as a service model which is offered by different paradigms namely Dew, Mist and Fog computing. According to this concept, Dew computing happens in the IoT devices and Mist computing occurs at the IoT gateways [57]. Nevertheless, among these contemporary paradigms, Fog computing is considered highly feasible due to its widespread support for the IoT applications.

2.2.3 Related Surveys

In the context of Fog computing, the resource and the application service management are equally important. In fact, it is difficult to leverage the capabilities of Fog resources without efficient application service management and vice versa. Nevertheless, in the existing Fog-based literature surveys, application service management is considerd as a

part of Fog resource management. Among these surveys, [58], [59], [55] and [10] provided the general discussion on Fog computing. They reviewed the researches on Fog computing from the architectural perspective and highlight the key technologies and limitations of Fog computing. Moreover, they illustrated the benefits of Fog computing over Cloud computing and clearly distinguished its concept from other related computing paradigms such as Mobile Cloud Computing and Mobile Edge Computing. Other Fog-based literature surveys including [23], [60] [38] and [61] explored the basic resource management approaches in Fog environments. They investigated various management frameworks, scheduling techniques and provisioning algorithms for Fog resources. Furthermore, they reviewed the resource orchestration techniques in layered Fog environments and studied the resource management policies in accordance with the application service requirements. There exist some other literature surveys that focused on a specific aspect of Fog resource management. For example, Osanaiye et al. addressed the virtual computing instances migration methods in Fog computing [62] and Baccarelli et al. inspected energy-efficient Fog resource management [63].

Moreover, Bellavista et al. [64], Nath et al. [65] and Puliafito et al. [12] conducted surveys to conceptualize the application service management in Fog environments. They discussed the communication, security, data and actuation management as part of application service management. Besides, they highlighted different application specific Fog architecture and gave an overview to realize them for various IoT-enabled CPSs. Nevertheless, there are some other literature surveys that target particular Fog computing-based applications and their service management issues. For example, Aazam et al. [66] studied computation offloading techniques in Fog computing environments. Similary, Kraemer et al. [67], Mukherjee et al. [68] and Perera et al. [69] investigated the existing approaches that enable Fog computing in smart health care, advanced networking and smart city-based applications respectively. On the other hand, Roman et al. [70], Shirazi et al. [71] and Zhang et al. [72] discussed the security aspects from both resource and application management perspectives for Fog computing.

In Table 2.2, a summary of existing Fog literature surveys and their comparative study with respect to our work is presented. As noted, the existing surveys do not explore application service management in Fog environments comprehensively. More

specifically, they barely discuss about application architecture, placement and maintenance in collective manner and illustrate the literature taxonomy accordingly. In this work, we address these shortcomings. We also identify the associate research gaps, demonstrate a perspective framework for distributed application management and provide future direction for the improvement of Fog computing concept. The following sections of this chapter present the detail review of existing application management strategies in Fog computing.

2.3 Application Architecture

The complexities of executing IoT applications in distributed, heterogeneous and resource constrained Fog nodes can be addressed if the architecture of applications is defined as per the specifications of corresponding Fog environment. An elastic architecture also helps interoperability between different versions of an applications. Moreover, the elements of application architecture such as programming model and workload type are used to determine the placement strategy and resource consumptions of the applications. The service type of an application denotes the scope of its external exposure that assists in application maintenance. Fig. 2.4 provides a taxonomy on application architecture highlighting the main elements. These elements are described below.

2.3.1 Functional Layout

An application performs different types of operations. For example, an image processing application reduces noises from an image, converts colors, extracts features and compares the results with predefined thresholds. The functional layout of an application defines the arrangement of these operations. It assists in realizing the possible distribution of applications in constrained Fog environments. The functional layout of applications can be classified into two types; *Monolithic* and *Distributed*.

Surveys	Issues					
	Discusses application architecture	Investigates application placement techniques	Explores application maintenance operations	Provides taxonomy on application management	Conceptualizes application management framework	Relates application and resource management
[58]	∂	∂				✓
[55]	✓	✓	∂	∂		
[59]		∂			✓	✓
[10]		✓	✓	∂		
[23]	∂	✓		∂		✓
[60]		∂	∂	∂		
[61]		✓		∂		✓
[38]		∂	∂	✓	✓	✓
[64]		✓		∂	✓	✓
[65]		∂	✓	∂	✓	
[12]	∂	✓				✓
[62]			∂		✓	✓
[63]	✓			∂	✓	✓
[66]	∂	✓	∂		✓	
[67]	∂	✓				
[68]	∂	✓	✓		✓	
[69]	✓	✓			✓	
[70]	∂		✓			✓
[71]			✓		✓	✓
[72]	∂		✓		✓	✓
This survey	✓	✓	✓	✓	✓	✓

✓ denotes broad discussion on the respective issue.
∂ denotes partial discussion on the respective issue.

Table 2.2: Summary of literature surveys in Fog computing

Monolithic:

In monolithic applications, all computational operations are encapsulated in a single program. These applications function independently to each other. Within such applications, developer specific parallelism tchniques are applied so that they can run over multiple processing cores of the host Fog node. The monolithic architecture of Fog-based

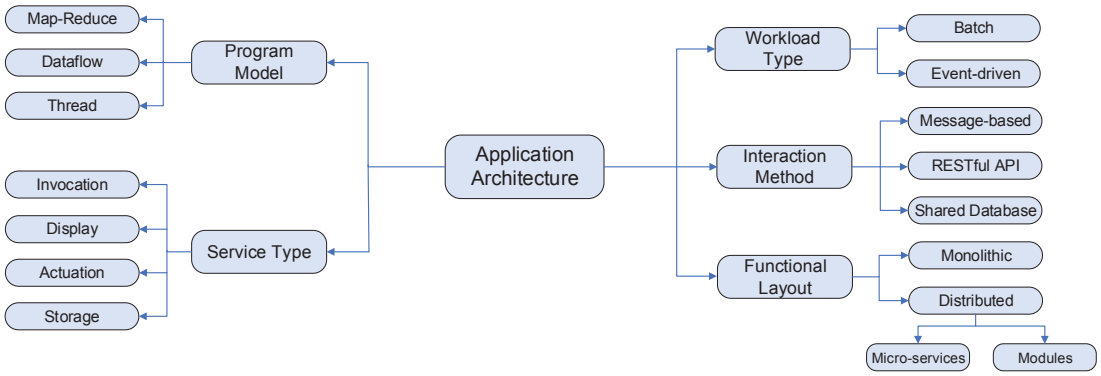

Figure 2.4: Taxonomy on application architecture

applications is discussed in [73], [35], [74], [75], [76], [56], [77], [78], [79] and [80].

Distributed:

In distributed applications, each computational operation is organized as a separate program. They can be executed in a single Fog node or can be operated by several Fog nodes in collaborative manner. Compared to monolithic applications, distributed applications are easier to expand. Based on the interactions of computational operations, distributed applications are classified into two categories.

• *Module-based:* In module-based distributed applications, the application programs are tightly coupled and dependent to each other. They are devotedly provisoned for serving data of a particular source. The module-based functional layout for applications is highlighted in [81], [82], [83], [84], [26], [85], [27], [86], [87] and [28].

• *Micro-services:* Through micro-service-based implementation, the computational operations of an application are shared among different entities to process their data simultaneously. Unlike application modules, micro-services are loosely coupled and function independently. Furthermore, due to explicit isolation, a micro-service of an application can be easily attached to other applications as per the requirements. The micro-service-based implementation of IoT applications is discussed in [88], [89], [90], [91], [92], [93], [94], [30], [95] and [96].

2.3.2 Program Model

It defines the execution order of computing operations in an application. It guides to provision resources for application as per their dimension and predicted life cycle. Three different types of application program models have been widely adopted in Fog.

Thread

To ensure simultaneous execution of independent computational operations within an application, thread program model is used. It is one of the primitive program models that helps to achieve concurrency in resource constrained Fog nodes. The thread model is adopted in [97], [56], [98], [99], [100] and [101] to compose IoT applications. The advanced versions of thread model such as map-reduce and dataflow are also used predominantly in Fog computing.

Map-Reduce

Through this model, the large-volume inputs for an application is divided into multiple chunks so that its all operations can run in parallel over the given inputs. Later, the processing outcomes of each chunk are combined to generate the overall output of the application. Such program model for Fog-based applications is discussed in [102], [103], [104], [105], [106], [107] and [108].

Dataflow

In dataflow program model, the output of a computational operation is fed to another operation as input and this process continues for the subsequent operations. Dataflow program model binds all computational operations of an application through a Directed Acyclic Graph (DAG). The size of input data handled through this model is not usually as large as that of the map-reduce model. Dataflow program model for the Fog-based applications is considered in [109], [110], [111], [112], [113], [85], [114], [94], [30] and [95].

2.3.3 Service Type

The service type of an application refers to its outcome that is delivered to the corresponding and requesting entities. The size of output of an application depends on its service types which helps to model the data propagation delay. Based on the working environment, application service outcomes can vary. The services of different IoT applications can be classified into four types.

Invocation

An IoT application can invoke the execution of another application as its service. For example, the IoT application monitoring forest fire can initiate a robotic application to meet the emergency requirements. Usually, in this type of services, an initiation command along with necessary arguments are forwarded from the source application to the requested application. Such application service type is discussed in [82], [75] and [90].

Display

There exist several IoT applications such as virtual reality gaming and smart surveillance that visualize the service outcomes to the users. The quality of such application services explicitly depends on the networking condition between the users and the associated Fog nodes. This type of application service is discussed in [74], [26], [115], [27], [116], [117], [118] and [94].

Actuation

After processing incoming data, several IoT applications trigger actuators to initiate the required physical action. For example, the remote patient monitoring system can actuate the Oxygen supply engine during emergency situations. Actuation is considered in [73], [81], [87], [119], [90], [109], [110] and [103] as a service type for the Fog-based applications.

Storage

For long-term data collection or crowd sourcing, IoT applications are often used. These applications aggregate these data and store in Cloud or Fog nodes for future analysis by other applications. Storage is mentioned in [81], [86] and [79] as an application service type for Fog.

2.3.4 Interaction Method

While processing data in a collaborative manner, the computing operations within an application require to interact with each other for sharing and storing the intermediate outcomes for further usages. It is also applicable for different applications pursuing a common goal. However, this interaction becomes very crucial when it operates across multiple Fog nodes. In the following subsections, different interaction methods for Fog-based applications are discussed.

Shared Database

It is one of the primitive methods of sharing data. In this method, data is stored in a particular location and all the applications and computing operations requiring the data have direct access to it. This method also supports multi-level distribution of data from local and global perspectives for large-scale systems. The shared database-driven interaction is discussed in [74], [79], [102] and [113].

Message-based

In this interaction method, the host Fog nodes of computing operations or applications exchange lightweight messages to notify the current sate of data processing. In most of the cases, this message transmission is supervised by a dedicated entity and follows the publish and subscribe protocol for machine to machine communication. Unlike shared database-driven interaction, this method is often used for small-scale systems. Message-based interaction is considered in [31], [120], [91] and [111].

Representational State Transfer

It offers a web service-based communication between the host Fog nodes on the top of http protocol. It allows data exchange through several stateless commands such as get and post, and often follows the push and pull approach during device level interaction. The representational state transfer is used in [29], [92], [94] and [24]. Due to the speed and ease of scalability, this method is being widely used by the IoT applications compared to shared database and message-based interactions.

2.3.5 Workload Type

Workload denotes the characteristics of input processed by an application. The knowledge on workload type is very important to set the appropriate configuration of host Fog node in terms of network bandwidth, processing cores and memory. The workload type for IoT applications are broadly classified into two categories as listed below.

Event-driven

It refers to the non-interactive inputs of an application. Once accumulated from multiple sources, the event-driven workload is submitted to the application for batch processing. The dispatch order of the inputs in such workload can be shuffled as per the availability of resources to ensure the desired performance of the application. The event-driven workload for Fog-based applications is considered in [112], [103], [121], [122], [101], [114], [11], [118], [123] and [96].

Stream

This type of workload is generated by different sources in periodic manner. Therefore, while developing real-time IoT solutions, the stream workload is preferred more than the event-driven workload. The specification and processing requirements of such workload can change with the course of time. It largely depends on the sensing frequency of associated IoT devices and the intermediate time between two consecutive

input explicitly defines the idle time of host Fog nodes. The stream workload for Fog-based applications is discussed in [120], [102], [124], [125], [93], [126], [127], [94], [80] and [95].

2.3.6 Research Gaps in Application Architecture

Table 2.3 summarizes the existing concepts related to application architecture in Fog computing. Although there are a notable number of works, some issues related to this aspect of application management are yet to be investigated. They are listed as:

1. The execution of one application having particular programming model can trigger another application with different programming model. In such cases, the dynamic reconfiguration of Fog nodes is required. However, in existing works, only the static configuration of Fog nodes have been discussed [128].

2. The varying service type of applications can affect the networking capabilities of the host Fog nodes and degrade the service time the applications. Nevertheless, the existing approaches barely consider multiple service types of an applications simultaneously while determining a suitable placement options for them [129].

3. There are some research works denoting that the higher sensing frequency of IoT devices is required for better accuracy [24]. However, they have not considered that the high streaming rate of data creates immense processing burden on the Fog nodes.

4. Although, monolithic applications alleviate the constraints of inter-nodal communication delay, they are less modular. Conversely, the distributed application offer scalability but their service often gets affected by the limitations of underlying network. Although dynamic modularization of applications as per the context of Fog network is required, current researches only focus on the fixed functional layouts [26].

Works	Application Architecture					Works	Application Architecture				
	Program Model	Service Type	Workload Type	Interaction Method	Functional Layout		Program Model	Service Type	Workload Type	Interaction Method	Functional Layout
[73]		Actuation	Event-driven		Monolithic	[82]		Invocation			Module
[35]			Stream		Monolithic	[97]	Thread		Stream		
[81]	Dataflow	Actuation, Storage	Event-driven		Module	[74]		Display	Stream	Shared Database	Monolithic
[130]			Stream		Monolithic	[26]	Dataflow	Display			Module
[84]	Dataflow				Module	[115]		Display	Event-driven		
[75]			Event-driven		Monolithic	[83]	Dataflow	Invocation, Display	Stream		Module
[76]			Event-driven		Monolithic	[27]	Dataflow	Display			Module
[56]	Thread		Event-driven		Monolithic	[116]		Display	Event-driven		
[86]	Dataflow	Storage			Module	[87]		Actuation	Event-driven		Module

[131]			Stream		Monolithic	[77]			Event-driven		Monolithic
[79]		Storage	Stream	Shared Database	Monolithic	[132]	Dataflow				Module
[78]			Stream		Monolithic	[133]	Dataflow		Stream		Module
[98]	Dataflow, Thread		Stream			[88]	Dataflow				μ-service
[117]	Dataflow	Display	Stream		Module	[119]	Dataflow	Actuation			Module
[89]	Dataflow				μ-service	[90]		Invocation, Actuation			μ-service
[99]	Thread		Event-driven			[134]			Stream		Monolithic
[31]			Stream	Message		[135]	Dataflow		Stream		Module
[29]	Dataflow			REST	Module	[136]			Event-driven		Monolithic
[120]			Stream	Message		[100]	Thread	Display	Event-driven		Monolithic
[91]	Dataflow		Event-driven	Message	μ-service	[102]	Map-Reduce	Display	Stream	Shared Database	Monolithic
[109]	Dataflow	Actuation				[124]			Stream		Monolithic

[125]	Thread			Stream				[110]	Dataflow	Actuation				Module
[111]	Dataflow				Message	Module		[28]		Display				Module
[92]					REST	μ-service		[93]				Stream		μ-service
[112]	Dataflow			Event-driven		Module		[126]				Stream		Monolithic
[113]	Dataflow				Shared Database	Module		[103]	Map-Reduce	Actuation		Event-driven		
[85]	Dataflow					Module		[127]				Stream		Monolithic
[121]				Event-driven		Monolithic		[122]				Event-driven		Monolithic
[101]	Thread			Event-driven				[114]	Dataflow			Event-driven		
[11]				Event-driven		Monolithic		[118]		Display		Event-driven		Monolithic
[94]	Dataflow	Display		Stream		μ-service		[30]	Dataflow				REST	μ-service
[80]				Stream		Monolithic		[123]				Event-driven		Monolithic

Table 2.3: Summary of existing concepts on application architecture

2.4 Application Placement

The task distribution problem in Fog computing can be solved to a great extent if the applications are placed considering the future processing commitments of the Fog nodes. Additionally, the opportunistic placement of the applications can be a potential factor to standardize the Fog and Cloud integration. Moreover, while placing the applications, the resource orientation and their status are studied extensively. Such studies can play a vital role to update the application architecture dynamically and ensure proactive application maintenance. Fig. 2.5 depicts a taxonomy of various elements relevant to the application placement. Their descriptions are given below.

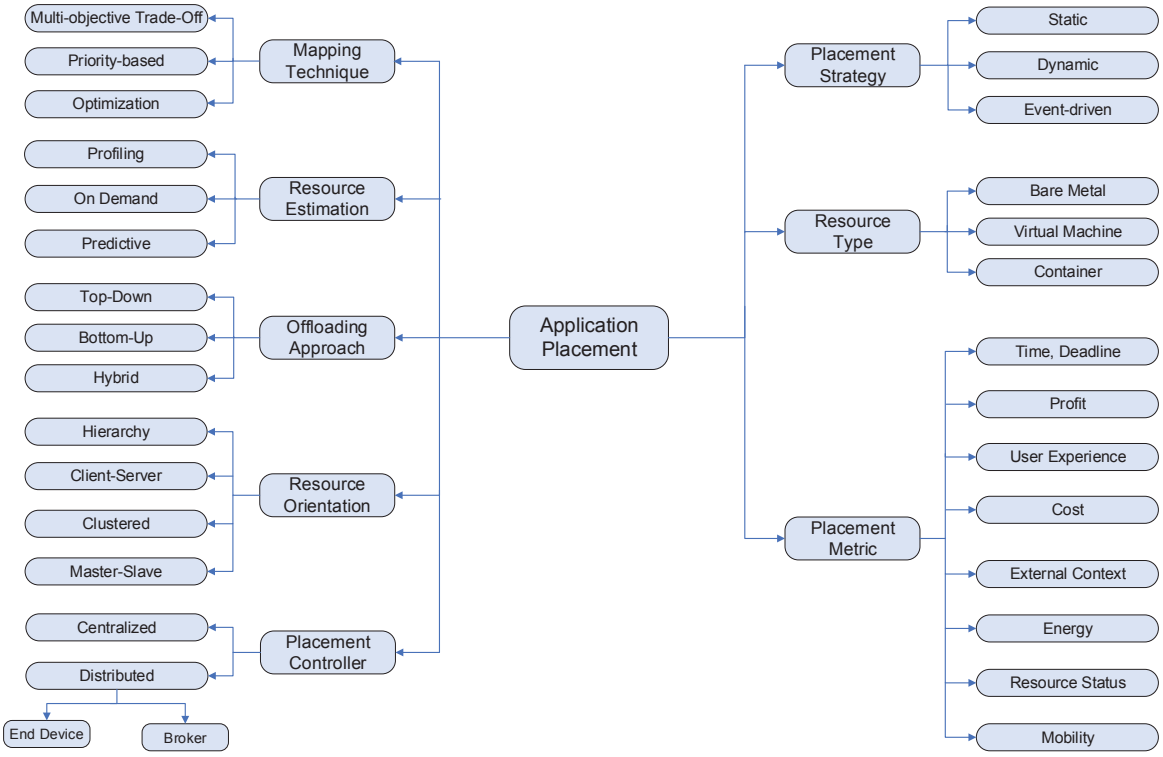

Figure 2.5: Taxonomy on application placement

2.4.1 Resource Estimation

The approximate resource consumption of an application is needed to estimate while placing the application in a resource-constrained Fog environment. It assists in defining

the characteristics of an application; whether it is compute intensive, I/O intensive or disk intensive. Three different types of resource estimation techniques are discussed for Fog computing.

Profiling

When a limited number of Fog nodes reside in a Fog environment and the specifications of requested applications remain static, the profiling technique is predominantly used to estimate the resources for an application. In this technique, an application is executed separately on each Fog node and associated performance parameters such as processing time, propagation time and energy consumption are monitored. Based on the accumulated information, the suitable resources for the subsequent executions of the application are selected.. Application profiling is discussed in [115], [119], [137], [127], [96] and [138].

Predictive

In this technique, based on the past execution patterns, the appropriate resources for an application are determined. This technique is highly applicable when a Fog environment supports dynamic provisioning of its component Fog nodes and the specifications of requested application vary. Compared to the profiling of applications, this technique is highly scalable, however, its results can be less precise. On the other hand, profiling depends on the physical deployment whereas prediction relies on the mathematical implications. The predictive resource estimation for Fog is discussed in [139], [140], [86], [95] and [141].

On Demand

In some cases, the resource provisioning is conducted based on the expectations of user and their instant demand. This technique differs from profiling or predictive techniques where the resource estimation depends on the application characteristics. On-demand estimation of resources is considered in [130], [142], [143], [11] and [24].

2.4.2 Offloading Approach

An offloading approach helps to manage the applications and their associated data as per the up-link and down-link network overhead of the host Fog nodes. In literature, three types of offloading approaches have been found for the Fog-based applications.

Bottom-Up

It is the most generic approach for offloading. in this approach, the requests regrading application services and relevant data are directly forwarded to the Fog nodes from the IoT devices or users. This approach for Fog-based applications is highlighted in [134], [144], [29], [124], [145], [121], [34] and [146].

Top-Down

Unlike bottom-up offloading, the top-down approach pushes the workload and programs of an application from Cloud to Fog nodes as per the requests of the end-users. It is often applied when Fog nodes are used for the content distribution. This offloading approach is discussed in [147], [148], [119], [149], [150], [151] and [152].

Hybrid

Apart from the aforementioned offloading approaches, Fog nodes can share application programs among themselves. During such interactions, the offloading follows a hybrid pattern of top down and bottom up. This approach is considered in [120], [37], [122], [153], [80], [154], [155] and [96] for Fog computing environments.

2.4.3 Resource Orientation

It signifies how the computing resources are arranged in Fog environments. When different applications are working collaboratively or deployed in distributed manner across multiple Fog nodes, the inter-nodal communication delay becomes very impor-

tant factor to ensure their QoS. It largely depends on the resource orientation. There are four types of resource orientation in Fog.

Hierarchy

In this orientation, Fog nodes are arranged in hierarchical levels between the communication path of IoT devices and Cloud datacentres. In the lower level, the number Fog nodes is higher compared to that of the higher levels. Conversely, as the level number goes higher, the inter-nodal communication delay increases. This orientation assists in vertical scaling of the resources. Hierarchical resource orientation is highlighted in [73], [81], [156], [157], [74], [84], [75], [26], [158] and [138].

Clustered

In clustered resource orientation, all Fog nodes are directly or logically connected to each other and share the information among themselves with high throughput communication channels. In most of the cases, the external communication, resource management and resource discovery operations within a Fog cluster are supervised by a dedicated Fog node. The clustered orientation provides more scope for horizontal scaling than the hierarchical orientation. This type of resource orientation is discussed in [35], [76], [116], [133], [88], [98], [159], [89], [100] and [99].

Client-Server

It enables a set of Fog nodes to work as the servers while lets others to act as the clients. The client Fog nodes request the server Fog nodes to process their forwarded data. This interaction can took place both in vertical and horizontal directions. This orientation is often regarded as a combination of clustered and hierarchical orientation. The client-server-based resource orientation is discussed in [160], [161], [148], [83], [162], [115], [140] and [56].

Master-Slave

In this orientation, a master Fog node distribute the data processing responsibilities to other slave Fog nodes and explicitly manages their operations. After receiving the service outcomes from the slave nodes, the master node accumulates them and forward the final results to the destination as per the service type. This orientation is more efficient in distributing the computing responsibilities than the client-server orientation. Master-Slave resource orientation is considered in [163], [142], [78], [92], [102], [30], [152] and [24].

2.4.4 Placement Controller

It defines the logical location of an entity that manages the overall application management operations in Fog computing. Furthermore, it assists in estimating the waiting time from requesting to placing an application in Fog environment which consequently drives the overall performance of the system. There are two types of placement controller widely visible in Fog computing.

Centralized

This type of placement controller locates in a commonly accessible place by the Fog nodes and poses a global view of the Fog environment. Generally, they are hosted in Cloud datacentres and supervise the Fog nodes residing at the edge network. The centralized placement controller for Fog-based applications is dissussed in [86], [119], [149], [92], [164], [102], [165], [166], [145] and[121].

Distributed

Unlike the centralized controller, the distributed placement controllers manage the Fog nodes based on a local view of the Fog environments. They are broadly classified into two categories.

• *End Device:* In this type of distributed placement controller, the IoT devices and the Fog nodes not only perform their predefined responsibilities such as data sensing

and data processing, but also take the application management decisions for other Fog nodes. End devices are used in [167], [99], [134], [31], [135], [144], [136], [168], [120] and [169] as application placement controllers for Fog environments.

• Broker: In contrast to end devices, this type of controllers are considered dedicated for application management operations in Fog environments. The brokers reside in proximity of the Fog nodes and interact with the external entities on behalf of the Fog nodes and vice-versa. The broker based placement controller for IoT applications is discussed in [170], [36], [78], [171], [79], [143], [137], [124], [91] and [172].

2.4.5 Mapping Technique

Based on different parameters, an application placement policy provides the mapping of the applications with respect to the Fog nodes and their virtualized instances. The complexity of the adopted mapping technique defines the runtime of the policy which consequently denotes its responsiveness. Three different types of mapping techniques are commonly used in Fog computing.

Priority-based

This technique prioritizes an application placement on particular Fog node or virtualized instances. Generally, the heuristic approaches such as best fit and first fit are commonly used for prioritization of the applications. The priority-based mapping technique is discussed in [161], [82], [128], [100], [92], [136], [169], [137], [125] and [85].

Optimization

This technique either maximizes or minimizes one particular objective function while placing the applications in Fog environments. Although optimization provides the best mathematical solution of a problem, this technique takes more time to operate than prioritization. Different types of optimization approaches such linear, non-linear and constrained are widely studied in Fog computing. This type of mapping technique is applied in [84], [115], [26], [139], [87], [165], [121] and [138].

Multi-objective Trade-off

Unlike optimization, the multi-objective trade-off can maximize or minimize two or more objectives such as time-energy, time-cost and cost-QoE simultaneously while placing the applications. Different metaheuristic approaches such as particle swarm, evolutionary algorithms, game theory and multiobjective optimization are used for making trade-off among various application placement metrics. The multi-objective trade-off for placing applications in Fog environments is highlighted in [81], [163], [76], [140], [164], [124], [122], [153] and [96].

2.4.6 Placement Strategy

The iterative execution of applications in Fog environments on the arrival of their inputs, or data processing life cycle. The placement algorithms need to consider these issues so that it can detect suitable hosts for different application. Placement strategy helps to define how frequently the placement algorithms are required to be executed for subsequent execution of an application. There are three types of placement strategies.

Static

In this strategy, placement algorithm is executed once for each application and at the host, the application is kept running. Inputs of an application are directly sent to its host from the IoT devices as the processing destination remain same for all of them. The static application placement strategy for Fog is discussed in [160], [156], [148], [87], [131], [173], [133], [79], [149] and [167].

Dynamic

In Fog, an application can have multiple replica running or can be terminated by processing a single input. For both cases, the placement algorithm requires to be executed for each arrival of its input to determine where to schedule the input or where to execute the application next. Such dynamic of placement strategy is highlighted in [73], [147], [157], [88], [159], [89], [164], [165], [172] and [174].

Event-driven

An application often requires to be relocated from one host to another. It can be occurred by mobility of the source and destination, preemption, Fog node consolidation and service migration. In such cases, after initial placement, further scheduling of applications is conducted occasionally based on the occurrence of the event. Event driven strategy for Fog-based application placement is considered in [74], [75], [26], [100], [166], [121], [34], [30], [175] and [96].

2.4.7 Resource Type

Fog nodes contain necessary resources such as processing cores, memory and bandwidth to assist the execution of applications. They can also support multi-tenancy. Resource type denotes the internal features of the host of the applications. It helps in validating the scope of dynamic allocation of resources during application run-time. Three different types of resource type is discussed in Fog computing.

Bare Metal

In this type of resources applications are directly placed to the Fog nodes. Applications access its physical hardware through the host operating system. It can support multi-tenancy without explicit isolation of the application execution unit. Bare metal is considered in [115], [75], [26], [142], [56], [131], [176], [159], [168] and [124].

Virtual Machine

In contrast to bare metal resources, virtual machines exploits hardware level virtualization so that multiple operating systems can run independently on a single Fog node. They run on top of an abstraction layer named hypervisor that enables the sharing of hardware among different virtual machines. Such type of resources is highlighted in [130], [177], [170], [173], [167], [135], [144] and [164].

Container

This type of virtualized resources is lightweight compared to virtual machines and offers operating-level virtualization. In opposition to bare metals, containers isolate processes with required application packages and they are highly portable across multiple Fog nodes. Containers are used in [163], [143], [31], [92], [29], [120], [166], [91], [101] and [30] for application placement in Fog.

2.4.8 Placement Metric

The main intention of placing applications in Fog can vary according to the requirements of users, service providers and working environments. Placement metric refers to the parameters that set the objectives of application placement in Fog environments. A wide variety of placement metrics are noted in Fog computing. They are described bellow.

Time, Deadline

It signifies the aim of minimizing application service delivery time and meeting the specified deadline. It can also incorporate the computation time, data propagation time and service deployment time while setting the placement objective. This placement metric is used in [161], [115], [26], [158], [33], [176], [119], [167], [99] and [159].

Profit

Service providers get benefited when the applications are deployed in Fog with a view to maximizing their profit and revenue. It often leads the providers to offer application execution in Fog as an utility. Profit is used in [35], [177], [121] and [152] as a placement metric for the applications in Fog.

User Experience

Service requirements of users and their affordability level can change with the course of time. If these issues are not met during the application placement, user experience can

degrade. It can also resist users to execute the applications through Fog in future. This placement metric is used in [147], [128], [142], [170] and [144].

Cost

There are different monetary costs such as infrastructure deployment cost, operational cost and instance rental cost are associated with Fog computing. Cost as placement metric refers to its minimization during the application placement in Fog. Cost is considered in [74], [83], [76], [116], [133], [88], [144] and [138] to place the applications in Fog.

External Context

There exist several external parameters including the relinquish rate and the activity of users, reliability of Fog nodes, the popularity of application services, data size and the sensing frequency of IoT devices that drive the decision of application placement in Fog. Such external contexts are discussed in [148], [97], [82], [140], [86], [77], [95], [168], [178] and [96] while placing the applications in Fog.

Energy

Fog nodes can utilize both renewable and non-renewable energy to operate. However, the energy consumptions of Fog nodes are subjected to the environmental and supply-demand related issues. There are some researches including [81], [160], [157], [82], [84], [162], [142], [140], [56] and [171] that highlight energy as one of the dominant factors for making application placement decisions in Fog computing.

Resource Status

Fog nodes are widely heterogeneous in terms of their processing power, networking interfaces, storage capacity and operational platform. Assessment of these status parameters is very important to efficiently manage the applications over them. The resource

status is given higher preference in [73], [156], [157], [128], [130], [75], [163], [139], [140] and [87] while placing the applications in Fog environments.

Mobility

In the context of Fog computing, both the IoT devices and the Fog nodes can move from one location to another very frequently. This feature of Fog computing can affect the service delivery and occur migration of application execution among the Fog nodes. Taking cognizance of these issues, mobility is considered in [170], [36], [29], [165], [85], [166], [121], [175] and [96] as an application placement metric.

2.4.9 Research Gaps in Application Placement

Table 2.4 summarizes the existing application placement techniques in Fog computing. Although an extensive amount of research has been conducted on this aspect of application management, there are still some gaps. They are listed below.

1. For remote areas, many researches suggest to use renewable power sources to run the Fog nodes as the grid-based energy is costly to facilitate [124]. However, very few of them consider that the availability of renewable energy is subjected to uncertainty and environmental context and take the required measures to solve the problem.

2. The distribution of application placement tasks across multiple entities can reduce the management overhead. However, the existing works have not considered the elevation in decision-making time that can occur due to such distribution [122].

3. Most of the existing works prefer Cloud to place applications when there is no resource available in the corresponding Fog infrastructure [35]. However, they have not discussed the confederation of Fog infrastructure owned by different service providers. As a consequence, the scope of performance improvement lessens and the providers fail to harness the monetary benefits.

4. The consolidation of Fog nodes can save the energy. However, this operation can alter the topology and orientation of Fog resources and affect the collaborative execution of applications. Despite of such impact, current researches barely look into this issue [170].

Works	Application Placement							
	Mapping Technique	Resource Estimation	Offloading Approach	Resource Orientation	Placement Controller	Placement Strategy	Resource Type	Placement Metric
[73]				Hierarchy	Centralized	Dynamic	VM	Time, Resource
[81]	Trade-off		Bottom-Up	Hierarchy	Broker		Bare Metal	Time, Energy
[160]	Optimization		Bottom-Up	Client-Server	Broker	Static	Bare Metal	Time, Energy
[147]	Optimization		Top-Down			Dynamic		QoE
[156]	Optimization			Hierarchy	End Device	Static	Bare Metal	Time, Resource
[157]			Bottom-Up	Hierarchy	End Device	Dynamic	VM	Energy, Resource
[161]	Priority			Client-Server	Centralized	Dynamic	VM	Time
[148]			Top-Down	Client-Server	Centralized	Static	Bare Metal	Context
[35]	Optimization			Cluster	Centralized	Static	VM	Profit
[97]			Bottom-Up		Broker	Dynamic	VM	Context
[82]	Priority	Predictive				Dynamic	Bare Metal	Context, Energy
[74]	Optimization			Hierarchy		Event-driven	VM	Cost
[128]	Priority		Bottom-Up	Client-Server	Broker	Dynamic	VM	QoE, Resource

[130]	Optimization	On Demand	Hybrid	Client-Server	End Device, Broker		VM	Time, Resource
[83]	Optimization			Client-Server	End Device	Static		Time, Cost
[84]	Optimization			Hierarchy			Bare Metal	Time, Energy
[162]	Optimization		Bottom-Up	Client-Server	Broker	Static		Time, Energy
[115]	Optimization	Profiling	Bottom-Up	Client-Server		Static	Bare Metal	Time
[75]				Hierarchy	Broker	Event-driven	Bare Metal	Time, Resource
[163]	Trade-off		Bottom-Up	Master-Slave	End Device		Container	Resource
[26]	Optimization			Hierarchy		Event-driven	Bare Metal	Time
[142]	Priority	On Demand		Master-Slave	Broker		Bare Metal	Time, QoE, Energy
[139]	Optimization	Predictive				Event-driven	Bare Metal	Resource
[76]	Trade-off			Cluster			Bare Metal	Time, Cost
[158]	Priority	Profiling		Hierarchy		Static	Bare Metal	Time
[177]	Optimization		Hybrid		Centralized	Dynamic	VM	Time, Profit
[140]	Trade-off	Predictive		Client-Server	End Device			Context, Energy, Resource
[170]	Optimization				Broker	Event-driven	VM	QoE, Resource, Mobility

[36]	Optimization			Hierarchy	Broker	Event-driven	VM	Mobility
[56]	Optimization		Bottom-Up	Client-Server	End Device	Static	Bare Metal	Time, Energy
[116]	Priority			Cluster	Centralized	Static	VM	Time, Cost
[86]		Predictive		Hierarchy	Centralized	Event-driven	Bare Metal	Context
[87]	Optimization			Hierarchy		Static		Time, Resource
[131]				Hierarchy		Static	Bare Metal	Time, Energy
[77]				Hierarchy		Dynamic	Bare Metal	Time, Context
[95]	Priority	Predictive		Hierarchy	End Device	Event-driven		Context
[78]				Master-Slave	Broker		Bare Metal	Cost, Resource
[33]	Optimization			Hierarchy			Bare Metal	Time
[171]	Priority		Bottom-Up	Client-Server	Broker	Dynamic	Bare Metal	Time, Energy
[176]	Optimization		Bottom-Up	Client-Server	End Device	Dynamic	Bare Metal	Time
[173]	Optimization			Hierarchy		Static	VM	Time, Cost
[133]	Optimization			Cluster		Static	Bare Metal	Time, Cost
[79]	Optimization	Profiling		Hierarchy	Broker	Static	Bare Metal	Time, Cost
[88]	Optimization			Cluster	End Device	Dynamic	Bare Metal	Cost, Energy

[98]	Priority		Hybrid	Cluster	End Device	Dynamic	Bare Metal	Time, Resource
[119]	Optimization	Profiling	Top-Down		Centralized	Event-driven		Time
[143]		On Demand			Broker		Container	Cost
[159]	Optimization		Hybrid	Cluster	End Device	Dynamic	Bare Metal	Time
[89]	Priority			Cluster		Dynamic	Bare Metal	Resource
[149]	Optimization		Top-Down	Hierarchy	Centralized	Static		Cost
[100]	Priority		Hybrid	Cluster	End Device	Event-driven		Resource
[167]			Bottom-Up	Hierarchy	End Device	Static	VM	Time
[99]	Optimization		Hybrid	Cluster	End Device	Static	Bare Metal	Time
[134]	Optimization		Bottom-Up		End Device		Bare Metal	Time, Cost, Energy
[31]				Hierarchy	End Device		Container	Time, Resource
[135]				Hierarchy	End Device		VM	Energy
[92]	Priority			Master-Slave	Centralized		Container	Resource
[144]	Optimization		Bottom-Up	Hierarchy	End Device	Dynamic	VM	QoE
[29]			Bottom-Up	Client-Server	End Device	Dynamic	VM, Container	Resource, Mobility
[136]	Priority			Client-Server	End Device	Dynamic		Time

[168]				Client-Server	End Device		Bare Metal	Context
[120]	Optimization		Hybrid	Hierarchy, Cluster	End Device		Container	Energy
[164]	Trade-off			Client-Server	Centralized	Dynamic	VM	Time, Energy
[102]				Master-Slave	Centralized		Bare Metal	Energy
[169]	Priority			Hierarchy	End Device			Resource
[109]	Optimization			Client-Server			Bare Metal	Time
[137]	Priority	Profiling		Client-Server	Broker		VM	Time
[124]	Trade-off		Bottom-Up	Hierarchy	Broker		Bare Metal	Time, Cost, Energy
[125]	Priority			Hierarchy	End Device		VM	Energy
[165]	Optimization			Client-Server	Centralized	Dynamic	Bare Metal	Mobility
[85]	Priority			Hierarchy			Container	Mobility
[37]			Hybrid	Cluster	End Device		Bare Metal	
[166]					Centralized	Event-driven	Container	Mobility
[179]	Priority			Hierarchy			Bare Metal	Resource
[91]	Priority			Cluster	Broker		VM, Container	Time, Resource

[126]	Optimization			Cluster	End Device			Time, Cost
[172]	Optimization			Hierarchy	Broker	Dynamic		Time, Energy
[113]	Priority			Cluster			Bare Metal	Cost
[145]	Optimization		Bottom-Up		Centralized		Bare Metal	Time, Energy
[180]	Optimization			Cluster			Bare Metal	Time
[101]	Priority				Broker		VM, Container	Resource
[93]	Optimization	Profiling		Cluster	Broker			Time, Resource
[127]		Profiling		Cluster	Broker	Dynamic		Time, Resource
[121]	Optimization		Bottom-Up	Hierarchy	Centralized	Event-driven		Profit, Mobility
[34]			Bottom-Up	Client-Server	Broker	Event-driven		Time, Cost
[178]	Optimization			Hierarchy	Broker			Time, Context
[122]	Trade-off		Hybrid	Client-Server	End Device			Time, Energy
[146]			Bottom-Up	Client-Server	Broker		Bare Metal	
[30]				Master-Slave	End Device	Event-driven	Container	Resource
[114]	Priority			Hierarchy		Static	Bare Metal	Time
[150]			Top-Down	Client-Server	Centralized	Dynamic	Container	

[153]	Trade-off		Hybrid		End Device	Static	Bare Metal	Time, Energy
[11]	Optimization	On Demand		Client-Server			VM	Time, Cost
[118]	Optimization			Client-Server	End Device		Container	Time
[80]	Priority		Hybrid		Centralized	Static		Context, Resource
[103]				Cluster		Static	Bare Metal	Resource
[94]	Optimization			Cluster			Bare Metal	Energy, Resource
[154]			Hybrid	Client-Server			Bare Metal	Time, Cost, Energy
[141]	Priority	Predictive		Cluster	Centralized	Dynamic		Energy
[174]	Optimization			Client-Server	End Device	Dynamic	Bare Metal	Time
[132]	Trade-off			Cluster	Broker		VM	Time, Cost, Energy
[175]	Optimization			Cluster		Event-driven		Mobility
[151]	Optimization		Top-Down	Hierarchy	Centralized	Dynamic	Bare Metal	Resource
[155]	Optimization		Hybrid	Client-Server	End Device		Bare Metal	Energy
[123]	Optimization		Top-Down	Client-Server	Centralized			Time, Cost
[152]	Priority		Top-Down	Master-Slave	Centralized	Dynamic		Profit
[96]	Trade-off	Profiling	Hybrid	Client-Server	Broker	Event-driven	Container	Time, Context, Mobility

[24]		On Demand	Bottom-Up	Master-Slave	Broker	Dynamic	Bare Metal	
[138]	Optimization	Profiling		Hierarchy	Broker		Bare Metal	Cost
[181]	Priority		Bottom-Up	Hierarchy	End Device	Event-driven	Bare Metal	Time, Mobility
[182]	Priority			Cluster	End Device		Bare Metal	Context
[183]	Priority		Hybrid	Hierarchy	Broker			Time, Energy
[184]		Predictive		Hierarchy		Dynamic	VM	Resource
[185]	Optimization				End Device	Static	VM, Container	Energy, Resource
[186]	Priority	On Demand		Cluster	Centralized			Resource
[187]	Priority			Hierarchy, Cluster	Broker		Bare Metal	Time, Resource
[188]	Optimization		Top-Down		Centralized		Container	Context
[189]	Optimization			Hierarchy	Broker			Time, Energy
[190]	Priority			Hierarchy	Broker	Static		Time, Cost
[32]		Predictive		Hierarchy	Broker		VM	Time, Cost
[191]			Hybrid	Hierarchy	End Device		Bare Metal	Energy
[129]	Optimization		Bottom-Up	Master-Slave	Broker		Bare Metal	Cost

[192]			Hybrid		Broker	Event-driven	Container	Energy, Context
[193]	Priority		Hybrid	Hierarchy	End Device	Static	Bare Metal	Time, Resource
[194]				Hierarchy	Centralized		Bare Metal	Time, Mobility
[195]	Priority		Top-Down		Centralized		VM	Energy, Resource
[196]	Optimization	On Demand				Dynamic	Container	Mobility
[197]	Trade-off			Hierarchy	End Device		Bare Metal	Time, Energy

Table 2.4: Summary of existing application placement techniques

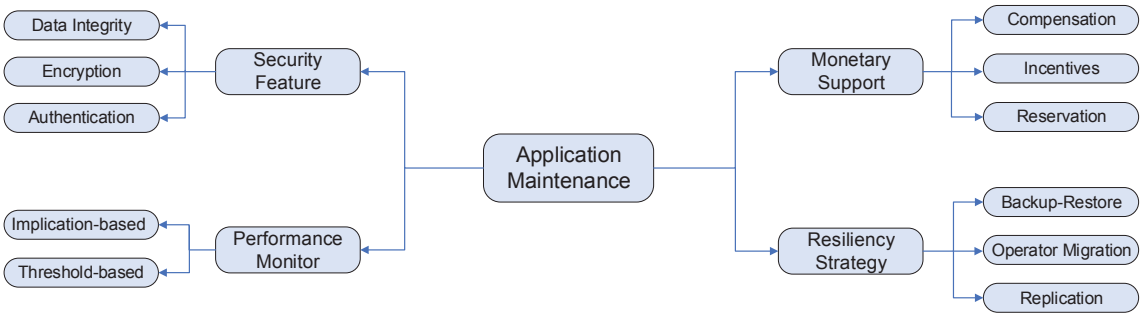

Figure 2.6: Taxonomy on application maintenance

2.5 Application Maintenance

Robust application maintenance operations are required to secure the access of all legitimate users to the application outcomes in Fog environments. Additionally, if these operations are conducted in both proactive and reactive manner, the uncertain failure of Fog nodes and the performance degradation of applications can be mitigated to a certain extent. The demand of application maintenance can also trigger the necessity to introduce additional features such as check pointing and partitioning to the functional layout of application architecture. Moreover, the requirements of operator migration can initiate the event-driven application placement. Fig. 2.6 illustrates a taxonomy of different elements of application maintenance. In following subsections, they are discussed in detail.

2.5.1 Security Feature

Fog computing functions at the edge network. The attackers can access the Fog infrastructure easily and resist the smooth execution of applications by generating security threats including information impairment, identity disclosure, replay and Denial of Service (DoS) attacks. Therefore, it is required to specify the security features while executing the applications. Three different types of security features are widely used in Fog.

Data Integrity

In some cases, various sensitive data and their processing outcomes; for instance, the electronic health report and the patient's conditions are consistently analysed by different parties including hospitals and insurance companies. Fog computing needs to offer easy access to those data with a guarantee of no alteration. It helps to ensure the data integrity of the applications in Fog environments. This security feature is discussed in [24], [169] and [85] for maintaining applications in Fog.

Encryption

In Fog computing, extensive data exchange operations are conducted between the IoT devices, Fog nodes and Cloud datacentres. Encryption not only hides the details of the transferred data, but also protects the credentials of legitimate users to access the Fog resources. Encryption is used in [148], [27], [86], [37] and [126] as a security feature.

Authentication

It helps to identify the legitimate user of application services and Fog resources. Sometimes, authentication is robustly applied at the receiver side to control the data access rate of different entities. Authentication is considered in [86], [31], [120], [169], [178], [146] and [150] for application maintenance in Fog.

2.5.2 Performance Monitor

A continuous performance monitoring of resources is required to take the management decisions during application run-time. Consequently, it helps to maintain the execution flow of the applications at the desired level. Two different types of performance monitoring techniques are widely used in Fog computing.

Implication-based

In this technique, the current synthesis of application and resources are implied to predict the future performance trends. Implication-based performance monitoring assists in maintaining the execution of the applications in a proactive manner. Implication-based performance monitoring for Fog is discussed in [147], [130], [139], [167], [137], [28] and [121].

Threshold-based

When an application is executed in a Fog node, its key performance indicator such as processor usage and memory consumption are compared with a dynamically set or predefined threshold value. If the state of the indicator does not match with the threshold, necessary decisions are taken to continue the execution of the application in Fog environments. Unlike the implications, this approach helps in reactive application maintenance. Threshold based performance monitoring is applied in [128], [115], [75], [87], [77], [117], [88], [159], [112], [111], [28] and [101].

2.5.3 Monetary Support

It defines how Fog service providers nurture the economic interests of the users while executing their requested applications. It helps providers to attain the trust of the users that controls the relinquish rate. Three types of monetary supports are studied in Fog.

Compensation

Due to some uncertain events such as node failure and security threats, the service level agreement between the users and the providers can be violated. In these cases, provider offers compensation to users so that they can rely on the Fog based execution of their requested applications. It also helps providers in quantifying the aim of reducing service violations. Compensation as a monetary support is discussed in [198], [199] and [143].

Incentives

At the edge network, there exist different idle computing components that can be used as potential Fog nodes. Fog service provider often harness those resources to meet the additional demand of users by providing incentives to the owner of the resources. In some cases, for relaxing the stringent requirements, the users also receive incentives from the service providers. This type of moetary support is adopted in [78], [176], [149], [154] and [152].

Reservation

It denotes that users can provision of a certain number of applications at any given time on fixed charges despite of the current load of the Fog infrastructure. In this case, Fog computation acts as a subscription based utility. Reservation is considered in [77], [200] and [201].

2.5.4 Resiliency Strategy

It denotes how Fog computing continues application execution after the occurrences of uncertain events and failures. It is one of the main features of application maintenance that assists in enhancing the reliability of the system. Three types of resilience strategies are widely adopted in Fog computing.

Backup-Restore

Here, the intermediate results and different execution phase of an application are continuously stored so that the execution of the application can be re initiated soon after the anomaly without starting it from the very beginning. It is performed by setting some check points and temporary storage operations during application run-time. Backup-restore is feasible when a one-to-one interaction happens between the users and the applications. It is used in [115], [168], [90], [37], [110] and [111] as a resiliency strategy for application maintenance.

Replication

In this resiliency strategy, multiple instances of an application are run across different Fog nodes. It ensures the satisfaction of user requests through the application even after the failure of its several instances. Unlike backup-restore, replication is well-suited for supporting the one-to-many interactions between the users and the applications. Replication is discussed in [97], [75], [163], [87], [33], [110], [126] and [34] for Fog computing environments.

Operator Migration

In case of node failure or mobility of the requesting entities, the execution of applications is often shifted from one node to another Fog node. It happens dynamically so that application execution continues without interrupting the user experience. As a resiliency strategy, operator migration differs from the backup-restore and replication because of its ease of scalability. It is considered in [170], [36], [95], [137], [166], [85], [127], [121] and [150].

2.5.5 Research Gaps in Application Maintenance

Table 2.5 summarizes the existing application maintenance operations in Fog computing. Although extensive research initiatives have been taken, there are still some issues that require to be investigated for efficient application maintenance in Fog computing. They are listed as below.

1. In many research works, compute intensive algorithms are used to secure the data transmission within Fog environments [202]. However, they have not considered that heavyweight security techniques slow down the legitimate access to application services and resources.

2. Streaming applications require reserved resources so that their processing destinations do not change very frequently. However, while making such arrangements, the existing works barely consider the waiting time of the other less-interactive applications [200].

3. There are a good number of research works that mention check pointing and replication as the means of fault tolerance in Fog computing [110]. However, to deal with the scarcity of resources, they have not provided any concrete model that can dynamically tune the frequency of check points within an application and change the number of replications in Fog environments.

2.6 A Perspective Application Management Framework

Fog computing provides a scope to distribute the application management operations across different tiers of the computing infrastructure, however, in other paradigms, this scope is very limited. To illustrate this feature of Fog, a perspective framework is depicted in Fig. 2.7. At each infrastructure level, the components of this framework performs some specific operations related to application management. They are summarized below.

2.6.1 Cyber Physical System Level

At this level, IoT-enabled CPSs reside that request the Fog environment for the services of different applications. In this case, the IoT Application Brokers (IABs) deployed at the Fog gateway level assist the CPSs. While making the requests, the CPSs forward the specification of the applications including the workload type, the frequency of incoming data, the form of application services and the QoS requirements such as service deadline, budget and user expectations to the IABs. Additionally, the CPSs can host some components of the requested applications for data pre-processing.

2.6.2 Fog Gateway Level

In Fog Gateway Level, an IAB consists of CPS Manager, Application Placement Engine (APE) and Workload Scheduler. The CPS manager contains the meta-data of multiple versions of an application having different programming models, functional layouts and interaction methods. The CPS manager also interacts with the APE to get the state of resources such as their orientation and type within the Fog infrastructure and Cloud

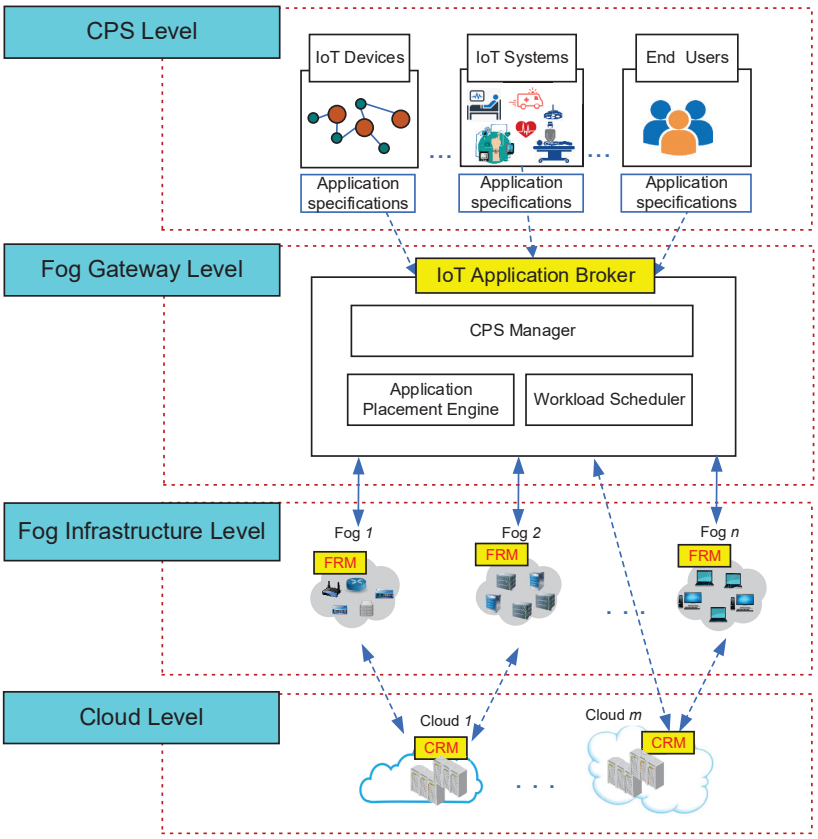

Figure 2.7: A perspective model for application management in Fog

level. Based on the accumulated information from different levels, the CPS manager determines the most suitable architecture of the applications for placement. Later, the APE estimates the resources of executing the application, identifies the placement metrics as per the application QoS requirements of the CPSs and sets the mapping technique accordingly. To place the applications physically over the resources, the APE communicates with the Fog Resource Manager (FRM) and Cloud Resource Manager (CRM) of the Fog infrastructure and Cloud level. After placement, the Workload Scheduler finds out the feasible placement strategy to dispatch the inputs to the applications based on the dynamics of the Fog environment.

2.6.3 Fog Infrastructure and Cloud Level

The FRMs and CRMs of Fog infrastructure and Cloud datacenter store the application executables and they are responsible for allocating resources for application execution. They also monitor the status and performance of the resources and conduct application maintenance operations including service backup and replication. Additionally, they deal with the uncertain node failures, resource outage and security attacks to ensure reliability during application execution. Based on their implications, the CPSs and IABs tune the specifications of the application architecture and modify the placement approaches.

Nevertheless, this framework only provides a abstract view of distributing application management operations in different infrastructure levels within the Fog computing environments. This framework can also contribute to develop new policies for runtime service orchestration, multi-level resource provisioning, application execution migration and Fog standardization.

Works	Application Maintenance				Works	Application Maintenance			
	Security Feature	Performance Monitor	Monetary Support	Resiliency Strategy		Security Feature	Performance Monitor	Monetary Support	Resiliency Strategy
[147]		Implication			[148]	Encryption			
[97]				Replication	[128]		Threshold		
[130]		Implication			[115]		Threshold		Backup-Restore
[75]		Threshold		Replication	[163]				Replication
[139]		Implication			[27]	Encryption			
[170]				Migration	[36]				Migration
[86]	Encryption, Authentication				[87]		Threshold		Replication
[77]		Threshold	Reservation		[117]		Threshold		
[95]				Migration	[78]			Incentives	
[33]				Replication	[176]			Incentives	
[88]		Threshold			[119]		Threshold		
[143]			Compensation		[200]			Reservation	
[159]		Threshold			[149]			Incentives	

[100]	Threshold	[167]	Implication
[31]	Authentication, Threshold	[168]	Backup-Restore
[120]	Authentication	[201]	Reservation
[90]	Backup-Restore	[112]	Threshold
[137]	Implication, Migration	[37]	Encryption, Backup-Restore
[110]	Backup-Restore, Replication	[111]	Threshold, Backup-Restore
[24]	Integrity	[28]	Implication, Threshold
[166]	Migration	[169]	Integrity, Authentication
[85]	Integrity, Migration	[126]	Encryption, Replication
[199]	Compensation	[101]	Threshold
[127]	Migration	[121]	Implication, Migration
[34]	Threshold, Replication	[178]	Authentication

[146]	Authentication	[150]	Authentication Migration
[94]	Replication	[154]	Incentives
[198]	Compensation	[175]	Migration
[155]	Threshold	[152]	Incentives

Table 2.5: Summary of existing application maintenance operations

2.7 Summary

Fog computing is gradually turning into an integral component of smart systems because of its wide-spread features for supporting IoT-driven use cases. In both academia and industry, numerous initiatives are taken to standardize the Fog computing concept for managing IoT applications. In this chapter, the existing application management approaches in Fog computing from the perspectives of application architecture, placement and maintenance are reviewed. Separate taxonomy for each of the aspects of application management and discussed their associated research gaps are presented. A perspective model for managing applications in Fog environments is also highlighted.

Chapter 3

Edge Affinity-based Application Management

The selection of applications for Fog-based placement becomes more complicated when the application characteristics in terms of urgency, size and flow of inputs are considered simultaneously. In this chapter, a policy for Fog environments is proposed that distributes application management tasks across the gateway and the infrastructure level. It classifies and places applications according to their Edge affinity. Edge affinity of an application denotes the relative intensity of different attributes coherent with its characteristics which are required to be addressed within Fog environments to meet its Quality of Service (QoS). The proposed policy also minimizes the service delivery time of applications in Fog infrastructure. Its performance is compared with existing application management policies in both iFogSim-simulated and FogBus-based real environments. The experiment results show that this policy outperforms others in combined QoS enhancement, network relaxation and resource utilization.

3.1 Introduction

Due to resource scarcity, it is difficult to accommodate every IoT application within Fog infrastructure. Inclusion of more nodes to resolve this issue can affect the economic aspects of Fog computing and intensify the communication complexities. In such a constrained scenario, infrastructure providers are often instigated to offer execution of IoT

This chapter is derived from:

- **Redowan Mahmud**, Kotagiri Ramamohanarao, and Rajkumar Buyya, "Edge Affinity-based Management of Applications in Fog Computing Environments", *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing (UCC'19)*, IEEE Press, Pages: 61-70, Auckland, New Zealand, December 2-5, 2019.

applications in Fog as a utility. It also urges users to provision a certain number of applications through Fog instances such as virtual machines and containers according to their affordability. Nevertheless, a system that deals with various applications; in particular, for a health monitoring system, it becomes complicated to select the applications for Fog-based placement. Assurance of their time-optimized service delivery also turns into a challenging task. However, these cases can be efficiently addressed in Fog by managing the applications based on their Quality of Service (QoS) requirements [158].

Distinctive characteristics of IoT applications help to identify their different QoS requirements. For example, user-defined deadline indicates whether an application is latency-sensitive or tolerant. Reduced data propagation delay is required for latency-sensitive applications to ensure their QoS [93]. Similarly, based on the data sensing frequency of associated IoT devices, execution of an application can be event-driven or stream-oriented. Streaming applications demand congestion-less data propagation so that their QoS can improve [203]. Moreover, applications that deal with images, audios, videos and large text files are required to process a huge amount of data per input than trivial applications addressing boolean data and short messages. They are usually known as data-intensive applications and encapsulate multiple data pre-processing operations such as data filtration, conversion and consolidation along with the actual data analysis operation [2]. Therefore, it is expected to execute them closer to data sources. Otherwise, the amount of data to be transferred through global Internet will increase, and both the computation and communication load on remote computing resources will aggravate. As a consequence, QoS of these applications will degrade. However, for a particular application, these characteristics are independent, and their intensity can vary from one to another. Therefore, it is not feasible to take management decision for different IoT applications based on a single characteristic.

There exist several policies that focus on service time, resource and workload-aware management of IoT applications in Fog computing environments [76] [204] [205]. They barely explore different application characteristics simultaneously and investigate their influence on application QoS requirements. In some cases, the Fog gateway devices that reside at the user premises and connect the IoT devices to Fog infrastructure, are assumed to perform all required tasks for managing the applications such as their selec-

tion and placement [127] [206]. When a large number of gateway devices interact with a Fog infrastructure, it is time-consuming to share the status of Fog instances among all gateways. For a gateway, it is also difficult to cope up with the dynamism of Fog infrastructure. Consequently, the synchronization problem amplifies, and the overhead of resource-constrained gateways increases.

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

- Proposes a policy for Fog environments that manages applications based on multiple characteristics and requirements across the gateway and the infrastructure level.
- Defines an innovative way to apply non-dominated sorting for application classification in Fog environments.
- Selects applications for Fog-based placement as per their character-driving attributes and optimizes their service delivery time in Fog infrastructure.

The rest of this chapter is organized as follows: after discussing related work in Section 3.2, the application context and system model are presented in Section 3.3. Section 3.4 proposes the Edge affinity-based application management policy. Section 3.5 evaluates the performance of the proposed policy in respect to existing policies. Finally, Section 3.6 concludes the chapter.

3.2 Related Work

Different application management policies have already been developed for Fog environments. Binh et al. [76] and Choudhari et al. [116] modeled separate policies to optimize execution time and cost by prioritizing applications based on user expectations and service deadline respectively. Nan et al. [207] conducted trade-off among service time and request loss rate while placing the applications. Venticinque et al. [93] modeled a policy that classifies applications as per their resource and energy requirements, and maximizes QoS by meeting deadline. Stavrinides et al. [205] prioritized applications based on workload and ensures least completion time. The policy in [208] optimizes application service time and enhances the user experience. Skarlat et al. [91] also explored time-optimized execution of applications with high resource utilization.

Furthermore, Xu et al. [209] discussed a framework that classifies applications based on deadline, and assists service migration and load distribution. The application management policy in [210] optimizes energy usage of instances while executing the applications. Taneja et al. [204] also developed a policy that prioritizes application placement on robust Fog nodes to enhance resource utilization. The policy in [211] allocates resources according to user-driven popularity of applications and executes them locally as per a threshold of computing cost. Similarly, Guerrero et al. [95] placed the most requested applications in Fog and improved network utilization and service latency.

A summary of related works along with the proposed policy is given in Table 3.1. In existing works, different characteristics of applications are not exploited simultaneously to identify their QoS requirements. User-defined deadline, amount of data to be processed and sensing rate of IoT devices are also disregarded while placing the applications. Consequently, they fail to leverage the capabilities of Fog computing in dealing with different sorts of applications. Here, we classify applications and facilitate their placement based on the relative intensity of different attributes those are coherent with their characteristics and required to be supported through Fog infrastructure for meeting their QoS. Our proposed policy also optimizes service delivery time of applications in Fog environments.

Work	Application characteristics			Prioritized selection	Optimizes	
	Data flow	Input size	Urgency		Time	Load
[76]		✓		✓	✓	
[116]	✓		✓	✓		
[207]	✓		✓		✓	✓
[93]	✓		✓		✓	✓
[205]	✓	✓	✓	✓		
[208]	✓		✓	✓	✓	
[91]		✓	✓		✓	✓
[210]		✓			✓	✓
[209]		✓			✓	✓
[204]		✓	✓	✓		✓
[211]	✓	✓		✓		✓
[95]	✓			✓	✓	
Edge affinity (This work)	✓	✓	✓	✓	✓	✓

Table 3.1: Summary of related work for Edge affinity-based management

3.3 Application and System Overview

The detailed description of the application and the system model for this chapter are given below.

3.3.1 Motivating Scenario

To clarify the application context considered in this chapter, we have used the content service delivery of Netflix as an analogy. This analogy has no direct relation with the proposed application management policy but can help to understand why the efficient classification and selection of the applications is required. As noted, Netflix is a streaming service where based on the category of subscription, a user can watch one, two or three different media contents at a time. Netflix-users do not care about what sorts of resources are used to enable these media contents; all that matters to them is whether they can access the allowable number of contents on demand. If a user asks for more

media contents at the same time, the user usually sets the preferences according to the quality of contents on Netflix and obtain the extra contents from other streaming services like YouTube or Stan. Such service provisioning is economical for users and assists providers to manage their resources efficiently [212]. We extend this scenario from a localized perspective where Netflix resembles the Fog infrastructure and media contents are the applications. Users can execute a certain number of applications through Fog infrastructure based on their requirements, affordability and resource availability. When more applications are needed to be executed, the allowable number of applications for Fog-based placement are selected from them. Our proposed Edge affinity-based management policy is capable of dealing with such application context in Fog environments. It facilitates the selection of applications having stringent QoS requirements so that the capabilities of Fog infrastructure can be harnessed extensively. Moreover, it forwards the applications with unmet demand to other Fog or Cloud infrastructure for execution.

3.3.2 Fog Environments

Different providers can deploy cluster of Fog nodes in various locations. Fig. 3.1 presents the Fog Clusters (FCs) deployed by provider A and B on location L. Providers can adopt any existing business models to manage the capital and operating expenses of such Fog infrastructure. FCs are accessible through Fog Gateways (FGs) located at the user premises. Each Fog node within an FC is capable of hosting different number of Fog instances such as virtual machines and containers as per its capacity. In an FC, the assignment of applications on Fog instances is managed by a specialized node named Fog Resource Manager (FRM) [213]. FRMs maintain a persistent communication with FGs that helps to bind the IoT devices with FCs. FGs receive placement requests for applications from the users. These applications are monolithic and can work independently. The placement request for an application includes the details of its character-driving attributes such as user-defined deadline, average amount of data per input and sensing frequency of IoT devices. Conversely, FRMs extract the developer-specified minimum resource requirements of applications along with the necessary meta-data from a catalogue service [24]. The allowable number of applications for provisioning on an FC is

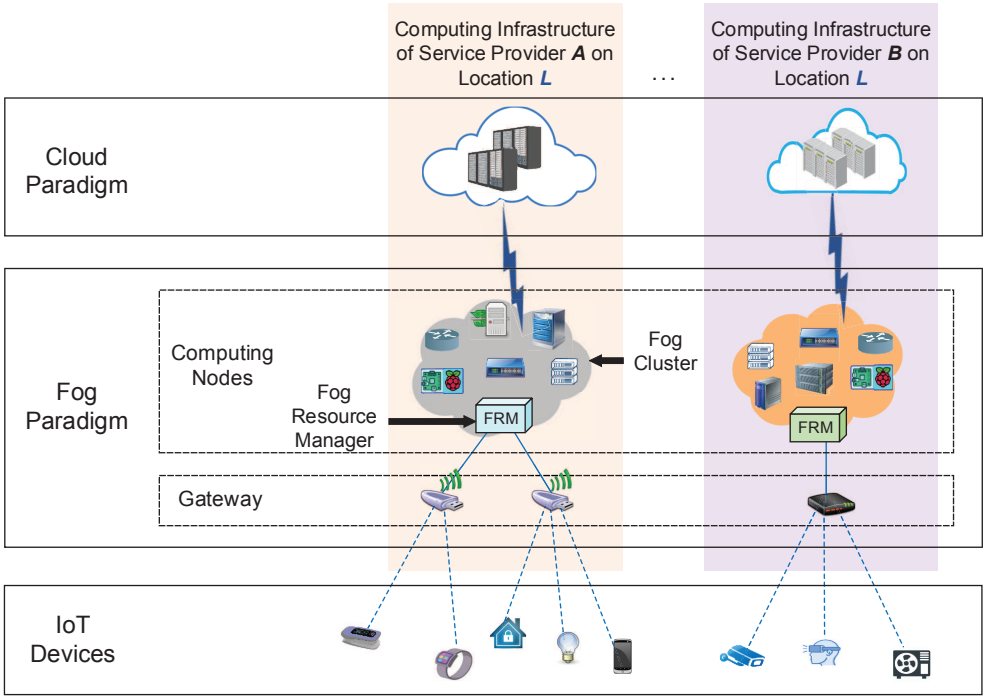

Figure 3.1: Fog environments for Edge affinity-based application management

set according to the resource availability on that FC and the affordability level of users so that it does not violate the capacity and budget constraints. Whenever the number of placement requests surpasses the allowable limit on an FC, the FGs communicate with the FRM of other FCs or remote Cloud datacentres to forward the references of additionally requested applications as per the subscriptions of users. The notations used in this chapter are shown in Table 3.2.

3.3.3 Definition of Edge Affinity

Fig. 3.2 presents the characteristics of different applications in a three-dimensional space of user-defined deadline, amount of data per input and sensing frequency of IoT devices. When the placement request for any application q is received, the values of its character-driving attributes are represented by the FG as a vector ϕ_q. For example, if user-defined deadline $\delta_q = 0.250$ seconds, average amount of data per input $\psi_q = 300$ kilobytes and data sensing frequency of IoT devices $\lambda_q = 7$ input per seconds for application q, its

$\phi_q =< 0.250, 300, 7 >$. Numerical domain and unit of these attributes are different. Therefore, their values are normalized within [0,1] by FGs using Eqs. 3.1, 3.2 and 3.3 in terms of maximum and minimum value for the respective attribute in all placement requests.

$$\overline{\delta_q} = \frac{\delta_q - min(\delta_{\forall q' \in Q})}{max(\delta_{\forall q' \in Q}) - min(\delta_{\forall q' \in Q})} \tag{3.1}$$

$$\overline{\psi_q} = 1 - \frac{\psi_q - min(\psi_{\forall q' \in Q})}{max(\psi_{\forall q' \in Q}) - min(\psi_{\forall q' \in Q})} \tag{3.2}$$

$$\overline{\lambda_q} = 1 - \frac{\lambda_q - min(\lambda_{\forall q' \in Q})}{max(\lambda_{\forall q' \in Q}) - min(\lambda_{\forall q' \in Q})} \tag{3.3}$$

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

However, for a single application q, its $\overline{\delta_q}$ can be closer to 1 whereas value of other two attributes $\overline{\psi_q}$ and $\overline{\lambda_q}$ can be closer to 0. Similarly, for any two applications q and q', $\overline{\lambda_q}$ can be greater than $\overline{\lambda_{q'}}$, although both $\overline{\delta_q}$ and $\overline{\psi_q}$ can be smaller than $\overline{\delta_{q'}}$ and $\overline{\psi_{q'}}$ respectively. These conflicting requirements can resist efficient management of applications in Fog environments. Hence, they need to be handled deliberately while making the management decisions.

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

Table 3.2: Notations for Edge affinity-based management

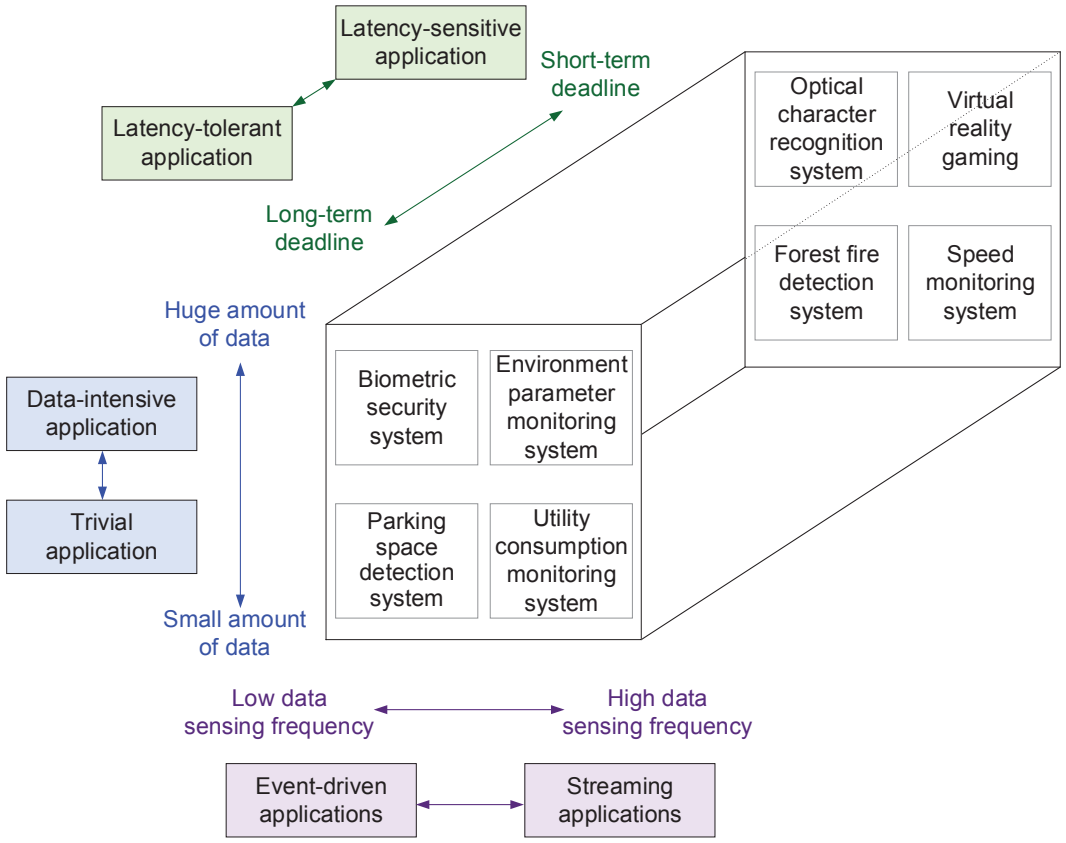

Figure 3.2: Variations of IoT applications

3.4 Edge Affinity-based Application Management

The proposed Edge affinity-based application management policy functions in distributed manner across the gateway and the infrastructure level of Fog environments (Fig. 3.3). It is divided into three phases. At first, FGs classify applications according to their Edge affinity. Later, the allowable number of applications for Fog-based placement are selected. FGs forward the references of selected applications to the FRM of subscribed FCs. Finally, FRMs determine the time-optimized application-instance mapping and assign them accordingly. These phases are described as follows.

3.4.1 Classification of Applications

At any FG g, the proposed policy sorts the requested applications in non-dominated order of their Edge affinity. Non-dominated sorting is applied to identify Pareto optimal solutions for multi-objective optimization problems. It also organizes the solutions in different ranks as per the dominance relationship [214]. Our policy adopts non-dominated sorting to deal with the conflicting cases in Edge affinity of different applications and classify them in numerical order so that their prioritized selection can be made for Fog-based placement. According to the adopted non-dominated sorting approach, an application q dominates another application q' when their Edge affinity η_q and $\eta_{q'}$ respectively meet the following conditions.

1. η_q is not greater than $\eta_{q'}$ for all normalized character-driving attributes.

2. η_q is strictly smaller than $\eta_{q'}$ for at least one normalized character-driving attribute.

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

1. The set of first-order non-dominated applications τ^1 is initialized (line 2). For each application $q \in Q_g$, another set Y_q and a variable v_q are introduced (line 3-5). Y_q refers to the applications dominated by q. On the other hand, v_q counts the number of applications that dominate q. If all normalized character-driving attributes such as $\overline{\delta_q}$, $\overline{\psi_q}$ and $\overline{\lambda_q}$ of application q are not greater than the same attributes of an application $q' \in Q_g$ and one of the attributes is strictly smaller than that of application q', then q' is considered dominated by q. Hence, it is included in Y_q (line 6-8). Conversely, if application q' dominates q, v_q is incremented by 1 (line 9-10). After checking with all $q' \in Q_g$, if v_q still holds the initial value, it signifies application q as non-dominated in respect of the rest. Therefore, application q is added to the set of first-order non-dominated applications τ^1 (line 11-12).

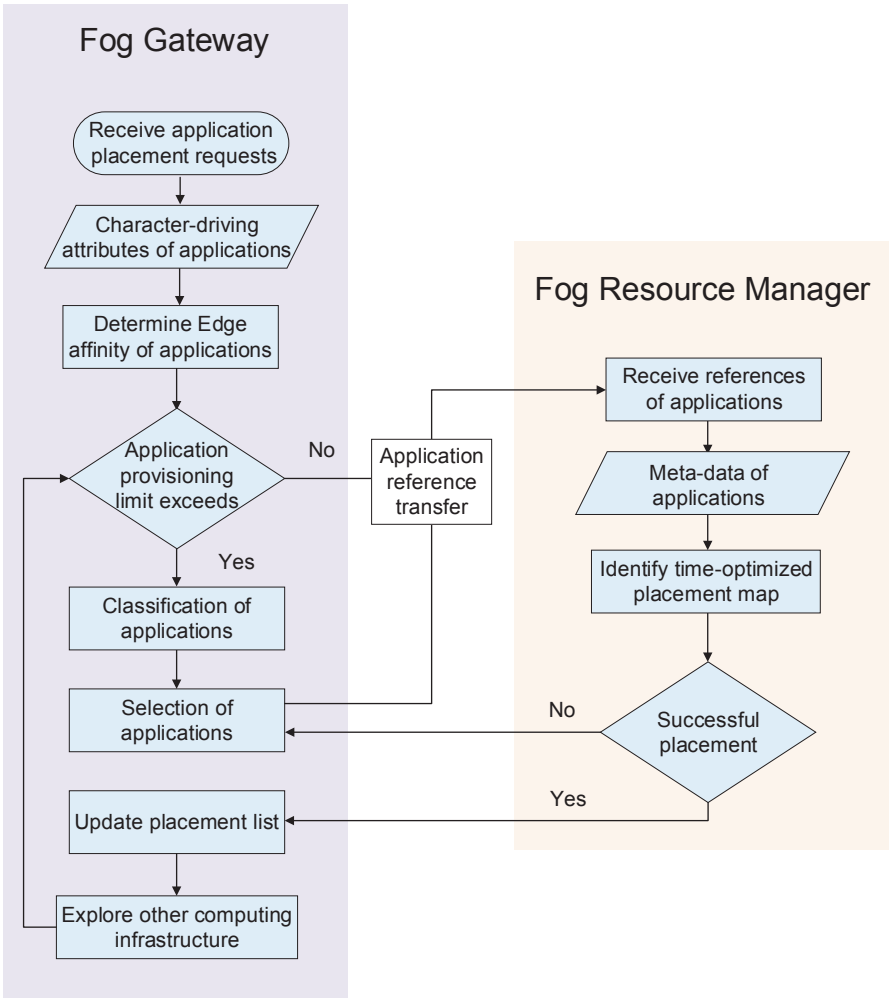

Figure 3.3: Flowchart for proposed Edge affinity-based application management

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

Algorithm 1 Algorithm for classifying applications

1: **procedure** APPLICATIONCLASSIFICATION(Q_g)
2: $\tau^1 \leftarrow \emptyset$
3: **for** $q := Q_g$ **do**
4: $\Upsilon_q \leftarrow \emptyset$
5: $v_q \leftarrow 0$
6: **for** $q' := Q_g$ **do**
7: **if** $\left(\overline{\delta_q} < \overline{\delta_{q'}} \ \& \ \overline{\psi_q} \leq \overline{\psi_{q'}} \ \& \ \overline{\lambda_q} \leq \overline{\lambda_{q'}}\right) \ ||$
$\left(\overline{\delta_q} \leq \overline{\delta_{q'}} \ \& \ \overline{\psi_q} < \overline{\psi_{q'}} \ \& \ \overline{\lambda_q} \leq \overline{\lambda_{q'}}\right) \ ||$
$\left(\overline{\delta_q} \leq \overline{\delta_{q'}} \ \& \ \overline{\psi_q} \leq \overline{\psi_{q'}} \ \& \ \overline{\lambda_q} < \overline{\lambda_{q'}}\right)$ **then**
8: $\Upsilon_q \leftarrow \Upsilon_q \cup q'$
9: **else if** $\left(\overline{\delta_{q'}} < \overline{\delta_q} \ \& \ \overline{\psi_{q'}} \leq \overline{\psi_q} \ \& \ \overline{\lambda_{q'}} \leq \overline{\lambda_q}\right) \ ||$
$\left(\overline{\delta_{q'}} \leq \overline{\delta_q} \ \& \ \overline{\psi_{q'}} < \overline{\psi_q} \ \& \ \overline{\lambda_{q'}} \leq \overline{\lambda_q}\right) \ ||$
$\left(\overline{\delta_{q'}} \leq \overline{\delta_q} \ \& \ \overline{\psi_{q'}} \leq \overline{\psi_q} \ \& \ \overline{\lambda_{q'}} < \overline{\lambda_q}\right)$ **then**
10: $v_q \leftarrow v_q + 1$
11: **if** $v_q = 0$ **then**
12: $\tau^1 \leftarrow \tau^1 \cup q$
13: $i \leftarrow 1$
14: **while** $\tau^i \neq \emptyset$ **do**
15: $\tau^{i+1} \leftarrow \emptyset$
16: **for** $q := \tau^i$ **do**
17: **for** $q' := \Upsilon_q$ **do**
18: $v_{q'} \leftarrow v_{q'} - 1$
19: **if** $v_{q'} = 0$ **then**
20: $\tau^{i+1} \leftarrow \tau^{i+1} \cup q'$
21: $i \leftarrow i + 1$

i is incremented by 1 so that the set of following non-dominated ordered applications can be traversed in similar way (line 21).

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

3.4.2 Selection of Applications

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

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

2. For each application $q \in \tau^i$, value of its bottleneck character-driving attribute ϑ_q is identified (line 14). For example, if $\overline{\delta_q} = 0.10$, $\overline{\psi_q} = 0.15$ and $\overline{\lambda_q} = 0.20$ for application

Algorithm 2 Algorithm for application selection

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

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

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

3.4.3 Placement of Applications

Each FG $g \in G$ interacting with an FC c forwards a reference list of selected applications χ_{cg} to the corresponding FRM. The FRM accumulates the received application lists in Γ using Eq. 3.4. Thus, Γ refers to the set of all applications selected for placement on FC c.

$$\Gamma = \bigcup_{\forall g \in G} \chi_{cg} \tag{3.4}$$

In FC c, before placing an application $q \in \Gamma$ on an instance $p \in P$, FRM calculates the input propagation time t^{ι}_{pq}, execution time t^{e}_{pq} and output transfer time t^{o}_{pq} of q on that instance using Eqs. 3.5, 3.6 and 3.7 respectively. They explicitly depend on the downlink speed Φ_p, processing speed Λ_p and uplink speed Ψ_p of instance p, and the average input data size ψ_q, number of instruction μ_q and output data size σ_q of application q. Based on them, the expected service delivery time t_{pq} of q on instance p is also determined using

Eq. 3.8.

$$t^{\iota}_{pq} = \frac{\psi_q}{\Phi_p} \tag{3.5}$$

$$t^{e}_{pq} = \frac{\mu_q}{\Lambda_p} \tag{3.6}$$

$$t^{o}_{pq} = \frac{\sigma_q}{\Psi_p} \tag{3.7}$$

$$t_{pq} = t^{\iota}_{pq} + t^{e}_{pq} + t^{o}_{pq} \tag{3.8}$$

An FRM aims to place an application on that instance which minimizes its service delivery time. For the set of all selected applications Γ, this objective is formulated using a constrained Integer Linear Program (ILP) model as shown in Eq. 3.9. Solution of the ILP model is defined by a binary decision variable x_{pq} that becomes 1 if application q is mapped to instance p and 0 otherwise. Constraints of the ILP model ensure that an application will not be placed to multiple instances (Eq. 3.10), its service delivery time will meet the deadline (Eq. 3.11) and its host instance will have sufficient resources to meet its minimum requirements (Eq. 3.12).

$$\min \sum_{q \in \Gamma} x_{pq} t_{pq} \tag{3.9}$$

subject to,

$$x_{pq} \leq 1; \forall q \in \Gamma \tag{3.10}$$

$$t_{pq} \leq \delta_q; \forall q \in \Gamma \tag{3.11}$$

$$\omega^{r}_{q} \leq \Omega^{r}_{p}; \forall q \in \Gamma, \forall r \in R \tag{3.12}$$

The optimization problem in Eq. 3.9 deals with fixed number of applications and instances. They are set according to the resource availability in an FC and the capacity of FRM in solving the problem within acceptable time limit using any ILP solvers like Solving Constraint Integer Programs (SCIP) [215]. However, if an application misses placement for the constraints, another application is selected by the FG using Algorithm 2.

3.5 Performance Evaluation

The performance of the proposed policy is evaluated in both real-world and simulated Fog environments. It is also compared with several existing application management policies. Among them, the *Time-aware* management policy [76] optimizes application service time in respect of user's budget. The *Resource-aware* management policy [204] reduces the scope of resource over provisioning while placing applications on Fog instances and meets their minimum requirements. The *Workload-aware* management policy [205] schedules less compute-intensive applications with high bandwidth requirements in Fog infrastructure as per their deadline constraints. We have implemented the basic concepts of these policies separately. Moreover, we use different sets of performance metrics for both experimental setup so that the efficacy of the policies can be analyzed from diverse perspectives. Details of the experiment environments, performance metrics and results are discussed below.

3.5.1 Experiments in a Real Environment

Fig. 3.4 presents a sample setup of the real Fog environment. We organize the environemnt using FogBus framework [24]. FogBus helps to integrate IoT devices and Fog infrastructure through a dedicated software system and supports the creation of scalable Fog environments. In our real experimental setup, eight different smart phones act as IoT devices. They are connected with an *AMD Dual-Core M320 2.10 GHz 2.00 GB RAM* configured computer which is regarded as an FG. The FG communicates with a cluster of computers that plays the role of FC. Within the cluster, there exists two *Intel Core i7-6700T 2.80 GHz 16.00 GB RAM* and three *Intel Core i7-7700T 3.80 GHz 16.00 GB RAM* configured computers acting as Fog nodes along with an *Intel Core i3-2350M 2.30 GHz 4.00 GB RAM* configured computer performing the duty of FRM. The Fog nodes are capable of hosting twelve different Fog instances through VirtualBox [216]. The instances adapt the bridged networking mode so that they can be accessed by all components within the Local Area Network (LAN). Using NetLimiter [217] software the uplink and downlink speed within Fog infrastructure are controlled and its resource utilization is monitored by Process Explorer [218] software.

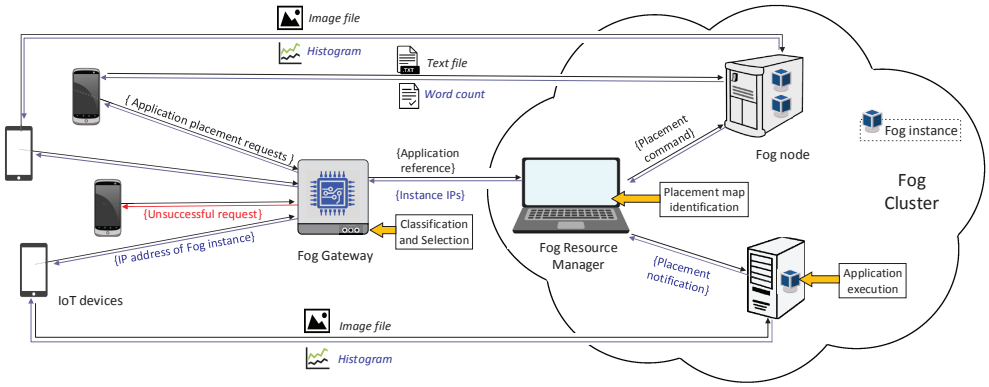

Figure 3.4: Real experimental setup for Edge affinity-based application management

Moreover, we profile the execution time of two applications in this environment. One of the applications analyses histogram of an image file whereas another counts the number of words in a text file. We define three different file sizes for their inputs. Each smart phone launches placement requests to the FG for placing these applications in Fog infrastructure with inputs having any of the defined file sizes. Besides, a placement request denotes the data sensing frequency and expected application service delivery time of the associated smart phone. Since application service requirements vary from one request to another, we treat each request as the demand for a separate application. We also enforce the FG to provision at most ten such applications in the Fog infrastructure. Different settings of this environment are listed in Table 3.3.

Performance Metrics

The following metrics are used to evaluate the proposed policy in this experimental setup.

- Average Amount of Data Handled (Avg. ADH): If an application management policy utilizes the Fog infrastructure extensively, value of this metric increases. It also denotes the lower amount of load sent to other computing infrastructure.
- Average Management Load (Avg. ML): It denotes the average CPU usage of FG and FRM while classifying, selecting and identifying application-instance map. The balanced Avg. ML between FG and FRM reflects the efficacy of a policy in distributing the management tasks across the gateway and the infrastructure level.

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

Table 3.3: Settings of real Fog environment for Edge affinity-based management

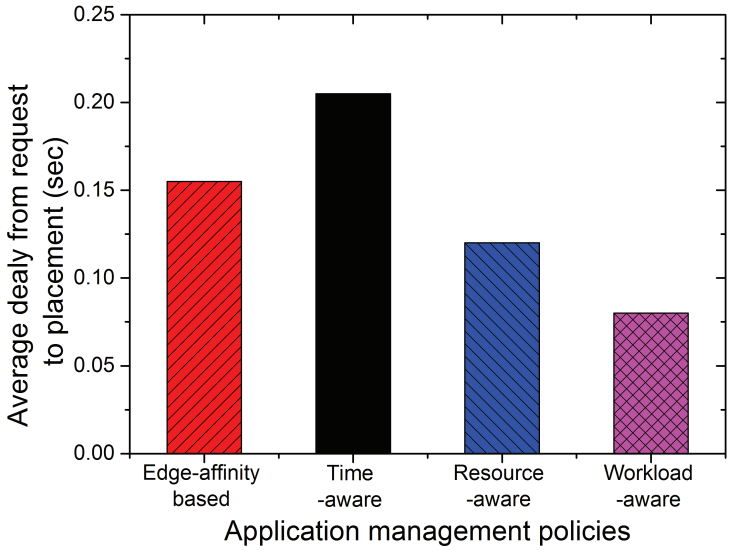

Figure 3.5: Average delay from request to placement for different management policies

•Average Delay from Request to Placement (Avg. ADRP): Lower value of this metric points to the enhanced performance of a policy in reducing waiting of IoT devices while accessing Fog infrastructure services and initiating data processing.

Result Analysis

The Time-aware policy applies evolutionary algorithm to determine application-instance map. Compared to Time-aware policy, our policy performs better in improving Avg. ADRP as it conducts low complexity approaches to classify and select applications for Fog-based placement and reduces the dimension of optimization problem. As the Workload and the Resource-aware policy adapt simplified earliest deadline first and earliest completion time first, and conduct multi-phase sorting and searching for placement map identification respectively, they perform well in terms of Avg. ADRP than all others (Fig. 3.5)

As the proposed policy explicitly prioritizes applications for Fog-based placement according to their input data size, it increases Avg. ADH in Fog infrastructure (Fig. 3.6). By reducing the scope of resource over-provisioning, the Resource-aware policy also improves Avg. ADH compared to the rest. However, for enhancing application service

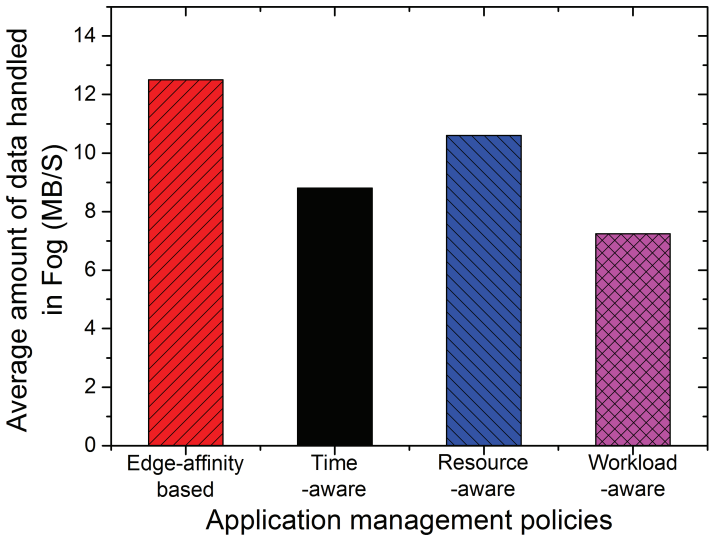

Figure 3.6: Average amount of data handled by different management policies

delivery time and deadline-prioritized placement, the Time and the Workload-aware policy often places applications having small amount data on powerful computing instances. As a result, they fail to increase Avg. ADH like other policies.

In addition, to illustrate the efficacy of our policy in distributing application management tasks, we compare its performance with two more variations namely *Infrastructure only* and *Gateway only*. In Infrastructure only, all management tasks are executed by the FRM whereas in Gateway only, the opposite happens. Nevertheless, our policy facilitates balanced Avg. ML on both gateway and infrastructure while conducting application management tasks (Fig. 3.7). Hence, it neither increases computational burden on resource poor FGs like Gateway only approach nor overwhelms the FRMs with additional responsibilities as Infrastructure only approach.

3.5.2 Experiment in a Simulated Environment

Besides the real setup, several experiments are also conducted in iFogSim-simulated [219] Fog environment so that we can demonstrate the large-scale comparisons between ours and the other application management policies easily. Since the practical workload is not available for simulating different scenarios in Fog, we model a synthetic workload for the experiments. Its parameters are listed in Table 3.4 and they are aligned with

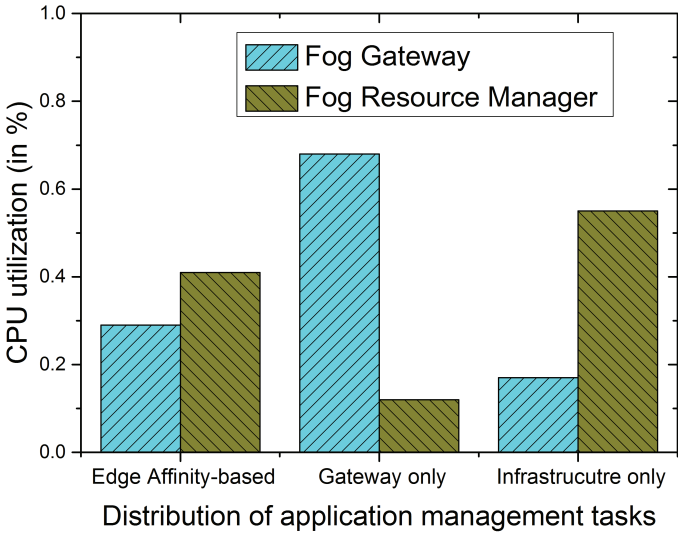

Figure 3.7: Average management load in different management setup

the settings of real Fog environment discussed in Section 3.5.1. Here, the arrival rate of placement requests for applications and numerical value of their character-driving attributes follow the Poison distribution. Furthermore, there exists a linear relationship between the number of instructions of an application and its input data size. In the simulated setup, if an application is not selected for Fog-based placement, it is forwarded to a Cloud datacentre for execution. The simulation experiments are conducted on an *Intel Core 2 Duo CPU @ 2.33-GHz 2GB-RAM* configured computer and the Fog environment is considered virtualized.

Performance Metrics

The following metrics are used in the simulation experiments:

• Percentage of QoS Satisfied Applications (Per. QSA): Increased value of this metric refers to the enhanced performance of management policies in meeting application service delivery deadline. This metric explicitly depends on the communication and computation latency of an application. If Y and Z denote the set of deadline satisfied and the set of placed applications in both Fog and Cloud instances respectively, Per.

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

Table 3.4: Simulation parameters for Edge affinity-based management

QSA is calculated using Eq. 3.13:

$$Per.\ QSA = \frac{|Y|}{|Z|} \times 100\% \tag{3.13}$$

• Average Network Relaxation Time (Avg. NRT): Increased value of this metric signifies reduced communication overhead among the instances that consequently decreases the possibility of network congestion. If ζ_p is the set of all placed applications on instance p during the simulation round, Avg. NRT is referred by Eq. 3.14:

$$Avg.\ NRT = \frac{1}{|P|} \sum_{\forall p \in P} \frac{\sum\limits_{\forall q \in \zeta_p} \frac{1}{\lambda_q} - t^{\iota}_{pq}}{|\zeta_p|} \tag{3.14}$$

• Average Resource Utilization Ratio (Avg. RUR) of Fog instances: Higher value of this metric denotes improved performance of a placement policy in increasing resource utilization of Fog instances. If F is the set of all Fog instances ($F \subset P$), Avg. RUR is determined through Eq. 3.15:

$$Avg.\ RUR = \frac{1}{|F|} \sum_{\forall p \in F} \frac{\sum_{\forall q \in \zeta_p} \frac{\lambda_q \times \mu_q}{\Lambda_p}}{|\zeta_p|} \tag{3.15}$$

Result Analysis

In this chapter, the results of simulation experiments are analysed in two phases.

• *Impact of Varying Number of Placed Applications:* The Workload-aware management policy mainly focuses on delivering application services within the deadline. Therefore, for increased number of placed applications, it performs better in terms of Per. QSA than the proposed policy. However, our policy not only considers application deadline but also exploits their input size and sensing frequency of IoT devices during application placement (Fig.3.8). Conversely, the Time-aware policy optimizes service time for all applications regardless their deadline criticality and the Resource-aware policy targets to meet the minimum resource requirements of applications without explicitly prioritizing them. Hence, with the increasing number of placed applications, these policies fail to achieve the same level of Per. QSA as the proposed policy.

Furthermore, our policy places applications having high frequency of IoT devices and larger data size in Fog instances. Thus, it reduces the overhead of distant communication even when the number of placed applications in computing environments is increasing. Consequently, it helps to offer improved Avg. NRT than others (Fig. 3.9). Moreover, due to exploiting Fog instances with lower possibility of resource over provisioning and facilitating the applications having high bandwidth requirements, the Resource and the Workload-aware policy perform nearly as the proposed policy. On the other hand, the Time-aware policy fails to improve Avg. NRT like others since it barely considers the data flow characteristics of applications while placing them in Fog infrastructure.

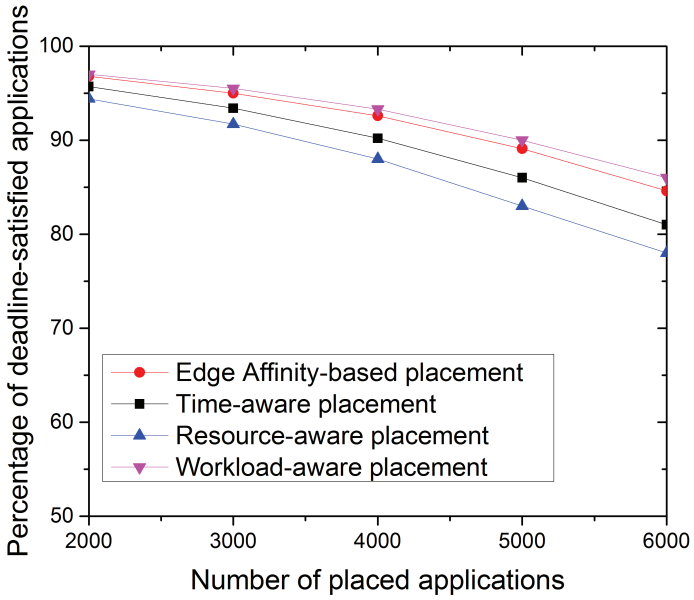

Figure 3.8: Percentage of QoS satisfaction for varying number of placed applications

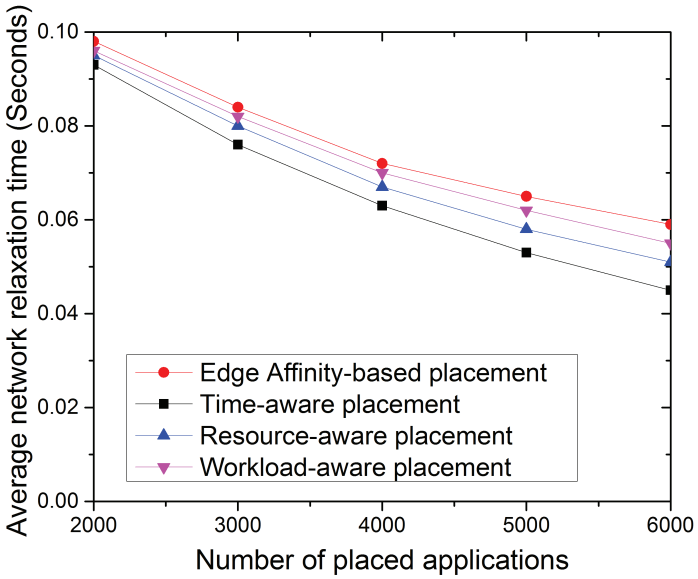

Figure 3.9: Average network relaxation for varying number of placed applications

Moreover, the huge amount of data handled by our policy helps to increase Avg. RUR of Fog instances as the number of placed applications increases. It also works in favour of the Resource-aware policy (Fig. 3.10). However, for executing less compute intensive applications in Fog environments and optimizing application service time with-

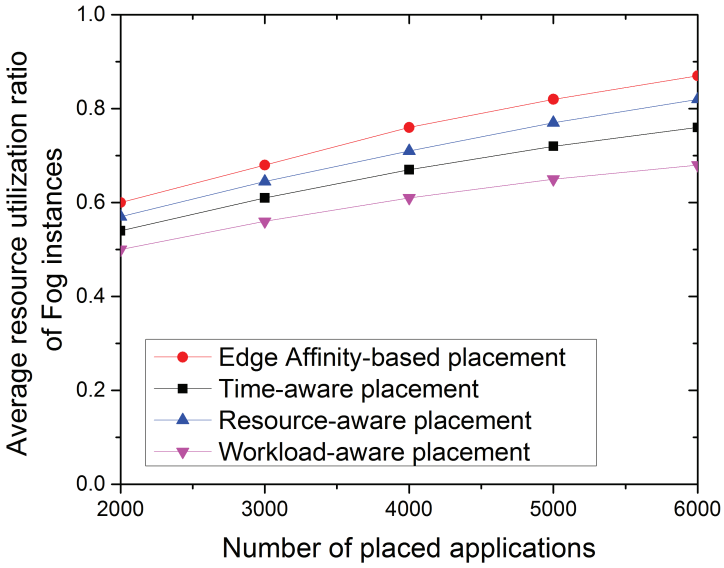

Figure 3.10: Average resource utilization for varying number of placed applications

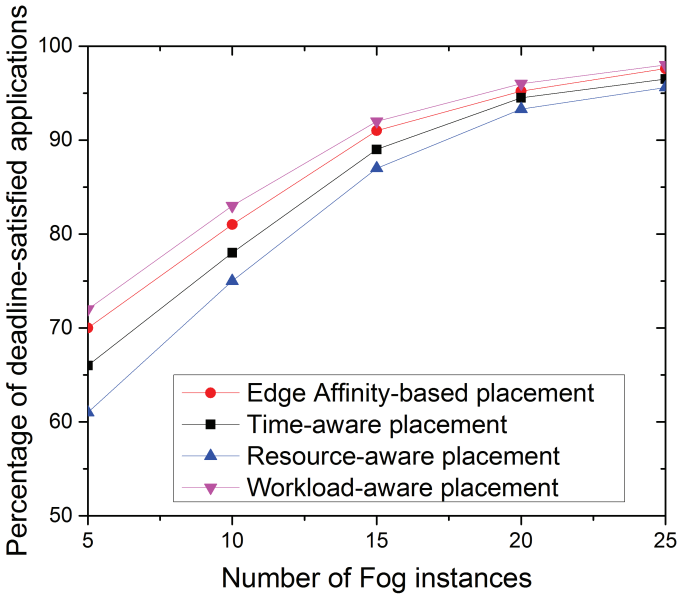

Figure 3.11: Percentage of QoS satisfaction for varying number of Fog instances

out setting any precedence, the Time and the Workload-aware policy often fail to exploit the Fog instances comprehensively. As a result, Avg. RUR degrades for these policies compared to others.

- *Impact of Varying Number of Fog Instances:* As the number of Fog instances increases,

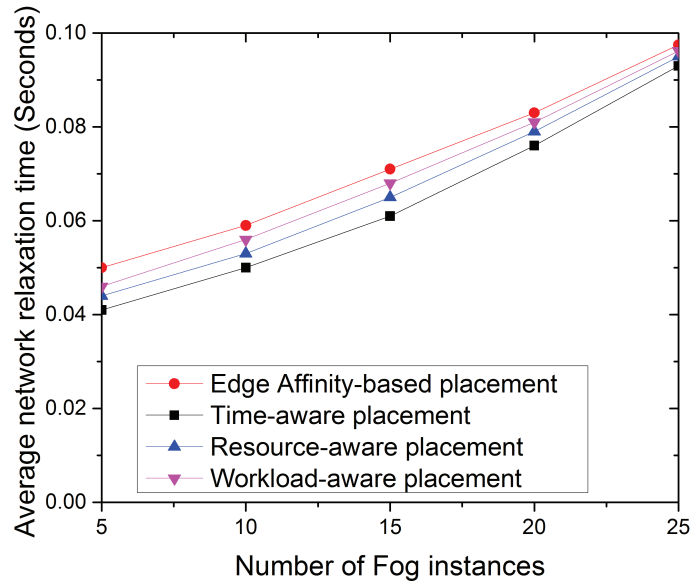

Figure 3.12: Average network relaxation for varying number of Fog instances

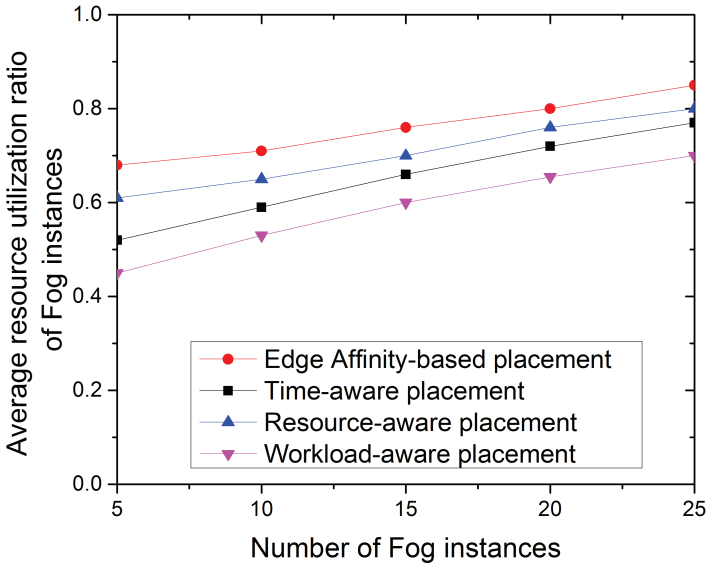

Figure 3.13: Average resource utilization for varying number of Fog instances

the scope of placing applications in proximity of data sources expands. It reduces the data propagation delay for a large portion of applications and increases Per. QSA for all application management policies. However, due to prioritizing applications based on their deadline constraints, the proposed and the Workload-aware application manage-

ment policy performs better in this case compared to the other policies (Fig. 3.11).

Additionally, the increased number of Fog instances resists the transfer of huge amount of data to other infrastructure. Although it elevates the data exchange rate at the edge of network, it is still trivial compared to the reduction in distant communication overhead. Hence, Avg. NRT increases for all policies (Fig. 3.12). Since our policy and the Workload-aware policy explicitly handle the data flow and bandwidth issues of applications, they perform better than others in improving Avg. NRT with the increment of Fog instances.

Furthermore, the decreased idle time of instances refers to their higher utilization rate. The proposed and the Resource-aware policy engage the increased number Fog instances in executing the applications having high data load and stringent resource requirements. It helps both the policies to reduce the idle of the instances significantly compared to others. As a result, the Avg. RUR improves for these policies (Fig. 3.13).

3.6 Summary

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

Chapter 4

Latency-aware Application Management

To fully leverage the capabilities of the Fog nodes, large scale applications that are decomposed into inter-dependent Application Modules, can be deployed orderly over the nodes based on their latency sensitivity. In this chapter, we propose a latency-aware Application Module management policy for Fog environment that meets the diverse service delivery latency and amount of data signals to be processed in per unit time for different applications. The policy aims to ensure applications Quality of Service (QoS) in satisfying service delivery deadline and optimize resource usage in Fog environment. We model and evaluate our proposed policy in iFogSim-simulated Fog environment. Results of the simulation studies demonstrate significant improvement in performance over alternative latency-aware strategies.

4.1 Introduction

Typically, different IoT applications carry out some common activities such as receiving data from IoT devices, pre-processing and analysis of the received data, handling event of interests, etc. [1]. An Application Module contains necessary instructions so that one of these aspects for the respective application can be attained. For a given input, an Application Module performs some specific operations to generate corresponding output. Later, based on data-dependency, the output is sent to another module as input. In order

This chapter is derived from:

- **Redowan Mahmud**, Kotagiri Ramamohanarao, and Rajkumar Buyya, "Latency-aware Application Module Management for Fog Computing Environments", *ACM Transactions on Internet Technology (TOIT)*, Volume 19, No. 1, Article 9, Pages: 1-21, ISSN:1533-5399, ACM Press, New York, USA, January 2019.

to process input within a fixed timeframe, each module requires a certain amount of resources, e.g. CPU, memory, bandwidth, etc. Hence, Application Modules together with their allocated resources constitute the data processing elements for different applications. This sort of decomposition is effective for distributed development of large scale applications. In literature, similar concept is used to divide Component-based applications into multiple Application Components [220]. However, while executing applications in distributed manner, latency related issues such as node to node communication delay, application service delivery deadline and service access frequency often become predominant and influence Quality of Services (QoS) and resources utilization. In different computing paradigms although various latency-aware management strategies for distributed applications are proposed [221] [222] [223][224], the aforementioned latency issues have not been addressed simultaneously. Besides, due to dependency towards centralized management and lack of latency-sensitive application prioritization, the existing policies often get interrupted to meet the challenges of IoT-enabled real-time interactions [225]. Therefore, in this chapter, we propose a latency-aware Application Module management policy for Fog computing environment that considers different latency aspects of distributed applications in a body with decentralized co-ordination. The objective of the policy is to manage latency-sensitive and latency-tolerant IoT-applications in different ways so that deadline driven QoS provision can be ensured for all types of applications while optimizing resources in Fog computing. The main **contributions** of this work are:

- A latency-aware approach for placing Application Modules on distributed Fog nodes that ensures deadline satisfied service delivery for different types of applications.
- Explored latency-aware Application modules forwarding strategy that re-locates modules in order to optimize number of computationally active Fog nodes.
- Our proposed latency-aware policy is applied in iFogSim-simulated Fog environment and compared with other latency-aware policies from different perspectives. The performance results show significant improvement in favor of our policy.

The rest of the chapter is organized as follows. In Section 4.2, we highlight several

related works. In Section 4.3, different event-driven IoT-application scenarios are discussed with a general application model. Section 4.4 provides the system overview. The proposed latency-aware Application Module management policy is presented in Section 4.5. Section 4.6 reflects the simulation environment and the performance evaluation. Finally, Section 4.7 concludes the chapter.

4.2 Related Work

Famaey et al. [226] proposed a dynamic and latency-aware distributed service placement policy over multiple homogeneous servers. Their policy assigns services with a utility-driven priority value. A server applies the policy to determine whether it can execute a request with the service placed within it or should forward the request to another service hosted in the nearby servers. While forwarding the service, server to server latency is taken into account. The policy shuffles active services within a server so that resources can be accommodated to execute the newly requested services and the service forwarding latency can be mitigated.

Kang et al. [221] proposed an Iterative Sequential co-deployment algorithm for distributed services. Based on user's proximity, their algorithm at first generates some virtual random placement for the services, and then performs iterative re-placement to improve user's service access and inter-service (nodal) communication time. The algorithm deals with both latency-sensitive and latency-tolerant services in the same way. After deployment of services, to handle dynamic changes of users and their access pattern, authors recommend to rerun the deployment process periodically.

Takouna et al. [224] indicated towards communication latency and energy-aware placement of parallel applications in virtualized datacentres. Their policy dynamically identifies the bandwidth demand and communication characteristics of the distributed applications and re-allocates the applications through migration if the current placement fails to handle the issues. A migration manager supervises the operations at the time of migration. The migration manager sorts the virtualized instances based on their current traffic and selects an instance as migration target by checking the resource availability.

Gupta et al. [227] proposed a transfer-time aware workflow scheduling policy for

multi-Cloud environment. It prioritizes inter-dependent tasks for scheduling to the multi-Clouds based on the computation cost and communication time. Within the clouds, the tasks are mapped considering earliest start and finish time. The policy aims to enhance service delivery time. The operations are handled by a global Cloud manager.

Fan et al. [228] discussed data placement in geo-distributed multi-Clouds. Their placement policy sorts data chunks in non-ascending order according to the collective access probability from all the users and merge them into larger data segments based on the capacity of the servers. Later it iteratively searches for appropriate servers where placement cost of such data segment gets reduced. The placement cost is calculated centrally considering service access latency, energy consumption of the servers and the network. The algorithm also iteratively turns-off of the free servers.

Based on the requirements during development, deployment and management of component-based IoT applications, Yangui et al. [220] proposed a Platform as a Service (PaaS) architecture to facilitate application provisioning in hybrid Cloud-Fog environment. Being a centralized coordinator, the PaaS architecture allows developing applications according to the target domain; discovering, initiating, configuring and scaling resources for deploying and executing the application components; managing execution flow between of the components; monitoring SLA and component migration; providing resource and component management interfaces. In evaluation, different component placement scenarios are discussed based on the end-to-end latency.

Taneja et al. [204] presented a Module Mapping Algorithm to place distributed applications in Cloud-Fog environment. It aims to ensure proper resource utilization. The algorithm is aware of the network issues. It sorts both the nodes and application modules according to the available capacity and requirements and maps the modules when the constraint satisfies. In one sense, it prioritizes the modules based on the resource expectation. The policy reflects the way to reduce resource under-utilization for distributed IoT application. It also highlights how Fog-Cloud inter-operation can minimize end-to-end latency compared to the Cloud-based approaches.

Ottenwalder et al. [223] proposed a plan-based operator placement and migration policy for Mobile Complex Event Processing (MigCEP) applications. It supports mobility of the users and creates time-graph model to identify possible migration targets.

Considering the shortest path from data source, it selects the appropriate target instance from the time graph model. On the selected instance, the policy applies coordination to accommodate the migrating operator. The main intentions of the proposed policy are to reduce network overhead and end-to-end delay.

Nishio et al. [222] discussed latency-aware application deployment for Mobile Cloud Computing (MCC). In their proposed system, a coordinator manages all incoming requests and resources in order to meet service latency for different applications. While sharing resources, smaller amount of tasks are forwarded from one node to another node under supervision of the coordinator. Policies running in the system trade-offs utility gain and energy consumption for resource optimization.

A cost optimized offline and online service placement policy for Mobile Micro Cloud (MMC) is discussed in [229]. The policy determines an optimal configuration of service instances that minimize the average cost over time. A centralized controller predicts the cost and computes the service instance configuration for the next time slots. The cost function can include resource consumption, service access latency and other monetary issues. The policy supports mobility of the entities and migration of the services accordingly. The authors also consider the errors while predicting the cost and develop a method to identify the optimal look-ahead window size.

Chamola et al. [230] considered Software Defined Network (SDN) enabled communication of multiple Cloudlets to place services at the proximity of the mobile users. The task assignment solution can improve the QoS in respect of service delivery and service access time. According to the proposed policy, if a Cloudlet gets overloaded, the tasks offloaded to it, are processed on another relaxed Cloudlet of the network. Necessary operations to conduct the policy are supervised by a central Cloudlet manager.

To facilitates latency-aware scheduling of applications in virtual machines, Xu et al. [231] introduced a scheduler named vSlicer. vSlicer nurture the concept of *differentiated-frequency microslicing*. Unlike, traditional scheduler, vSlicer divides a CPU slice into many microslices and according to microslices, it schedules applications in higher frequency. By doing so it increases CPU access probability of applications.

Table 4.1 provides a brief summary of state-of-the-art for latency-aware application or service placement in distributed servers, multi-Cloud, Mobile-Cloud and Cloud-Fog

Work	Distributed application	Meets latency			Forwards application	Optimizes resources	Decentralized management	Prioritized placement
		Service access	Service delivery	Inter nodal				
[226]	✓			✓	✓		✓	✓
[221]	✓	✓		✓			✓	
[224]	✓			✓	✓			✓
[227]	✓		✓	✓				✓
[228]	✓	✓				✓		✓
[220]	✓		✓		✓	✓		
[204]	✓		✓			✓		✓
[223]	✓		✓		✓		✓	
[222]	✓		✓		✓	✓		
[229]		✓			✓	✓		
[230]		✓	✓		✓			
Latency-aware (This work)	✓	✓	✓	✓	✓	✓	✓	✓

Table 4.1: Summary of related work for latency-aware management

environment. Compared to the existing works, the unique aspect of our work is that we have considered service access delay, service delivery time and inter-nodal communication delay simultaneously while placing and forwarding inter-dependent application modules over distributed Fog nodes. Besides, the proposed policy decentrally coordinates the placement and forwarding operation to overcome the constraints of centralized supervision for example; application management overhead, single point of failure, additional communication and decision making delay etc. Our policy can place the modules both horizontally and vertically and in order to facilitate low energy usage, can optimize resources by forwarding all the modules from one node to the others. Moreover, it enhances priority of latency-sensitive applications to place closer to the data source by deploying latency-tolerant applications in the upper level Fog nodes. Simulation results support the applicability of our policy in terms of QoS satisfaction, resource optimization, module placement and forwarding time.

4.3 Application Scenarios

4.3.1 IoT-enabled Systems

In advanced healthcare and smart home based system, the structure of different IoT applications reflects through some common operations. Two such application scenarios and their basic operations on the received data can be described as follows.

Patient Respiratory Monitoring System

In order to monitor breathing and oxygen level of asthma patients, Pulse Oximeters are widely used at the hospitals. Usually Pulse Oximeters are connected to the bed-side monitors, and continuously shows oxygen level carried in the body, heart beat rate and changes in blood volume of the skin [232]. The bed-side monitors provide the interface for authentication, aggregate data signals and usually forward the sensed data to Cloud or other computational entities for further processing to detect Hypoxemia, Hypercapnia and sleep apnea of the patients. Since, some Pulse Oximeter generated data signals can be irrelevant, incomplete and diverse in format; data filtering techniques are applied in this context. Later, different data analytics in respect of Hypoxemia, Hypercapnia, etc. operate on the filtered data. Sometimes, the analysed outcome can indicate to an emergency situation. Based on the outcome, required actions for example ventilation, injection, medication, etc. are triggered at the patient's bed-side actuators. For critical asthma patients, corresponding application has to perform the aforementioned operations in real-time that Cloud-based placement of the application often fails to deal with. In addition, placement of such large-scale applications in distributed and heterogeneous Fog nodes is not as simple as the Cloud-based placement.

Visitor Identification System

To identify a visitor in smart-home based system, usually entrance-side cameras take the pictures of the visitors and send to Cloud or other computing entities for image processing [233]. Sometimes due to weather conditions and other external effects, the taken

pictures get a lots of noise. In this context, image filtration techniques are required to apply for selecting the most appropriate picture and reducing its noises. Image analytics in respect of face and gesture recognition, object detection, etc. are also applied to the filtered pictures for identifying the visitor and the hand held objects. Once the visitor is identified and the hand held objects are found allowable, necessary information is parsed from the respective databases. The information can include contact number, address and access rights of the visitor. If the visitor is authorized to enter the house, entrance-side actuators open the door otherwise create notification to the residents containing the details of the visitor. During urgent period, corresponding application of Visitor Identification System is required to coordinate the aforementioned operations within a reasonable time that may not be possible if the application is placed in distant Cloud. Besides, necessary resources to execute this kind of compute intensive applications in resource constrained Fog nodes are often difficult to manage.

4.3.2 Application Model

Based on the aforementioned scenarios and data-operations, we have considered the following application model for the associate event-driven applications. We assume that each application is composed of *Client Module* (provides initial application interface), *Data Filtering Module* (applies data filtering techniques), *Data Analysing Module* (executes data analytics) and *Event Handler Module* (generates appropriate response to the event). Data dependency exists among the modules of same application which can be expressed through a sequential unidirectional dataflow as shown in Fig. 4.1. After placement of an Application Module, from the respective dataflow the next module is identified for placement. To foster concurrency, the modules can be replicated. Besides, if a module is allocated resources according to its requirements, it is expected that the module will execute its operation within a fixed time.

Client Module is the entrance module for each application. Application initializing information such as authentication, data signal sensing frequency, service delivery deadline, meta-data of subsequent modules and their inter-dependency are notified to the system through the Client Module. After deployment of all modules, Client Module

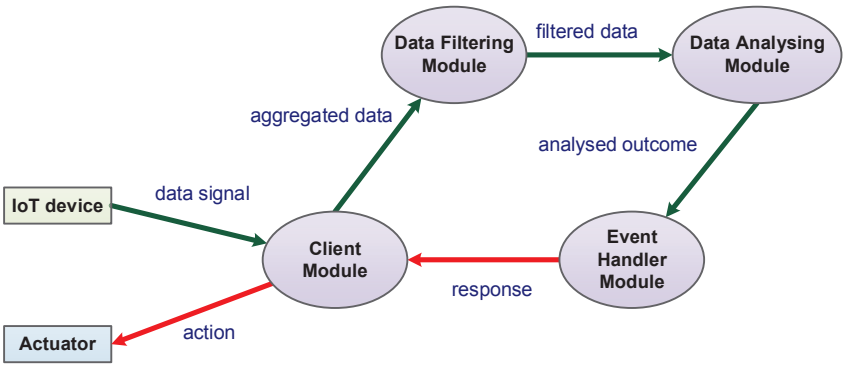

Figure 4.1: Dataflow model for a distributed application

of a particular application directly communicates with respective IoT devices to grab the data signals and forwards in form of aggregated data to the subsequent modules for further operations. If the ultimate analysed outcome of a forwarded data signal invokes any event of interest, Event Handler Module sends corresponding response towards the Client Module. This response is eventually considered as the final application service for an IoT-data signal. Based on the response, Client Module triggers action in the actuators.

Since Client Module plays the role of root module for the applications and closely associates with the IoT devices and the actuators, this particular module for every application is expected to be placed at the proximity of the users.

4.4 System Model and Assumptions

The detailed description of the system model and the associated assumptions are discussed in the following subsections.

4.4.1 Organization of Fog Layer

In this work, Cloud is considered as a standalone computational platform and IoT devices only generate data signals without further processing due to resource and energy constraints. In this circumstance, Fog computing acts as an intermediate layer in between Cloud and IoT devices. Within this layer, nodes are organized in a hierarchical order as shown in Fig. 4.2.

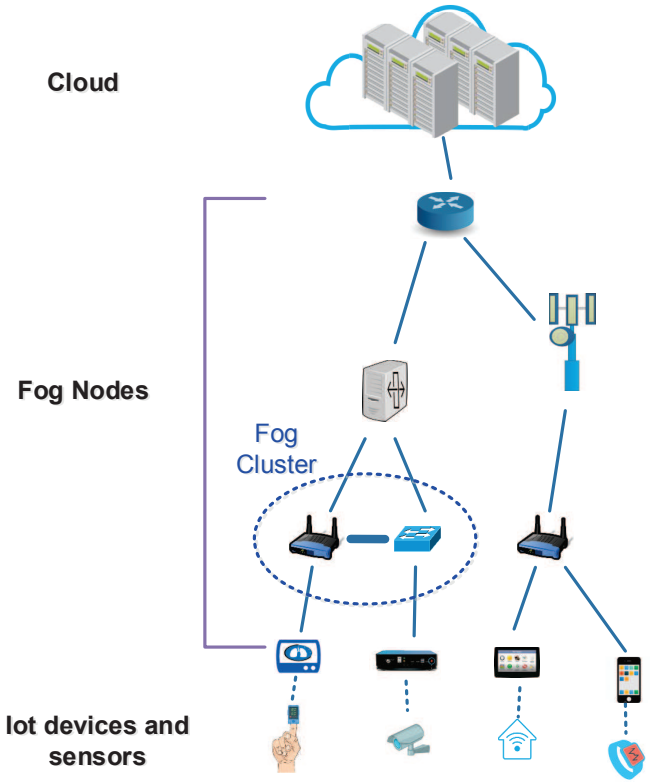

Figure 4.2: Fog environments for latency-aware application management

Lower level Fog nodes are closer to the IoT devices. As the level number goes higher, distance of Fog nodes from IoT devices increases which can be reflected in lower level to higher level uplink latency and delay in service delivery. Compute, storage and networking capabilities of lower level Fog nodes are less compared to that of higher level nodes [234]. Each node in a particular level is directly associated with a node of immediate upper level. Fog nodes can form clusters among themselves and rapidly communicate with each other through Constrained Application Protocol (CoAP) or Simple Network Management Protocol (SNMP). [235]. Therefore, maximum nodal communication delay ϵ_C within a Fog cluster C is negligible and does not impact on service delivery time extensively. We assume that at lower Fog levels, if two nodes from the same level are connected with identical uplink node and experience approximately equal amount of uplink latency, the nodes can belong to the same cluster. Besides, in a reliable IoT-enabled system, it is expected that the Fog infrastructure providers have applied efficient

networking techniques to ensure persistent communication among the nodes through less variable inter-nodal latency [236].

4.4.2 Fog node architecture

Recently, the OpenFog Consortium has proposed a reference architecture for Fog nodes where the computation, management and networking operations are conducted on discrete components [20]. Based on the reference architecture, we assume that a Fog node is composed of *Controller Component*, *Computational Component* and *Communication Component* (Fig. 4.3).

Computational Component provides resources to execute Application Modules. Inside Computational Component, modules are assigned to *Micro Computing Instances, MCI* where resources e.g. CPU, memory, bandwidth, etc. are allocated according to the requirements. Due to resource constraints, each Fog node can configure a certain number of individually working MCIs at a time. In a Fog node when no MCIs are running, its Computational Component is turned off. In this case, the node only serves networking functionalities like routing, packet forwarding, etc. through its Communication Component. If the load of applications increases in Fog, Computational Component of that node can be reactivated to handle the event. Controller Component of a Fog node monitors and manages the operations of Computational and Communication Component.

Moreover, Controller Component maintains several data structures. Among them, the *Module Sleeping Block (MSB)* contains non-executing Application Modules. When an Application Module has no input to process, it is withdrawn from the assigned MCI and placed to the MSB. Application Modules forwarded from another node also reside inactively in the MSB while they are waiting for scheduling. In addition, a Fog node tracks the Application Modules that are deployed within it using the *Placement List (PL)* and stores route related information of other Application Modules in its *Routing List (RL)*. After the execution of a module, to determine host (both node and MCI) of the next module, either PL or RL of corresponding Fog node is referred to. Besides, a Fog node preserves meta-data of the placed modules even when they are no longer associated with it and the context information of other nodes in a *Data container*.

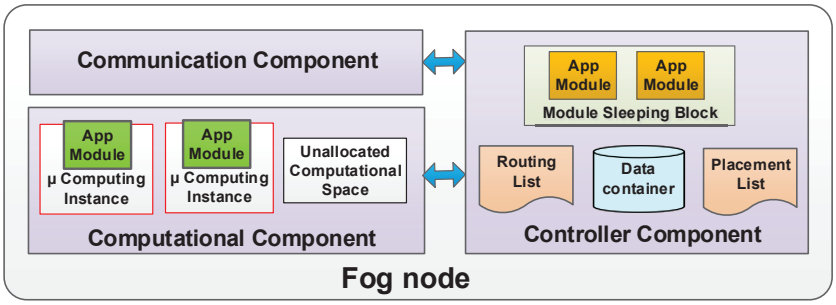

Figure 4.3: Fog node components for latency-aware management

The relevant notations and definitions used in modeling the system are listed in Table 4.2.

4.4.3 Latency Model

Due to data-dependency, generated output of an Application Module is sent to another module as input. The tolerable inter-module data dependency delay $\delta_a^{m'm}$ of module m in application a refers to the maximum amount of time that the module can wait without affecting the application's service delivery deadline to get the input from the previous module m' for a particular data signal. For any application a, the service delivery deadline can be set according to $\sum \delta_a^{m'm}; \forall m \in M_a$.

Here, $\delta_a^{ClientFilter} + \delta_a^{FilterAnalysis} + \delta_a^{AnalysisEvent} < \delta_a^{EventClient}$ is assumed so that tolerable inter-module data dependency delay-aware placement of Data Filtering, Analysing and Event Handler module can spontaneously justify the placement of Client module.

However, inter-module data dependency delay of a module m placed in node n can be estimated ($\gamma_a^{m'm}$) based on the input processing time $\phi_{n'}^{m'}$ of the previous module m' placed in node n' and the inter-nodal communication delay $\Delta_{n'n}$ between their host nodes. In some cases, input scheduling delay of the previous module can also contribute to the estimated inter-module data dependency delay of the respective module.

In addition, the service access rate of data signals for an application and replication number of the previous module play an important role to the input receiving frequency of a particular Application module. The input receiving frequency f_m of a module m itself helps to identify the possible idle period of module m. For example, if $f_m = 2/ms$

Symbol	Definition
N	Set of all Fog nodes.
A	Set of all Applications.
M	Set of all Application Modules.
R	Set of all resources (e.g. CPU, memory, bandwidth, etc.).
M_a	Set of all Application Modules belong to application $a \in A$; $M_a \subset M$.
M_n	Set of all Application Modules placed in node $n \in N$; $M_n \subset M$.
$\gamma_a^{m'm}$	Estimated inter-module data dependency delay of module m from its previous module m' on the dataflow of application a; $m', m \in M_a$
$\delta_a^{m'm}$	Tolerable inter-module data dependency delay of module m from its previous module m' on the dataflow of application a; $m', m \in M_a$
$\Delta_{n'n}$	inter-nodal communication delay between node n' and n; $n, n' \in N$.
f^m	Input receiving frequency of module m; $m \in M$.
$T_{n'n}^m$	Required time to forward module $m \in M$ from node n' to n; $n', n \in N$.
ϕ_n^m	Input processing time of module $m \in M$ in node $n \in N$
r_{req}^m	Requirement of resource $r \in R$ for module $m \in M$.
ψ_{nr}	Capacity of node $n \in N$ for resource $r \in R$.
r_{avail}^n	Available resource $r \in R$ in node $n \in N$.
C_n	Cluster to which node $n \in N$ belongs.
ϵ_C	Maximum communication delay within Cluster C of Fog nodes
t^{now}	Current timestamp.
t_m^{last}	Timestamp when module $m \in M$ received last input.
μ_n^m	Assigned MCI to module $m \in M$ in node $n \in N$.
$y_n \in \{0,1\}$	Equals to 1 if node $n \in N$ is computationally active, 0 otherwise.
$x_{mn} \in \{0,1\}$	Equals to 1 if module $m \in M$ is mapped to node $n \in N$, 0 otherwise.
$x'_{mn} \in \{0,1\}$	Equals to 1 if module $m \in M$ was earlier deployed in node $n \in N$, 0 otherwise.

Table 4.2: Notations for latency-aware management

and $\phi_n^m = 0.2$ ms in node n, then it can be expected that after processing an input, module m in node n will be remain idle for the next 0.3 ms. Within this idle period of module m, its assigned MCI μ_n^m can be allocated to other module for input processing.

4.4.4 Module Management Problem

Usually lower level Fog nodes are not resource enriched like upper level nodes although placement of applications on lower level nodes facilitate faster service access and delivery. Besides, not all applications show identical response to latency related issues. For latency-sensitive applications, service delivery deadline so as to module's tolerable data dependency delay is stringent compared to latency-tolerant applications. In this case, placement of latency-tolerant applications in limited number of lower level Fog nodes can obstruct many latency-sensitive applications to meet their requirements. Conversely, by considering lower Fog level scalable, if all applications are placed there, upper-level nodes will remain under-utilized. Therefore, an efficient module placement policy is required that can prioritize applications to place in closer proximity of data source meeting necessary latency-related issues. More precisely, the policy should identify which applications (modules) should be placed at lower Fog level and which are required to move towards upper level.

Moreover, to minimize energy usage and expenses in Fog environment, the number of computationally active nodes can be optimized. In this case, some modules are required to forward from one node to another. The selection of source and destination node for such module forwarding is very crucial. In addition, while forwarding modules, constraints on nodes capacity, service delivery deadline and forwarding cost (e.g forwarding time) should be observed simultaneously.

In distributed environment like Fog, if the decisions regarding application management is taken decentrally, both application placement time and overhead from the centralized controller will be reduced. Thus application management can be done without relying on a single entity although it will be very difficult to coordinate the nodes.

4.5 Proposed Application Module Management Policy

Our proposed latency-aware Application Module management policy runs on the Controller Component of each Fog nodes without supervision of any external entity. This management policy basically targets application module placement to ensure deadline

satisfied QoS and resource optimization in the Fog layer.

4.5.1 Assuarnce of QoS

To initiate any application a in Fog, the corresponding IoT devices subscribe with a Fog node. This node acts as the *Application gateway node* for application a. Usually during subscription, the Client Module of application a is by default placed on its Application gateway node. Therefore, Application gateway nodes of different applications are located at the lower Fog level. However, placement of the module next to Client Module also initiates from Application gateway node. In order to initiate placement of a module, Fog node executes the *PlaceAppModules* procedure given in Algorithm 3.

PlaceAppModules procedure takes the to be placed Application Module, m, its previous module, m' and observed network delay, ω as arguments.

As shown in Algorithm 3, PlaceAppModules procedure basically consists of 4 steps: At first, the procedure inquires context of the current node (line 2). If the current node is Cloud, rest unassigned modules will be placed there, otherwise the procedure inquires context of the corresponding uplink node and host node of m' (line 3-7). Then the following steps are executed:

1) Sum of the input processing time of previous module m', observed network delay from host node of m' to the current node and the current node's uplink latency is checked with tolerable inter-module data dependency delay of the to be placed Application Module, m (line 9). If it is feasible to route module m to the uplink node, the current node updates its RL for the module. At the uplink node, deployment process of the module is re-initiated by invoking its PlaceAppModules procedure with an updated value of observed network delay, ω (line 10-14).

2) If it is not efficient to route module m to the uplink node, the current node intends to place the module within itself. In order to do so, the resource availability of the current node is checked with the requirement of module m. If the resource availability supports requirements of module m, the current node update its PL for module m (line 15-16). However, as the module is deployed in a computationally active node, boolean variable η is set to *true* (line 17) and availability of the resources in the current node is updated

Algorithm 3 Module placement algorithm

```
1:  procedure PLACEAPPMODULES(m, m', ω)
2:      p ← this node
3:      if p = Cloud then
4:          place rest modules in cloud
5:          return
6:      q ← p.uplinkNode
7:      z ← m'.hostNode
8:      if m ≠ null then
9:          if φ_z^{m'} + Δ_pq + ω < δ_a^{m'm} then
10:             ω ← ω + Δ_pq
11:             p.RL.add(m, q)
12:             q.PlaceAppModules(m, m', ω)
13:         else
14:             η ← false
15:             if r_req^m < r^p_avail, ∀r ∈ R then
16:                 p.PL.add(m)
17:                 η ← true
18:                 p.update(r^p_avail)
19:             else
20:                 for u := C_p.activeNodes do
21:                     if r_req^m < r^u_avail, ∀r ∈ R then
22:                         p.RL.add(m, u)
23:                         u.PL.add(m)
24:                         η ← true
25:                         u.update(r^u_avail)
26:                         break
27:             if η = false then
28:                 select node v ∈ C_p.inactiveNodes
29:                 p.RL.add(m, v)
30:                 v.PL.add(m)
31:                 v.update(r^v_avail)
32:             m' ← m
33:             m ← m'.getNext
34:             p.DeployAppModules(m, m', ε_{C_p})
35:     else
36:         return
```

(line 18).

3) Another computationally active node from the same cluster as the current node is selected to place the module m if available resources at the current node do not meet the module's requirements. This selection is also conducted based on the resource availability of other cluster nodes. In this case, current node updates its RL and the selected cluster node updates its PL, resource availability for module m (line 20-25).

4) If all computationally active cluster nodes fail to allocate resources for deploying module m, an arbitrary computationally inactive node from the cluster will be selected

to place the module. RL and PL of the respective nodes will be updated for the module (line 27-31).

Step 2-4 of the algorithm operate on the same cluster. Therefore, placement process of the next Application Module can be initiated from any node of the cluster. In this algorithm, current node is selected to do so as it simplifies management of routing information. Since, Fog nodes residing in same cluster are connected with faster networking protocols (e.g. CoAP, SNMP, etc.), observed network delay ω in this case is considered negligible and set equal to ϵ of the Cluster.

Algorithm 3 can be extended to handle the scenario when there exist no inactive nodes to host a module within a cluster. In this case, the module can be bypassed to the proximate clusters provided that the tolerable inter module data-dependency delay is not violated. If still the module is failed to deploy, it can be sent either to the uplink nodes or to the proximate cluster nodes where tolerable inter module data-dependency delay gets less violated. It is done so that even if the deadline cannot be meet, service delivery time remain as low as possible. However, if the Fog infrastructure is unable to allocate resources for all the modules of an application, it notifies the user through Application gateway node to flexible the deadline so that it can be placed to the Cloud.

In a reliable IoT-system where the requirements of modules assist them to process input within a fixed amount of time, The proposed Algorithm 3 helps the applications to meet their service delivery deadline. It implicitly deals with latency-sensitive and tolerant applications in different way. According to the policy latency-tolerant applications (modules) are placed vertically whereas latency-sensitive applications (modules) are placed horizontally across the cluster and in lower Fog level resources are preserved for future latency-sensitive applications.

4.5.2 Optimization of Resources

Generally, if all deployed Application Modules of a particular Fog node are re-located to other nodes for further execution, Computational Component of that node can be turned off. Hence, the number of computationally active Fog nodes can be reduced. In Fog, this sort of optimization can be handled in terms of both Integer Linear Programming (ILP)

and heuristic based approaches.

Formulation of a Integer Linear Programming problem

A constrained ILP problem is formulated in Eq. 4.1 to minimize the number of computationally active Fog nodes. It helps to identify possible target Fog node n for re-locating Application Module m through a binary decision variable x_{mn}. Binary variables x'_{mn} and $x'_{mn'}$ tracks whether module m had been available in node n or n' since the last placement. The constraints ensure that a module will not be mapped to multiple nodes (Eq. 4.2), resources of the target node satisfy the module's requirements (Eq. 4.3), placement of the module to target node does not affect the tolerable inter-module data dependency delay of the next module (Eq. 4.4) and the required time to forward the module from source node to target node fits within the input arrival interval of that module (Eq. 4.5).

$$\min \sum_{n \in N} y_n \tag{4.1}$$

subject to,

$$\sum_{n \in N} x_{mn} = 1; \forall m \in M \tag{4.2}$$

$$\sum_{m \in M} r_m^{req} x_{mn} \leq \psi_{nr} y_n; \forall n \in N, \forall r \in R \tag{4.3}$$

$$x_{mn} \gamma_a^{mm''} \leq \delta_a^{mm''}; \forall n \in N, \forall a \in A, \forall m \in M_a \tag{4.4}$$

$$T_{n'n}^m x'_{mn'} (x_{mn} - x'_{mn}) \leq \frac{1}{f^m}; \forall n, n' \in N, \forall m \in M \tag{4.5}$$

This ILP problem is required to be solved periodically to optimize the number of computationally active nodes. Any integer programming solver e.g. SCIP [215] can be used in this case. However, based the solution of ILP problem, modules can be re-located in optimal number of nodes and Computational Component of other active nodes can be turned off.

Proposed heuristic solution

In a Fog environment with large number of computationally active nodes, the aforementioned ILP problem takes much time to be solved. As a consequence, in making real-time forwarding decisions the ILP-based solution will not be acceptable. Therefore, here we propose a heuristic based solution to the problem.

In the heuristic approach, we consider that after latency-aware Application Module placement of any application a, Fog nodes belonging to the same cluster share their context (e.g. PL, RL, Data container information, etc.) with each other. This sort of context sharing among the nodes is conducted within T_s amount of time which is termed as *context sharing period*. After context sharing period, different Fog nodes are found hosting different number of Application Modules. Based on a predefined threshold percentage of allocated resources, some nodes are identified as highly-occupied while others are considered under-occupied.

Due to step 2 and 3 of Algorithm 1, the number of under-occupied nodes in a cluster is comparatively less than highly-occupied nodes. Moreover, to make an under occupied node computationally inactive, only a few Application Modules will be required to relocate. For example, let us assume, there is a cluster of four nodes, each of them can host up to three Application Modules with similar resource requirements. At a particular time, two of them are occupied with two modules each and rest are occupied with one module. The resource allocation threshold for each node is set to 60%. In this case relocation of two Application Modules from a highly-occupied node to other nodes make only one node computationally inactive whereas re-location of Application Modules from two under-occupied nodes can make two nodes computationally inactive. Taking this concept into account, the proposed heuristic approach aims at re-locating modules from under-occupied nodes to other highly-occupied cluster nodes.

In order to conduct re-location of Application Modules from an under-occupied node, n_u to other highly-occupied cluster nodes, at first n_u forwards the modules in non-executing form to each of the nodes. Within the highly-occupied cluster nodes, forwarded modules reside in MSB. If a highly-occupied node accommodates any of the forwarded modules in its Computational Component, the RL of n_u is updated for that module. In this case, other cluster nodes discard the module from their MSB. Otherwise,

at highly-occupied cluster nodes, forwarded modules are required to be scheduled in MCIs of other Application Modules.

After forwarding non-executing form of all placed modules, an under-occupied node n_u usually tries not to execute the modules in its Computational Component. In this case, if n_u receives input τ for module m of application a, it either routes the input to new host node of the respective module or asks a suitable highly-occupied node to schedule the module through *ForwardAppModules* procedure (Algorithm 4).

Algorithm 4 is consisted of two basic steps. After finding the context of current node (line 2), the following steps are executed:

1) In the RL of current node, if reference of new host node for module m is found, input τ will be sent to that node (line 3-6).

2) To identify a suitable host module and its assigned MCI for scheduling Application Module m, Algorithm 4 takes each highly occupied cluster nodes into account (line

Algorithm 4 Module forwarding algorithm

```
1:  procedure FORWARDAPPMODULES(m, a, τ)
2:      p ← this node
3:      q ← p.RL.get(m)
4:      if q ≠ null then
5:          send τ to m on node q
6:          return
7:      m'' ← m.getNext
8:      host_n ← null
9:      host_m ← null
10:     for n' := C_p.highNodes do
11:         for m' := M_n' do
12:             n'.getInfo(m')
13:             if t^now > t^last_m' + φ^m'_n' then
14:                 if t^last_m' + 1/f_m' − t^now > φ^m_n' then
15:                     if r^m'_req ≥ r^m_req, ∀r ∈ R then
16:                         if x_mn' = 1&γ^mm''_a ≤ δ^mm''_a then
17:                             if λ_m' = false then
18:                                 host_n ← n'
19:                                 host_m ← m'
20:                                 λ_m' ← true
21:                                 n'.updateInfo(m')
22:                                 break
23:         if host_n && host_m ≠ null then
24:             break
25:     if host_n && host_m ≠ null then
26:         schedule m in μ_host_m on node host_n
27:         send τ to m on node host_n
```

10-11), parse relevant information (line 12) and checks the following conditions:

i. host module is not currently processing any input (line 13). It ensures that, m will be scheduled in MCI of host module only when it is idle.

ii. host module will not receive any input until module m finishes input processing (line 14). This condition ensures that re-location process will not discard any input of host module.

iii. the assigned MCI to host module meets the resource requirements of module m(line 15).

iv. Placement and execution of module m on the host node will not affect the tolerable inter-module data dependency delay of its next module (line 16).

v. no other under-occupied nodes have selected the MCI assigned to host module for scheduling their Application Modules (line 17). For any host module, m' this condition is observed through boolean variable $\lambda_{m'}$. After identifying the host node, necessary information are updated (line 18-21). As soon as scheduled Application Module finishes input processing, $\lambda_{m'}$ is set to *false* again.

Here, Algorithm 4 employs first fit solution to schedule Application Module m. However, after re-location of modules from under-occupied nodes to highly-occupied nodes, there will be an observation period. Within this period if no anomaly (e.g. failure in scheduling forwarded modules, QoS degradation, etc.) is detected, soon after the observation period, Computational Component of under occupied nodes will be turned off. Hence, the number of computationally active nodes from the Fog can be reduced. Besides, the proposed policy dynamically determines host node and host module for the forwarded modules which helps to deal with sudden changes in input receiving frequency (e.g. due to add new replica) of the modules.

Our proposed heuristic based resource optimization through latency-aware Application Module forwarding operates within clusters. In this approach, usually a small number of Application Modules from under-occupied nodes are forwarded to highly-occupied cluster nodes. Since highly-occupied cluster nodes contain many potential host modules, there will be always a possibility of finding suitable MCI to schedule less amount of forwarded modules. Moreover, cluster nodes are connected with each other with faster communication protocols. Therefore, communication latency during

module forwarding is negligible and does not obstruct application QoS. However, for a forwarded module if no suitable host node and host module is found, the module will be executed in its initial placement. In that case, there will be no further scope of forwarding data signal endlessly without being accepted.

4.6 Performance Evaluation

The performance of the proposed Application Management policy is evaluated in two phases; At first, the proposed latency-aware module placement is compared with the approarches mentioned in [221] and [222]. In [221], a latency-aware iterative algorithm is introduced to place applications whereas in [222], a centralized resource coordinator-based Service Oriented Resource Sharing (SORS) is discussed. In this phase, deployment time of modules, percentage of deadline satisfied data signals are considered as the performance metrics.

Later, the proposed heuristic based solution for resource optimization is compared with the solution of ILP problem. In solving the ILP problem, SCIP solver [215] is used. The proposed latency-aware Application Module forwarding is also compared with MigCEP [223] and Peer VMs Aggregation (PVA) [224]. In MigCEP, to forward applications time-graph models are generated, algorithms for shortest path and co-ordination are executed whereas in PVA, a migration manager handles necessary steps to forward applications. In this phase, number of reduced Fog nodes, required time for identifying the target nodes and forwarding the modules are considered as performance metrics. Moreover, when scheduling of forwarded modules are required at the target nodes, the performance of the proposed approach in reducing the number of context switching is compared with vSlicer [231] and earliest start time-based scheduling [227]. Increasing number of context switching can incur high service waiting time and cost.

In addition, the performance of the proposed policy is discussed in terms of varying application contexts such as variable input processing and communication time of the modules along with sudden changes in application service access rate.

Parameter	Value
Simulation Duration	120- 240sec
Status sharing and observation period	10sec
Uplink latency:	
IoT device to LL nodes	10-15 ms
LL nodes to ML nodes	30-40 ms
ML nodes to HL nodes	60-80 ms
HL nodes to Cloud	140-160 ms
Processing time:	
Client Module	20-40 ms
Filter Module	10-20 ms
Analysis Module	150-200 ms
Event handler Module	20-40 ms
Applications service delivery deadline	350 - 750 ms
Delay to connect with centralized manager at ML	45-60 ms
Maximum nodal communication delay within Fog cluster	3-5 ms
Applications data receiving frequency	3/sec - 7/sec

Table 4.3: Simulation parameters for latency-aware management

4.6.1 Simulation Environment

To evaluate the performance of the proposed policy, a Fog environment is simulated in iFogSim [237]. iFogSim is built upon CloudSim [238] framework that is used widely for simulating different computing paradigms [239] [240]. The simulation parameters are summarized in Table 4.3.

In the modelled environment, we assume that Fog layer consist of three levels e.g. lower level (LL), mid level (ML), higher level (HL) and every node is heterogeneous to each other in terms of resource capacity and application execution environment. To conduct the experiments, we have used synthetic workload as compatible real workload for the proposed Application Management policy is not currently available. The value of simulation parameters within a specific range is determined by a pseudo random number generator. Here, application initiation request can be originated from any location

at any time. We consider that due to incompleteness of data, deployed applications discard 2-3% of received signals during data filtration and 65% of the placed applications are comparatively more latency-sensitive than the rest.

4.6.2 Performance in Application Module deployment

In Iterative algorithm, at first modules are deployed temporarily in different nodes. Then, for reducing service latency, modules are gradually re-located to suitable nodes through iterations. As the application number increases, required time for iteration also gets high. In SORS policy, to place modules, each time resource coordinator is required to be asked for suitable nodes. In the proposed module placement approach, neither iteration nor supervised resource discovery is applied. Therefore, to place increasing number of applications, the proposed requires less time compared to others (Fig. 4.4). This experiment also reflects that application placement decisions taken centrally can linger the placement of the applications in distributed Fog environment.

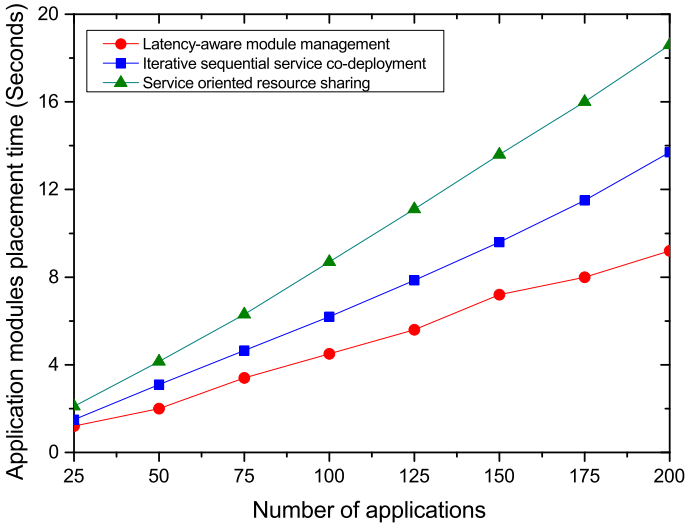

Figure 4.4: Module deployment time for varying number of applications

Fig. 4.5 depicts the percentage of deadline satisfied data signals for increasing number of applications. Iterative algorithm treats both latency-sensitive and tolerant applications in a similar way. As a result, in some cases, percentage of deadline satisfied data

signals for sensitive applications degrades. In SORS, for sending input to each modules of an application, resource coordinator is sent request for finding the host nodes. Additional time is required to conduct this operation which adversely affects the percentage of deadline satisfied data signals. In our proposed approach, host nodes send input from one module to another and due to place modules based on latency-endurance, neither latency sensitive nor tolerant applications are penalized in meeting deadline for processing the received data signals. This experiment result indicates that when in a system diversified applications in respect of latency-endurance exist, it is always a good policy to handle them separately.

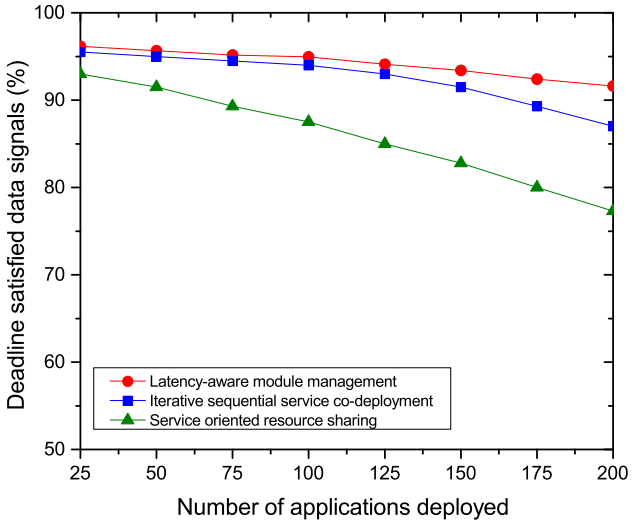

Figure 4.5: Percentage of QoS satisfaction for varying number of applications

4.6.3 Performance in Application Module forwarding

Fig. 4.6 shows the comparison of ILP based solution and the proposed heuristic solution in optimizing the number of computationally active Fog nodes. From the experimental results, it is found that the proposed heuristic solution is very much closer to the optimal. In this experiment, every after 10 seconds,the ILP problem has been solved.

In Fig. 4.7, required time to identify possible target nodes for module forwarding is depicted for both ILP and heuristic based solution. The heuristic based solution can find suitable target nodes within a cluster by executing a simple threshold compari-

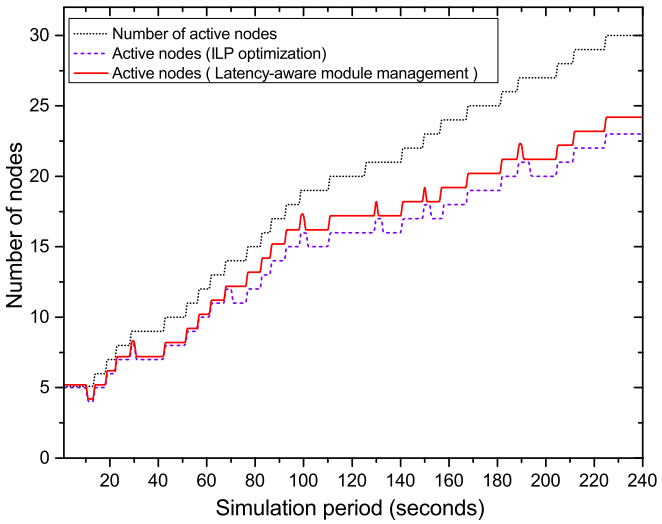

Figure 4.6: Comparison of ILP and heuristic based latency-aware solution

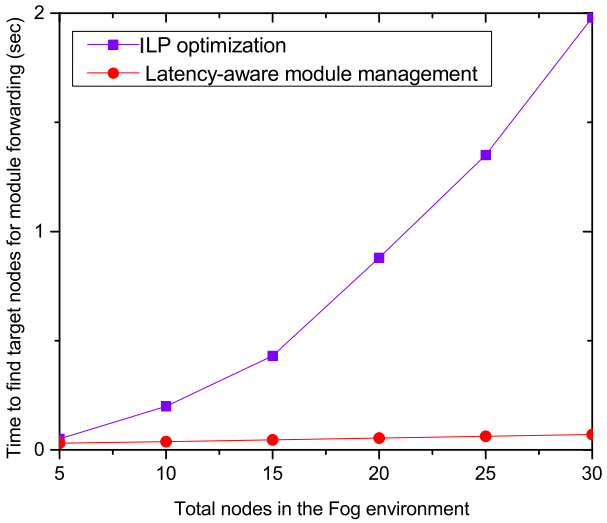

Figure 4.7: Required time for generating ILP and heuristic solutions

son. When all clusters in Fog applies the heuristic approach in contemporary basis, less amount of time will be required to identify possible target nodes from the whole system. However, in ILP based solution, required time for identifying target nodes exponentially increases as the number of nodes increases. This experiment result defines that in a system where real-time interactions happen very frequently, solving a time consuming ILP problem for forwarding modules is not very efficient.

A comparative study of the proposed module forwarding approach, MigCEP and

PVA is depicted in Fig. 4.8. In MigCEP, several operations such as time-graph model generation, shortest path identification and co-ordination are conducted to forward modules. In PVA, identification of target nodes, competence checking and communication management during module forwarding are observed by a migration manager. Due to aforementioned reasons, both approaches require higher amount of time. In the proposed approach, rather than identifying a suitable node, modules are forwarded to every competent nodes in the cluster. As cluster nodes are connected with each other through faster networking standard, this type of module forwarding requires less amount of time compared to others. Although it brings additional cost for storage, for management of real time applications, it can be overlooked. Besides, this experiment signifies that formation of high-speed clusters among Fog nodes can contribute extensively to forward modules so as to optimize resources.

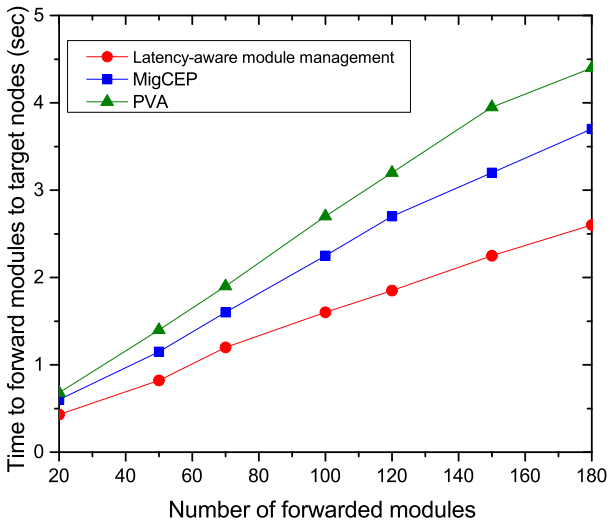

Figure 4.8: Required time for module forwarding

Fig. 4.9 depicts how input receiving frequency of host module influences context switching when a forwarded module is scheduled in host module's MCI. In the proposed approach, if the host module's frequency increases, number of context switching decreases whereas in *vSlicer* scheduler, this number remains the same (here, 16 context switching per second) and for early arrival time-based scheduling it increases. Rapid context switching increases overhead and waiting time at the host node node. In the

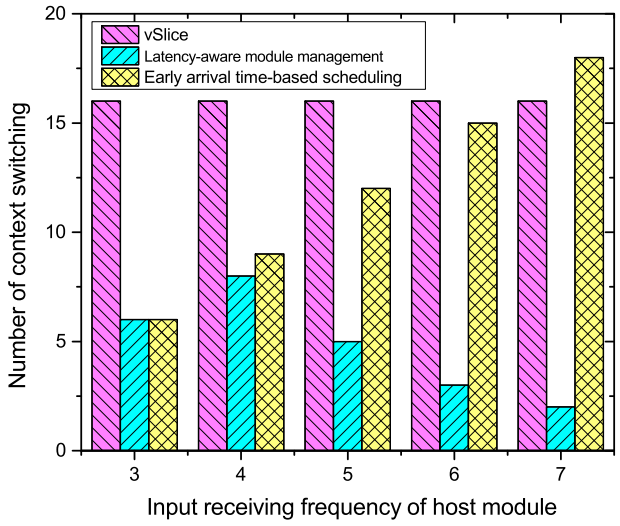

Figure 4.9: Number of context switches at different data receiving frequency of host

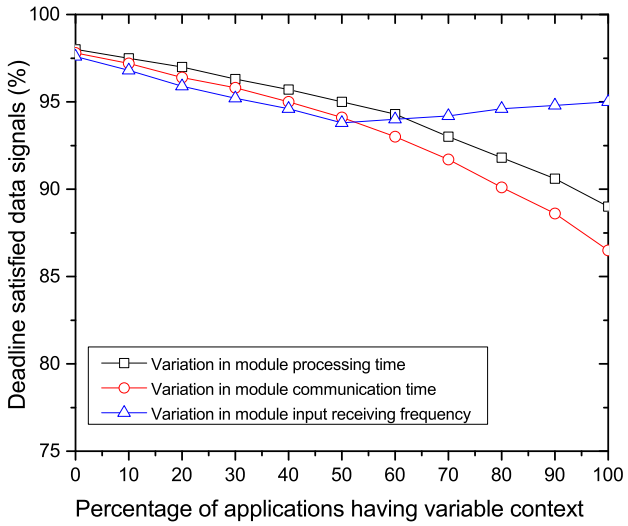

Figure 4.10: Performance of the proposed policy for varying application context

proposed approach, a forwarded module only get access to the host module's MCI when the module is idle. Therefore no data signal of both host and scheduled module waits for long time and additional overhead of context switching is reduced. This experiment highlights that module forwarding decisions in distributed Fog environment should be taken dynamically based on the context of the nodes.

Fig. 4.10 represents how the proposed policy deals with varying application context.

The application context can be varied in terms of input processing and communication time of the modules and the service access rate of the data signals. The experiment result shows that, if the processing time of the modules varies with course of time for most of the applications, percentage of QoS-satisfied data signals will be reduced. However, for varying inter-communication among the modules (inter-nodal communication latency), this QoS degradation rate is higher compared to processing time variations of the modules since in distributed placement, inter-nodal communication delay is considered as the dominating factor. Moreover, if the service access rate of the applications so as to the modules changes dynamically, initially QoS-degrades specially for the forwarded modules. When the percentage of varying applications gets increased, according to the policy, no modules are forwarded. As a consequence, QoS-satisfaction rate increases. The experiment is conducted by varying one parameter at a time and the results signifies that the proposed policy works well for reliable IoT enabled system where inter-nodal communication delay does not vary significantly and all the modules are allocated with resources according to their requirements.

4.7 Summary

The Fog computing paradigm has a great potential to support a wide variety of IoT applications. We propose a latency-aware Application Module management policy that targets both deadline based QoS of applications and resource optimization. The proposed management policy meets the latency in service delivery for applications having rigorous deadline. Besides, it investigates how to optimize number of resources without violating QoS of the applications. Two algorithms have been developed in support of our proposed application management policy. The first is about Application Module placement and the second one simplifies a constrained based optimization problem in forwarding modules towards the inactive resources of idle modules. We also conducted simulation experiments in iFogSim, which shows the potential of the proposed policy.

Chapter 5

Context-aware Application Management

In this chapter, Industry 4.0 is considered as a use case for Fog computing. The fourth industrial revolution, widely known as Industry 4.0, is realizable through widespread deployment of Internet of Things (IoT) devices across the industrial ambiance. Fog computing focuses on harnessing edge resources to place and execute different IoT applications assisting Industry 4.0. Since most of the Fog nodes are resource-constrained, it is challenging to place Industry 4.0-Oriented Applications (I4OAs) over them ensuring time-optimized service delivery. Diversified data sensing frequency of industrial IoT devices and their data size further intensify the application placement problem. To address this, we propose a context-aware application placement policy that coordinates the IoT device-level contexts with the capacity of Fog nodes and minimizes the service delivery time of various I4OAs. It also ensures that the streams of input data neither congest the network nor increase the computing overhead of host Fog nodes. Our policy offers overall 16% improvement in service latency, network relaxation and computing overhead management compared to other placement policies.

5.1 Introduction

Device to device connectivity, real-time data access, and advanced automation are rapidly leading the current industrial practice towards its fourth revolution named Industry 4.0. It focuses on building smart industries by enabling robotic assistance, digital twin, and

This chapter is derived from:

- **Redowan Mahmud**, Adel Nadjaran Toosi, Kotagiri Ramamohanarao, and Rajkumar Buyya, "Context-aware Application Placement for Industry 4.0 in Fog Computing Environments", *IEEE Transactions on Industrial Informatics* (in press, DOI: 10.1109/TII.2019.2952412, accepted on November 5, 2019).

proactive failure management [241]. Internet of Things (IoT) is one of the key elements for Industry 4.0 [242]. Industrial IoT devices generate a huge amount of data per unit time. These data require real-time processing through various Industry 4.0-Oriented Applications (I4OAs) so that different aspects of Industry 4.0 can be achieved [243]. For example, image processing and navigation applications help to launch robotic assistance in the industries. If these applications fail to deliver their services in due time, the performance of industrial robots will degrade significantly. Since Cloud data centers reside at multi-hop distance from IoT devices, processing of industrial IoT-data using Cloud-based I4OAs is not feasible. It increases data propagation delay, network congestion, and application service delivery time. Therefore, Fog computing aims to overcome such limitations of Cloud-centric IoT-models by harnessing the edge resources [244].

In Fog-enabled industries, machines with computing processors such as Raspberry PIs, computers, routers, and micro-data centers act as Fog nodes. These nodes offer Infrastructure as a Service (IaaS) like Cloud data centers to assist the execution of different I4OAs [245]. However, Fog nodes are deployed in a distributed manner, and they are heterogeneous in processing speed and networking standards. Additionally, the features of IoT devices such as their data sensing frequency and size of data differ from one to another [242]. These features play vital roles in defining application characteristics. For example, the compute intensity of an I4OA has a proportional relationship with its input data size. Similarly, the network intensity of an I4OA changes based on how frequently the associated IoT devices are sending data to that application [246]. Consequently, these features of IoT devices incite the computational and networking load of host Fog nodes during application execution. If the available capacity of Fog nodes fails to deal with them, network congestion can occur, and the computing overhead of Fog nodes can increase drastically. It also affects the deadline-satisfied service delivery of I4OAs. Hence, it is important to consider these issues while finding the suitable placement option for an I4OA in Fog environments.

Different application placement policies for Fog and other computing paradigms have already been proposed in the literature [247] [248] [127]. They narrowly exploit data size and sensing frequency of IoT devices while making placement decisions for applications. As a result, they often fail to grasp the characteristics of applications and

manage the resources efficiently. In some cases, an application is placed on multiple computing nodes, and input data are scheduled to them under the supervision of a centralized entity [247]. When the arrival rate of inputs becomes high, such an application placement policy increases overhead on the centralized entity. It also impels to change the processing destinations of input data very frequently. However, as an alternative, IoT devices themselves can schedule the input data to different replicas of an application. Nevertheless, it increases communication and computation burden for low-energy IoT devices [249]. Thus, in both approaches, application service delivery time degrades. To address these shortcomings, in this work, a context-aware application placement policy for Fog environments is proposed.

Context awareness denotes the ability of a system to deal with the state or contextual information of different entities interacting with the system at any given time and adapt its performance accordingly [250]. In industrial environments, IoT devices, Fog nodes, and I4OAs seamlessly interact with varying data size and sensing frequency, computing and networking capacity, and QoS requirements. Therefore, without context-aware approaches, it is challenging to improve the efficiency of decision-making operations in Industry 4.0. In our proposed placement policy, the sensing frequency and data size of IoT devices are regarded as the device-level contextual information because of their direct impact on Fog node functionalities and application characteristics. Here, based on their implications, the processing and the propagation time of input data for corresponding I4OAs are determined. The proposed policy jointly considers the computation and networking capacity of Fog nodes and the QoS requirements of applications, including their service delivery deadline during application placement. Additionally, it resists the increment of computing overhead on the host Fog nodes and prevents the streams of input data from congesting the network. Thus, it helps to improve service reliability and service time for different I4OAs in Fog environments. The main **contributions** of the work are:

- Proposes a placement policy for Industry 4.0-Oriented Applications (I4OAs) in Fog environments that optimizes their service time by coordinating IoT device-level contexts with the capacity of Fog nodes.

- Ensures processing of input data streams through placed applications by managing network congestion and computation overhead of host Fog nodes.
- Evaluates the performance of proposed policy in *FogBus*-enabled real [24] and *iFogSim*-based simulated Fog environments [219], and compares it with existing policies in respect of service delivery time, network relaxation and computing overhead management.

The rest of the chapter is organized as follows. In Section 5.2, related researches are reviewed. Section 5.3 provides the system overview and the software architecture of a Fog gateway node. Section 5.4 describes the implication of contextual information in the modeled system, formulates the application placement problem, and discusses our solution. The performance of our policy is evaluated in Section 5.5. Finally, Section 5.6 concludes the chapter.

5.2 Related work

In the literature, there exist several works that highlighted the applicability of Fog computing in Industry 4.0 [245] [251]. Additionally, different placement policies for I4OAs are proposed. For example, Verba et al. [127] profiled I4OAs as per their inputs. It helps to place applications with enhanced service time and minimizes the effect of context-variation. Lin et al. [252] proposed a Fog node deployment policy in a hierarchical platform that meets the latency and capacity constraints of applications. Similarly, Chekired et al. [253] prioritized the placement of I4OAs based on their latency sensitivity. Wan et al. [254] also developed a policy that balances the application execution load on manufacturing components and relates their energy usage with data size.

Not only in Industry 4.0, but the concept of Fog computing has also been extended to other IoT-enabled systems including Healthcare 4.0 [255] and digital agriculture [256]. There exist some application placement policies for such systems. For example, Minh et al. [248] proposed a context-aware framework for IoT-Fog-Cloud infrastructure that considers location, application deadline, and resource availability. The application placement policy proposed in [257] deals with the variations of device-level contexts and net-

Work	IoT contexts		Meets QoS	Manages overhead	Stable placement
	Sensing rate	Data size			
[247]		✓	✓		
[248]		✓	✓	✓	
[127]	✓	✓	✓		
[249]	✓			✓	✓
[252]	✓	✓	✓		
[253]		✓	✓	✓	
[254]		✓		✓	✓
[257]		✓		✓	✓
[258]	✓		✓		✓
[259]	✓	✓	✓	✓	
[260]		✓	✓	✓	
Context-aware (This work)	✓	✓	✓	✓	✓

Table 5.1: Summary of related work for context-aware management

work topology and places the applications accordingly. Moore et al. [258] also provided a placement policy that engages a centralized entity for context analysis and assists low-latency service delivery of the applications. Afzal et al. [259] considered data size and sensing rate of end devices while transferring inputs to applications in energy-efficient and timely manner.

Apart from Fog computing, various application placement policies are also discussed for other computing paradigms. For example, Haferkamp et al. [247] developed a policy for cyber-physical systems that exploits payload size and deadline to prioritize the scheduling operations. Lee et al. [249] proposed another policy for Mobile Computing that predicts the launching time of applications and improves the energy usage of smartphones. Gu et al. [260] explored the local and remote computation capabilities along with the network condition and latency constraints while placing applications in Mobile Edge Computing environments.

Table 5.1 presents a summary comparison of related works with the proposed pol-

icy. During application placement, IoT device-level contexts such as sensing frequency and size of data are not exploited thoroughly in existing works along with the capacity of Fog nodes and application QoS requirements. Furthermore, context parameters are not leveraged to determine the network and computing overhead of Fog nodes. Consequently, they often fail to assist data streams and lead different input of a particular stream to be processed on different Fog nodes. In this work, we address these issues by developing a placement policy for I4OAs. It applies IoT device-level contexts to determine the overhead of Fog nodes and takes the application placement decision accordingly. It ensures application QoS and manages the computational load of Fog nodes. Furthermore, it offers stable placement that resists the changing of processing destination for a particular stream until any context alteration occurs. The proposed policy can also run on different Fog nodes without the supervision of a centralized entity.

5.3 System Overview

5.3.1 Organization of Fog Computing Environments

In industry, IoT devices and Fog Computing Nodes (FCNs) are arranged in the conceptual hierarchical order, as shown in Fig. 5.1. At the lowest level, IoT devices reside. They sense industrial ambiance and forward data to FCNs for processing through placed I4OAs. Computing capabilities and peer-to-peer communication standards vary from one FCN to another. The applications placed on an FCN can directly access its physical resources through the host operating system. Based on the service outcome of these applications, IoT devices can trigger physical actions through actuators. In either case, service outcomes can be stored for further operations in the future. FCNs can form several clusters among themselves using faster communication protocols such as Constrained Application Protocol (CoAP) and Simple Network Management Protocol (SNMP). In Fog environments, there also exist some Fog Gateways (FGs) that assist the interfacing of IoT devices with the Fog Clusters.

IoT devices can subscribe with any of the FGs to launch placement requests for associate applications. They also notify the device-level contexts, such as the sensing fre-

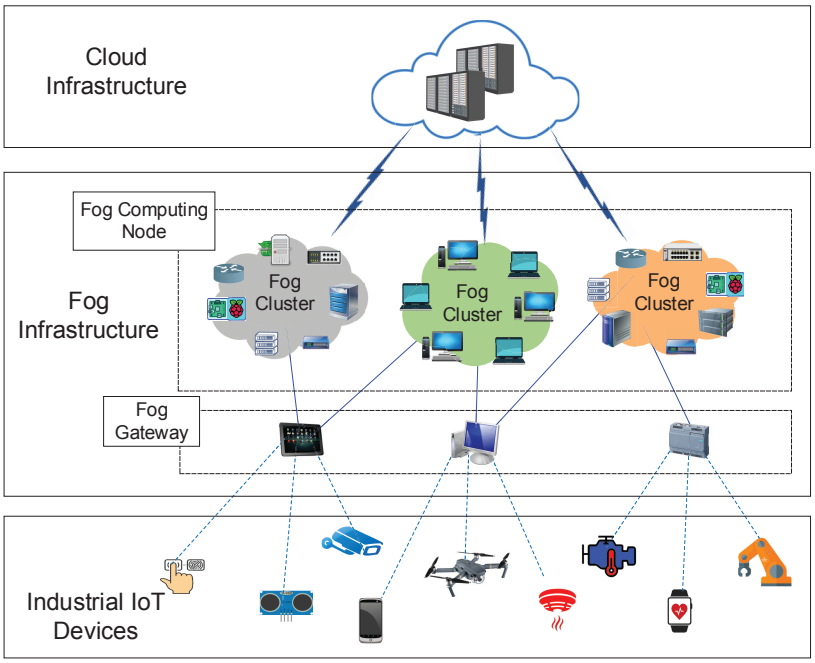

Figure 5.1: Computing environment in a industry

quency and size of data to the FGs. FGs communicate with different FCNs within Fog Clusters to grasp their capacity staus using RESTful APIs. Later, based on received contextual information of IoT devices, capacity status of FCNs, and QoS requirements of requested applications, FGs identify service delivery time optimized application-FCN placement map. If an FCN is accessible through multiple FGs, their operations on that node are synchronized by the FCN. Whenever the Fog environment becomes overloaded or any latency-tolerant application is requested, the FGs communicate with Cloud data centers to assist them using remote resources. Cloud data centers also help FCNs by offering scalable storage to preserve their accumulated data.

The system model mentioned above facilitates simplified third-party access to I4OAs and Fog Clusters. Consequently, it can get affected by security and privacy threats such as impairment of information, disclosure of device identity, replay, and Denial of Service (DoS) attacks. These threats resist Fog computing to support I4OAs with guaranteed performance. Therefore, we consider the existence of an edge network-based security framework [261] in the modeled Fog environment for securing the application services and infrastructure. We also deem that the system supports preemptive operator migra-

tion, request re-submission, and data replication through existing proactive and reactive fault tolerance techniques [262] [263] to resist different anomalies, including request time out, node failure, response error, and application breakdown, and ensures reliability.

5.3.2 Software Architecture for Fog Gateways (FGs)

Fig. 5.2 presents the software architecture of FGs for context-aware application placement. Its details are given below.

IoT device Handler: It grasps the context of IoT devices such as the sensing frequency and average size of data, and stores them in a repository. It also narrates the placement requests to a catalogue service and monitors the contextual changes of IoT devices.

Context Repository: It stores the contextual information of IoT devices and the capacity status of accessible FCNs. It also connects Cloud data centers for large-scale storage and maintains a data structure called *PlacementMap* to track which application is placed on which FCN.

Application Catalogue: It contains the details of different I4OAs, including their execution model, time and space complexity, dependency, and resource requirements. For various inputs, it can also enable the profiling information of an application, such as its processing time and the number of instructions on different FCNs.

Application Placement Engine: It assesses the compatibility of FCNs to host the requested I4OAs based on the contextual information of IoT devices and the capacity status of FCNs. It also initiates the application placement command for the host FCN. Once an application is placed to an FCN, its information is updated on the Context Repository.

5.4 Proposed Application Placement Policy

Based on the implications of contextual information, our proposed context-aware application placement policy identifies application-FCN map and ensures time-optimized service delivery for the requested applications. In a Fog environment, it is executed by the FGs. Basic notations to realize the policy are given in Table 5.2. We have discussed

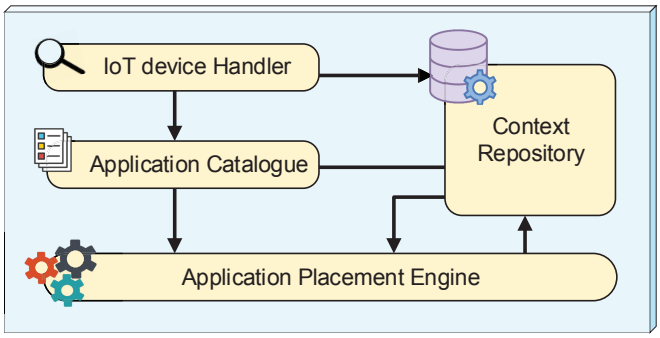

Figure 5.2: Software architecture of Fog Gateways

different aspects of our policy in the following subsections.

5.4.1 Implications of Contextual Information

The context of IoT devices along with FCN and application-centric information help to determine the data propagation time and processing time for I4OAs. Let R and C be the set of I4OAs and the set of accessible FCNs for an FG respectively. Data propagation time p_{rc} to an FCN $c \in C$ for an application $r \in R$ is formulated as Eq. 5.1, where average input data size $\overline{l^r}$ is extracted from IoT device-level context and λ^c denotes the network bandwidth of FCN $c \in C$.

$$p_{rc} = \frac{\overline{l^r}}{\lambda^c} \tag{5.1}$$

Similarly, data processing time q_{rc} for application $r \in R$ on FCN $c \in C$ is calculated using Eq. 5.2. In this case, based on the average size of input data $\overline{l^r}$, the number of instructions s^r in application r is extracted from its profiling information and μ^c refers to the instruction execution speed of FCN c.

$$q_{rc} = \frac{s^r}{\mu^c} \tag{5.2}$$

Since, the ultimate service delivery of an application ends with either a storage or actuation command, its transferring time within reliable Fog network is considered negligible. Therefore, the service delivery time t_{rc} of application $r \in R$ on FCN $c \in C$ for a single

input data is written as Eq. 5.3

$$t_{rc} = p_{rc} + q_{rc} \tag{5.3}$$

These calculations refer that an application requires to occupy networking and computing resources of an FCN for p_{rc} and q_{rc} amount of time to receive and process an input data. *Networking resource occupancy* α_{rc} of application r on FCN c denotes the total amount of time when application r occupies networking resources of that FCN for receiving all its input data sensed in per unit time. It is calculated using Eq. 5.4 where f^r signifies the data sensing frequency of IoT devices for application r. Likewise, using Eq. 5.5, its *Computing resource occupancy* β_{rc} is calculated. It refers the total time that application r requires to process f^r amount of input data occupying computing resources of FCN c.

$$\alpha_{rc} = f^r \times p_{rc} \tag{5.4}$$

$$\beta_{rc} = f^r \times q_{rc} \tag{5.5}$$

According to Eqs. 5.4 and 5.5, total Networking and Computing resource occupancy (Φ_c and Ψ_c respectively) for all placed applications $r' \in Z_c$ on FCN c is calculated as Eqs. 5.6 and 5.7.

$$\Phi_c = \sum_{r' \in Z_c} \alpha_{r'c} \tag{5.6}$$

$$\Psi_c = \sum_{r' \in Z_c} \beta_{r'c} \tag{5.7}$$

5.4.2 Identification of Placement Map

The applications requested for placement in an industrial scenario can have a diversified level of computing and network intensity. Additionally, their QoS requirements can vary from one to another. Therefore, it is required to focus on a generalized objective for all applications during their collective placement. The proposed context-aware application placement policy resolves this issue by minimizing the service delivery time of applications for each input data. It also helps the policy to deal with the characteristic variations of different applications as the computing and network intensity of appli-

Sign	Definition
C	Set of FCNs accessible through an FG.
R	Set of applications requested for placement to an FG.
Z_c	Set of applications placed on FCN $c \in C$.
f^r	Data sensing frequency of IoT devices for application $r \in R$.
$\overline{l^r}$	Average input data size for application $r \in R$.
σ^r	Amount of data dealt by application $r \in R$ in per unit time.
s^r	Number of instructions in application $r \in R$ based on $\overline{l^r}$.
δ^r	Service delivery deadline for application $r \in R$.
μ^c	Instruction execution speed of FCN $c \in C$
λ^c	Network bandwidth of FCN $c \in C$
p_{rc}	Data propagation time to FCN $c \in C$ for application $r \in R$
q_{rc}	Data processing time of application $r \in R$ on FCN $c \in C$
t_{rc}	Service delivery time of application $r \in R$ on FCN $c \in C$
α_{rc}	Networking resource occupancy of application $r \in R$ on FCN $c \in C$ for receiving f^r input data.
β_{rc}	Computing resource occupancy of application $r \in R$ on FCN $c \in C$ for processing f^r input data.
Φ_c	Total networking resource occupancy $\forall r \in Z_c$ on FCN $c \in C$
Ψ_c	Total computing resource occupancy $\forall r \in Z_c$ on FCN $c \in C$
N_r^τ	Set of inputs for application r received in τ amount of time
m_c	CPU usage of FCN c per unit time interval
M_c^τ	Set of m_c values of FCN c monitored for τ amount of time
x_{rc}	Equals to 1 if FCN $c \in C$ is hosting application $r \in R$, 0 otherwise.

Table 5.2: Notations for context-aware management

cations directly influence the service delivery time. Moreover, the application service delivery time on specific FCN does not vary significantly for each input when the IoT device-level contexts and the load on FCNs remain unchanged. However, in the real-world, the placement of multiple applications on a single FCN without considering the effect of different IoT device-level contexts can congest the network and increase the computational overhead of the FCN. As a consequence, service delivery time for all of its occupant applications degrade. Therefore, a balance between their input data admittance and processing on that FCN is required. Furthermore, the application service for

each input should be delivered within the deadline to meet QoS. Depending on such facts, we formulate the context-aware application placement as a multi-constrained Integer Linear Programming (ILP) problem as described below.

Formulation of Application Placement Problem

Eq. 5.8 signifies the objective function of proposed application placement policy. It minimizes application's service delivery time for each input data and identifies application-FCN mapping through a binary decision variable x_{rc}. Constraints of this ILP problem ensure that an application will not be placed to multiple FCNs (Eq. 5.9) and its service delivery time will meet the deadline (Eq. 5.10). Furthermore, Eqs. 5.11 and 5.12 refer that Networking and Computing resource occupancy of all applications placed on an FCN should not surpass the duration of per unit time. Thus it maintains a balance between per unit time data admittance and processing through the applications.

$$\min \sum_{r \in R} x_{rc} t_{rc} \tag{5.8}$$

subject to,

$$x_{rc} \leq 1; \forall r \in R \tag{5.9}$$

$$t_{rc} < \delta^r; \forall r \in R \tag{5.10}$$

$$\Phi_c + \alpha_{rc} \leq 1; \forall c \in C \tag{5.11}$$

$$\Psi_c + \beta_{rc} \leq 1; \forall c \in C \tag{5.12}$$

Any ILP solver, for example, SCIP [215] can be used to solve this optimization problem and identify the mapping of applications and FCNs. However, in Fog environments, when an FG maintains connections with large number of FCNs and receives placement requests for numerous I4OAs, a longer period of time is required by ILP solvers to solve such optimization problem. It is not acceptable during real-time interactions. Therefore, we propose a heuristic solution to solve the application placement problem.

Heuristic Solution for Application Placement

The heuristic solution for application placement is immanent in *PlaceApplication* procedure presented in Algorithm 5. It identifies the FCN for placing an application which ensures least service delivery time for its input. Details of Algorithm 5 is discussed here.

PlaceApplication procedure takes the set of accessible FCNs C and set of applications R requested to an FG g for placement as arguments. It consists of 3 steps:

1. For each application $r \in R$, the amount of data σ^r that an FCN requires to deal with in per unit time for hosting the application is calculated (line 2-3). It refers to the data load of the application which depends on the average size of input data $\overline{l^r}$ and sensing frequency of corresponding IoT devices f^r. An application that deals with huge data load is considered heavyweight and is more likely to promote network congestion and computing overhead compared to lightweight applications having less data load. To reduce the scope of any impediments, it is preferable to place heavyweight applications in earliest convenience than lightweight applications. Therefore, Algorithm 5 sorts all

Algorithm 5 Application Placement algorithm

```
1:  procedure PLACEAPPLICATION(C, R)
2:      for r := R do
3:          σ^r ← f^r × l^r
4:      R' ← descendingSort(R, σ^{∀r∈R})
5:      for r := R' do
6:          Υ_r ← ∞
7:          X_r ← null
8:          for c := C do
9:              p_rc ← l^r / λ^c
10:             q_rc ← s^r / μ^c
11:             t_rc ← p_rc + q_rc
12:             α_rc = f^r × p_rc
13:             β_rc = f^r × q_rc
14:             if t_rc < Υ_r then
15:                 if t_rc < δ^r then
16:                     if Φ_c + α_rc ≤ 1 then
17:                         if Ψ_c + β_rc ≤ 1 then
18:                             Υ_r ← t_rc
19:                             X_r ← c
20:         if X_r ≠ null then
21:             g.PlacementMap(r, X_r)
22:             Z_{X_r} ← Z_{X_r} ∪ r
23:             Φ_{X_r} ← Φ_{X_r} + α_{rX_r}
24:             Ψ_{X_r} ← Ψ_{X_r} + β_{rX_r}
```

application $r \in R$ on R' in descending order of their σ^r (line 4).

2. For each application $r \in R'$, two variables Y_r and X_r are initialized (line 5-7). Y_r stores the minimum service delivery time for application r and X_r tracks the reference of the FCN which delivers the application service in Y_r amount of time. Later, for each FCN $c \in C$, input data propagation time p_{rc}, processing time q_{rc}, service delivery time t_{rc}, Networking resource occupancy α_{rc} and Computing resource occupancy β_{rc} are calculated (line 8-13). Based on these calculations, it is checked whether the service delivery time t_{rc} of application r on FCN c is the least or not (line 14)). Subsequently other constraints noted in Eqs. 5.10, 5.11 and 5.12 are also verified (line 15-17). When all constraints are met, Y_r is updated with t_{rc} and X_r is set to c (line 18-19).

3. For an application $r \in R'$, if X_r refers to an FCN, then r is placed to that FCN. FG g updates its *PlacementMap* for application r and r is added to the set of applications placed on the host FCN (line 20-22). The total Networking and Computing resource occupancy for all placed applications on that FCN are also updated (line 23-24).

Whenever an FG receives placement requests for a set of applications, PlaceApplication procedure is executed. If an application is placed to an FCN, it will not be replaced to other FCNs until any device-level contextual parameter for that application is altered. If any alteration happens, the placement request is relaunched. Thus, the procedure helps stabilized placement of applications. However, from line 5 to 24 of Algorithm 5, there are $\mathcal{O}(|R'| \cdot |C|)$ iterations, where $|R'|$ denotes the number of applications requested to FG g for placement and $|C|$ is the number of accessible FCNs through FG g. If any simplified sorting approach such as binary sorting is used to conduct the operation noted in line 4, then the proposed context-aware application placement policy can function with polynomial time complexity.

5.5 Performance evaluation

The performance of the proposed policy is evaluated through practical and simulation experiments conducted in FogBus-enabled [24] real-world and iFogSim-based [219] simulated Fog environments respectively. In the FogBus-enabled setup, a realistic application case scenario is considered that can assist Industry 4.0. However, in simulated

setup, synthetic workloads in align with this realistic application case scenario are used. The efficacy of our policy is compared with the deadline prioritized [247], resource optimized [248], and service time enhanced [127] application placement policies in both the experimental setup. We have executed the policies separately in a conceptual FG. In deadline prioritized placement, the applications having stringent deadlines are placed faster compared to others. The resource optimized placement reduces the idle time of FCNs during application execution and the service time enhanced placement schedules the applications over FCNs by applying the first-come-first-serve principle and improves their service time. In the following subsections, the application scenario, performance metrics, and the details of both experiments are discussed.

5.5.1 Application Case Scenario

One of the essential aspects of Industry 4.0 is robotic assistance. In smart industries, for emergency management, different sorts of surveillance equipment such as analog and IP cameras are deployed. Image from these cameras are analyzed by Industrial robots to extract the important features of emergency events and take decisions [264]. Since the image quality of analog cameras is ordinary compared to IP cameras, several image processing applications and their services are required to enhance the quality of images before feeding them in robot-embedded image analysis programs. In this work, we consider different image processing applications such as histogram equalization, image noise reduction and linear contrast adjustment as I4OAs [265]. After processing images using these applications, the enhanced images are forwarded to the Industrial robots for further analysis and harnessing robotic assistance in an industry.

5.5.2 Performance Metrics

As performance metrics, Average Service Delivery Time (Avg. SDT), Average Computing Resource Overhead (Avg. CRO) and Average Network Relaxation Ratio (Avg. NRR) are used in the experiments. The reduced value of Avg. SDT denotes the higher potential of application placement policy. Similarly, the decreased value of Avg. CRO refers to the enhanced performance of the policy in managing computing overhead of FCNs.

Figure 5.3: Real experimental setup for context-aware application management

Conversely, the increased value of Avg. NRR signifies that the policy keeps a stable balance between networking load and networking capacity of FCNs. Avg. SDT and Avg. CRO are determined by following Eqs. 5.13 and 5.14 respectively where $\tau = 100$ seconds. Nevertheless, Avg. NRR for an FCN is calculated using Eq. 5.15. Moreover, the Percentage of Deadline Satisfied Inputs (Per. DSI) and Time to Identify the Placement Map (TIPM) are also used here to evaluate the performance of a policy.

$$\text{Avg. SDT} = \frac{\sum\limits_{\forall c \in C} \sum\limits_{\forall r \in Z_c} \sum\limits_{\forall i \in N_r^\tau} t_{rc}^i}{\sum\limits_{\forall c \in C} \sum\limits_{\forall r \in Z_c} |N_r^\tau|} \tag{5.13}$$

$$\text{Avg. CRO} = \frac{1}{|C|} \sum_{\forall c \in C} \frac{\sum\limits_{\forall i \in M_c^\tau} m_c^i}{\tau} \tag{5.14}$$

$$\text{Avg. NRR} = \frac{1}{|C|} \sum_{\forall c \in C} \frac{\lambda^c - \sum\limits_{\forall r \in Z_c} \sigma^r}{\lambda^c} \tag{5.15}$$

5.5.3 Experiments on Real Setup

Fig. 5.3 presents a sample illustration of real experimental setup. Here, different android smart phones are used as the camera-enabled IoT devices. They can capture images in different frequencies and resolution using a customized application. The smart phones are connected to a computer which performs the activities of an FG. Furthermore, we

Duration of experiment : 20 minute				
Number of FCNs : 15 Raspberry PIs				
Configuration of FG:				
Processor	RAM	Clock	Uplink	Downlink
Intel Celeron	2 GB	1.60 GHz	2 MBPS	2 MBPS
FCN type ⇒ Configuration ⇓	Raspberry PI 3 A+	Raspberry PI 3 B+	Raspberry PI 3	Raspberry PI 2
System-on-a-chip	Broadcom BCM2837B0	Broadcom BCM2837B0	Broadcom BCM2837	Broadcom BCM2836
RAM	512 MB	1 GB	1 GB	1 GB
Clock	1.4 GHz	1.4 GHz	1.2 GHz	0.9 GHz
Uplink	2 MBPS	2 MBPS	1.5 MBPS	1 MBPS
Downlink	2 MBPS	2 MBPS	1.5 MBPS	1 MBPS
Amount	3	5	4	3
Workload type ⇒ Attributes ⇓	VGA	HD	FHD	QHD
Resolution (Pixel)	640x480	1280x720	1920x1088	2560x1440
Average size (MB)	0.106	0.230	0.358	0.420
Frequency	4	3	2	1
Deadline (second)	0.240	0.320	0.460	0.700

Table 5.3: Settings of real Fog enviornment for context-aware management

deploy several Raspberry PIs as FCNs to form a Fog Cluster, and execute the image processing applications. The uplink and downlink speed of FG and FCNs are tuned using the Wondershaper software and they are set to follow a linear relationship with the clock speed of corresponding Fog nodes. Table 5.3 presents the details of this setup.

To conduct the experiments in aforementioned setup, we profile the propagation and processing time of all applications on different FCNs for varying inputs. For instance, Table 5.4 shows the profiling information of histogram equalizing application on Raspberry PI 3 B+. These information are directly used by the proposed policy while making the placement decisions for any requested applications. The results of experiments are discussed as follows.

r = histogram equalizing application				
Image type ⇒ FCN $c \in C \Downarrow$	VGA	HD	FHD	QHD
Raspberry PI 3 B+	p_{rc}=0.052 q_{rc}=0.085	p_{rc}=0.118 q_{rc}=0.154	p_{rc}=0.185 q_{rc}=0.226	p_{rc}=0.218 q_{rc}=0.308

Table 5.4: Application profiling information for context-aware management

Impact of Varying the Number of Applications

As the number of applications placed in a certain number of FCNs increases, the Avg. SDT of applications and the Avg. CRO of FCNs elevate (Figs. 5.4 and 5.5). It happens due to simultaneous sharing of resources among various applications. However, the proposed policy sorts applications in descending order of their per unit time data load σ^r that implicitly places heavyweight applications on computationally powerful FCNs. As a consequence, the Avg. SDT of applications for this policy remains in the lower values than others. Moreover, our policy makes a balance between the admittance rate of inputs and their processing on an FCN which helps to improve the Avg. CRO of FCNs. Although the service time enhanced placement shows the similar trend like ours, it often increases the load on computationally powerful FCNs by placing most of the applications over them. Additionally, the deadline prioritized placement often leads the latency tolerant applications with higher σ^r to be executed in less computationally powerful FCNs. Both degrade the Avg. SDT of applications and Avg. CRO of FCNs. On the other hand, while dealing with the applications having higher frequency of data, the resource optimized placement significantly increases the computing overhead of FCNs and affects the Avg. SDT and Avg. CRO.

The proposed policy also explicitly measures the effect of data sensing frequency of IoT devices on the networking capacity of FCNs during application placement. Consequently, it helps to offer improved Avg. NRR than other policies where such analysis is narrowly attended. Fig. 5.6 depicts this aspect of our proposed policy for the increasing number of applications in Fog computing environments.

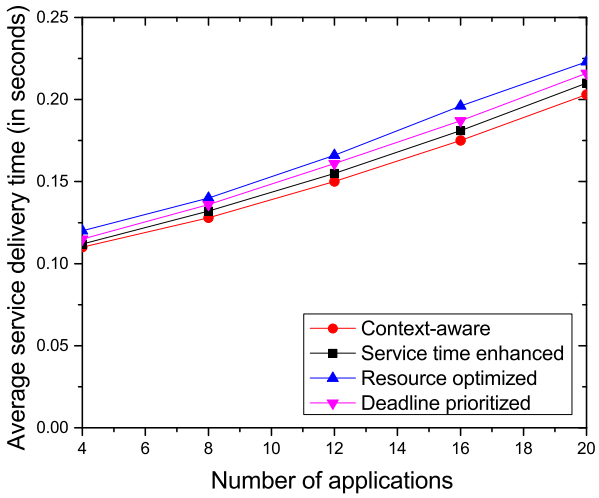

Figure 5.4: Average service delivery time for varying number of applications

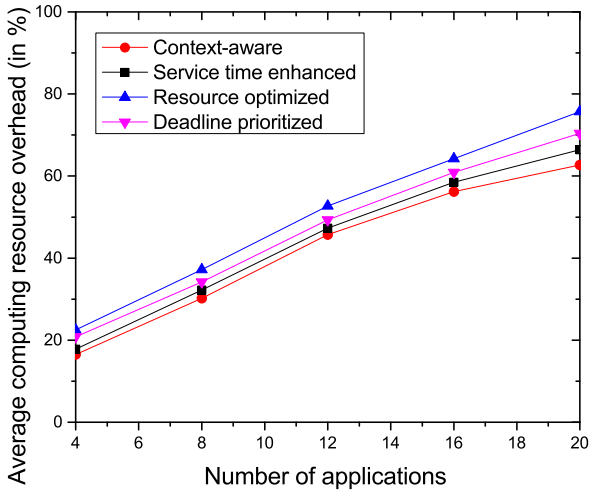

Figure 5.5: Average resource overhead for varying number of applications

Impact of Varying the Number of FCNs

With the increasing number of FCNs, Avg. SDT of applications and Avg. CRO of FCNs decreases, and Avg. NRR increases (Figs. 5.7, 5.8 and 5.9). For higher number of FCNs, most of the policies exhibit similar trend. However, when there are comparatively lower number of FCNs for application placement, the proposed policy outperforms others. It places applications on limited number of FCNs considering input data size and data sensing frequency of associated IoT devices simultaneously that consequently meets

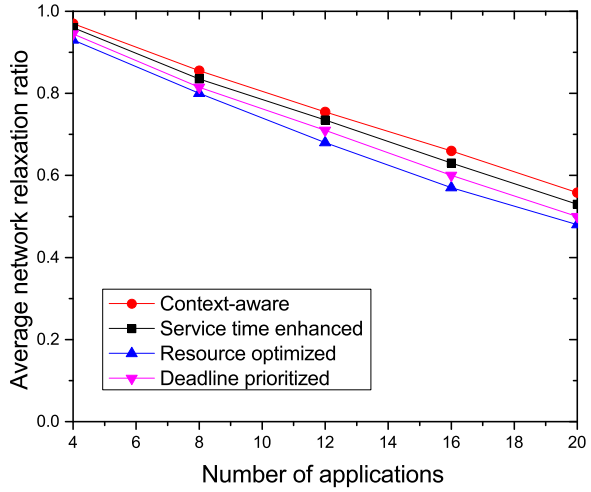

Figure 5.6: Average network relaxation for varying number of applications

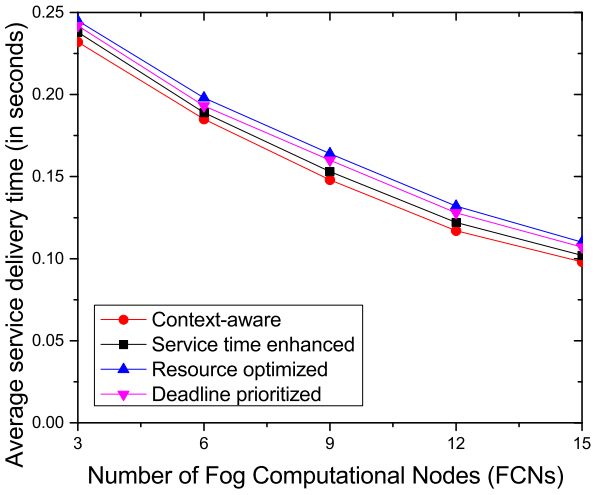

Figure 5.7: Average service delivery time for varying number of Fog nodes

computational and networking commitment of FCNs with their capacity. Thus, despite of having lower number of options for placing applications, computing overhead of FCNs and their networking load do not increase significantly and service delivery time of applications remain in lower values.

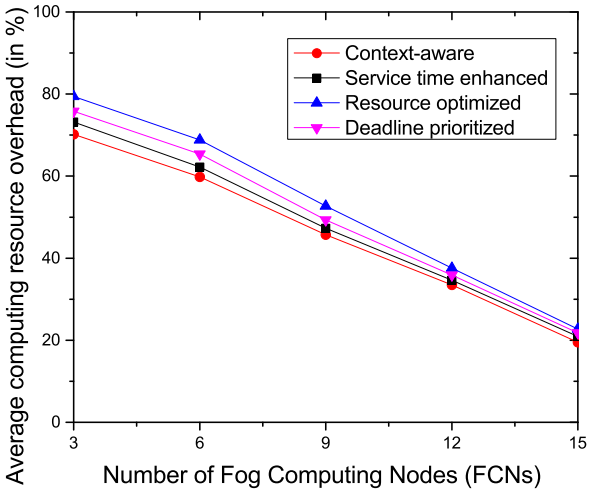

Figure 5.8: Average resource overhead for varying number of Fog nodes

Observation on Stream Processing

In our evaluation, we monitor application service delivery time for a set of inputs belonging to a particular stream (Fig. 5.10). In context-aware application placement, service delivery time of inputs do not vary significantly from one to another. Since the proposed context-aware application management reduces the scope of computing overhead and network congestion, the service times for different inputs of a stream remain almost same.

5.5.4 Experiments in Simulated Setup

Different parameters used in modelling the simulated setup are listed in Table 5.5. The workload attributes are aligned with the specification of real experimental setup as shown in Table 5.3. Additionally, the configuration of FCNs are set according to the processor benchmarking data provided in [266]. Linear relations are maintained among the parameters of different inputs and FCNs while setting their values from the given range. The simulation experiments are conducted on an *Intel Celeron, 1.60 GHz, 2 GB RAM* configured computer. The results of these experiments are discussed below.

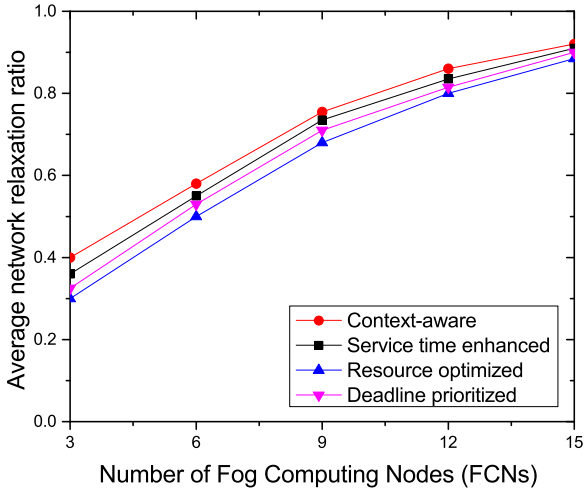

Figure 5.9: Average network relaxation for varying number of Fog nodes

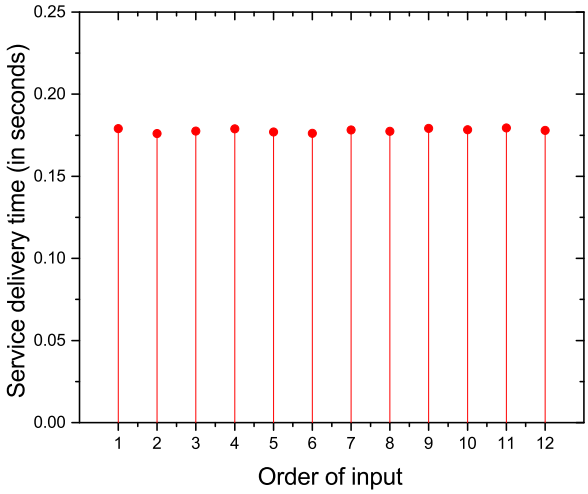

Figure 5.10: Service delivery time for different inputs of a stream

Impact of Varying the Arrival Rate of Requests

As the arrival rate of placement request increases, the Per. DSI decreases (Fig. 5.11). It happens because of the rising number of applications waiting in the queue for execution. However, the deadline prioritized placement performs better in this case as it immediately executes the deadline-critical applications. Compared to the service time enhanced placement, our proposed policy offers improved Per. DSI as it does not increase the communication and computation overhead of FCNs unevenly. Conversely, the intention of

Parameter	Value
Simulation time	200 Seconds
Sensing duration of IoT devices	1-3 Seconds
Arrival rate of placement requests	10-30 requests/second
Number of FCNs	50
Configuration of FCNs:	
Processing speed	1000-4500 MIPS
RAM	1-2 GB
Downlink bandwidth	1-3 MBPS
Uplink bandwidth	1-3 MBPS
Workload attributes:	
Number of instructions	100 - 1200 MI
Input data size	0.120-0.500 MB
Service deadline	0.300-0.700 seconds
Sensing frequency of IoT devices	1-4 input/second

Table 5.5: Simulation parameters for context-aware management

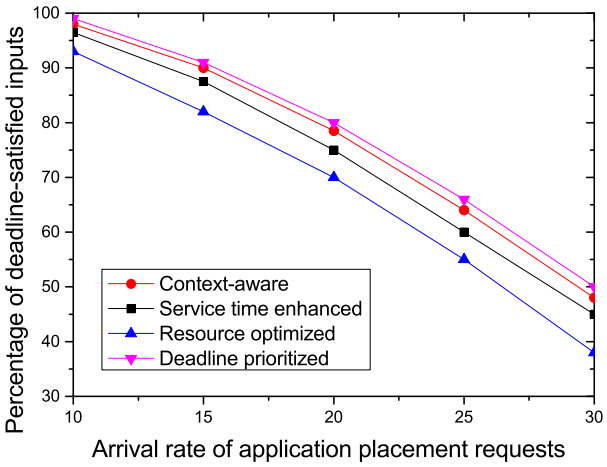

Figure 5.11: Percentage of deadline satisfaction for varying request arrival rate

reducing FCN's idle time often lead the resource optimized placement to disregard the deadline criticality of applications. As a result, Per. DSI degrades remarkably for this policy.

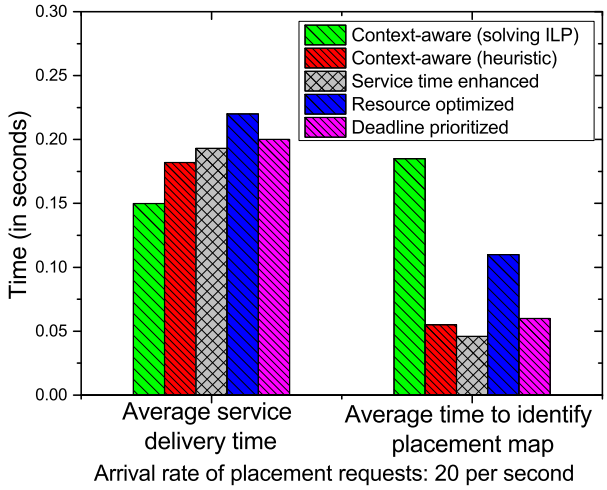

Figure 5.12: Average service and placement time for different placement policies

Comparison between Solution Approaches

The proposed context-aware application placement can be performed either by solving the optimization problem in Eq. 5.8 through any ILP solver or applying the heuristic method of Algorithm 5. Avg. SDT of applications in ILP-based approach is always lower than the heuristic implementation of proposed policy. However, at any arrival rate of placement requests, the heuristic implementation takes less TIPM compared to the ILP-based approach. Fig. 5.12 depicts such a scenario when the request arrival is 20 per second. Since the operations of heuristic implementation and the deadline prioritized placement are almost similar, their TIPM does not very significantly. The TIPM of service time enhanced policy is lower than others as it does not include any additional sorting operations. Conversely, the resource optimized placement poses high TIPM because of its extensive search for suitable applications that can reduce the idle time of FCNs.

5.6 Summary

I4OAs require to offer services in real-time. During the execution of I4OAs, the contexts of IoT devices such as their data size and sensing frequency play influential roles in defining the computing and networking intensity of the applications and help in measuring the processing and communication load of the host Fog nodes. Failure to support

them with the capacity of Fog nodes can degrade the application service time. Therefore, in this work, we proposed a policy that implies the IoT device-level contexts to place the applications in Fog environments. It also ensures that the flow of data towards the applications neither increases overhead of Fog nodes nor congests the network deliberately. Thus, it reduces application's service delivery time, relaxes network, manages computing overhead, and increases service reliability. Additionally, the experimental results derived from both real and simulated Fog environments demonstrate the efficiency of our policy.

In last three chapters, we discuss a framework that distributes the application management tasks across the gateway and the infrastructure level of Fog environments, a procedure to select applications for Fog-based placement, a latency-aware approach for placing distributed applications over Fog nodes, a module forwarding strategy to optimize the number of Fog nodes, a context-aware approach to optimize the service delivery time of applications and a strategy to manage the network congestion and computation overhead of Fog nodes are discussed. These concepts collectively ensure QoS satisfied service delivery for different types of applications in Fog environments.

Chapter 6

Quality of Experience (QoE)-aware Application Management

Placement of applications to compatible Fog instances based on user expectations can enhance Quality of Experience (QoE) regarding the system services. In this chapter, 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.

6.1 Introduction

To attain certain service level objectives in Fog, different application placement policies are required. Quality of Service (QoS) [267][158], resource [268], situation-aware [85] 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.

This chapter is derived from:

- **Redowan Mahmud**, Satish Narayana Srirama, Ramamohanarao Kotagiri, and Rajkumar Buyya, "Quality of Experience (QoE)-aware Placement of Applications in Fog Computing Environments", *Journal of Parallel and Distributed Computing (JPDC)*, Volume 132, Pages: 190-203, ISSN: 0743-7315, Elsevier Press, Amsterdam, The Netherlands, October 2019.

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 [269]. 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 [270] and resource estimation [271]. Additionally, the consideration of QoE while placing applications can help in improving the 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) [271][272] 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 [273] [274] 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 QoE does not degrade. Thus, discrepancy between user feedback and QoE on specific service attribute can be monitored.

In this chapter, 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 this work 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* [237]. The experimental results show significant improvement in QoE enhancement compared to other QoE and QoS-aware policies.

The rest of the chapter is organized as follows. In Section 6.2, relevant research works are reviewed. In Section 6.3 and 6.4, the motivation and the addressed problem of this research are discussed. Section 6.5 represents the system overview and assumptions. The proposed QoE-aware application placement policy and an illustrative example are described in Section 6.6 and 6.7 respectively. Section 6.8 enlightens the simulation environment and the performance evaluation. Finally Section 6.9 concludes the chapter.

6.2 Related Work

A summary of several QoS/QoE-aware application management policies in different computing paradigms is shown in Table 6.1. Mahmud et al. [240] proposed 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.

Work	Observes User Expectations in			Meets Instances Status regarding			Prioritized Placement	Compound QoE Gain
	Service Access	Resource Requirement	Processing Time	Proximity/ Response Rate	Resource Availability	Processing Speed		
[240]	✓		✓			✓	✓	✓
[275]	✓			✓	✓		✓	
[276]	✓	✓		✓	✓			✓
[277]			✓	✓		✓		
[278]	✓		✓	✓				
[279]			✓		✓	✓	✓	
[267]		✓	✓		✓	✓	✓	
[158]		✓		✓		✓		
[270]	✓		✓	✓		✓	✓	
[271]		✓			✓		✓	
[280]	✓			✓	✓		✓	
QoE-aware (This work)	✓	✓	✓	✓	✓	✓	✓	✓

Table 6.1: Summary of related work for QoE-aware management

Zhou et al. [275] proposed 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. [276] proposed 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.

A QoE-aware transcoding policy for MEC is discussed in [277]. 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 in [278]. 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. [279] proposed a QoE-optimized scheduler for multi-class system (e.g. web interactive, file downloads, etc.) in wireless networks. Their policy models QoE as a cost function of 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 scheduler is an extension of Gittin index scheduler.

In Fog computing paradigm, different QoS and QoE-aware application management policies are also studied. A Fog service placement problem is formulated in [267] that targets QoS-aware application placement on virtualized Fog resources. It considers 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 in [158]. 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.

Three factors (response time, network congestion, service coverage) that dominate user's QoE while playing interactive games are identified in [267]. It discusses a lightweight system named *Cloud-Fog* to extend service coverage for the 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 in [271]. 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.

A model for Fog-based Internet access networks that assist dynamic placement of Cloud or Web content at the edge networks have been developed in [280]. 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 cannot address individually.

6.3 Motivation and Requirements

6.3.1 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 users [281]. Conversely, ITU defines QoE as the total acceptability of a service that is determined by subjective perception of the users [282]. 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 [283].

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 [284]. 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, the perceived QoE can degrade the acceptability of a service greatly even when the QoS regarding the service is maintained [285]. It actually happens due to diversified characteristics of the users. Extending the aforementioned example, let's assume, on fixed charge, the Internet-enabled system pro-

visions 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.

6.3.2 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 [286], 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 [287]

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, cannot 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.

6.4 Problem Description

6.4.1 Exploration of Expectation and Status Metrics

From the discussion of Section 6.3, 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 sig-

nals 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 [20]. 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.

6.4.2 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.

6.5 System Overview

6.5.1 Application Model

Fog-enabled IoT applications are usually divided into multiple interconnected Application Modules [237]. The module running at the end-user 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 [220] or offloaded from the end-user devices due to resource constraints [288]. 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 chapter, by the term "application" we refer to the Main Application Module of Fog-enabled IoT systems.

6.5.2 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. 6.1. Here, Fog nodes are classified into two categories; *Fog Gateway Nodes (FGNs)* and *Fog Computational Nodes (FCNs)* [289].

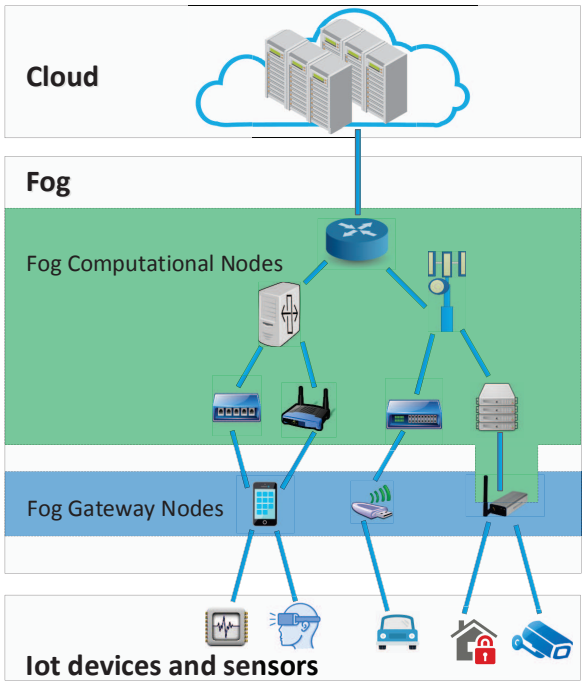

Figure 6.1: Organization of Fog environments

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 analysing the data signals. According to OpenFog Consortium [20],

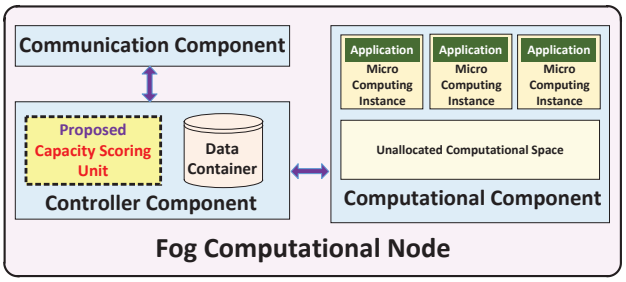

Figure 6.2: Architecture of a Fog computational node

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 [290] [291], 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 [235].

However, the distance between Fog node and IoT devices in hierarchical setting is reflected through the round-trip delay of the data signals. The computation capability and resource availability of lower level Fog nodes are less than 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.

6.5.3 Architecture of Fog nodes

Fog Computational Nodes (FCNs)

Recently, the OpenFog Consortium has proposed a reference architecture for Fog nodes where the computation, management and networking operations are conducted on discrete components [20]. Based on the reference architecture, we assume that an FCN is

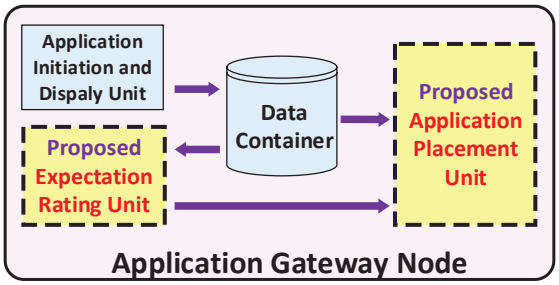

Figure 6.3: Architecture of a Fog gateway node

composed of *Controller Component*, *Computational Component* and *Communication Component* (Fig. 6.2).

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 (MCIs)*, where the applications are assigned for execution [55]. 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 an 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.

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 [292], a general architecture of FGNs is represented in Fig. 6.3. 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 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 6.2.

6.6 QoE-aware Application Placement

The flowchart of the proposed QoE-aware application placement policy is depicted in Fig. 6.4. 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.

6.6.1 Calculation of Rating of Expectation (RoE)

After subscribing to an 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

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 $a \in A_m$.
v_{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.

Table 6.2: Notations for QoE-aware management

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. 6.1:

$$\overline{U_x^{a_m}} = 2\left(\frac{U_x^{a_m} - \alpha_x}{\beta_x - \alpha_x}\right) - 1 \tag{6.1}$$

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. 6.5.

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. 6.6. 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.*

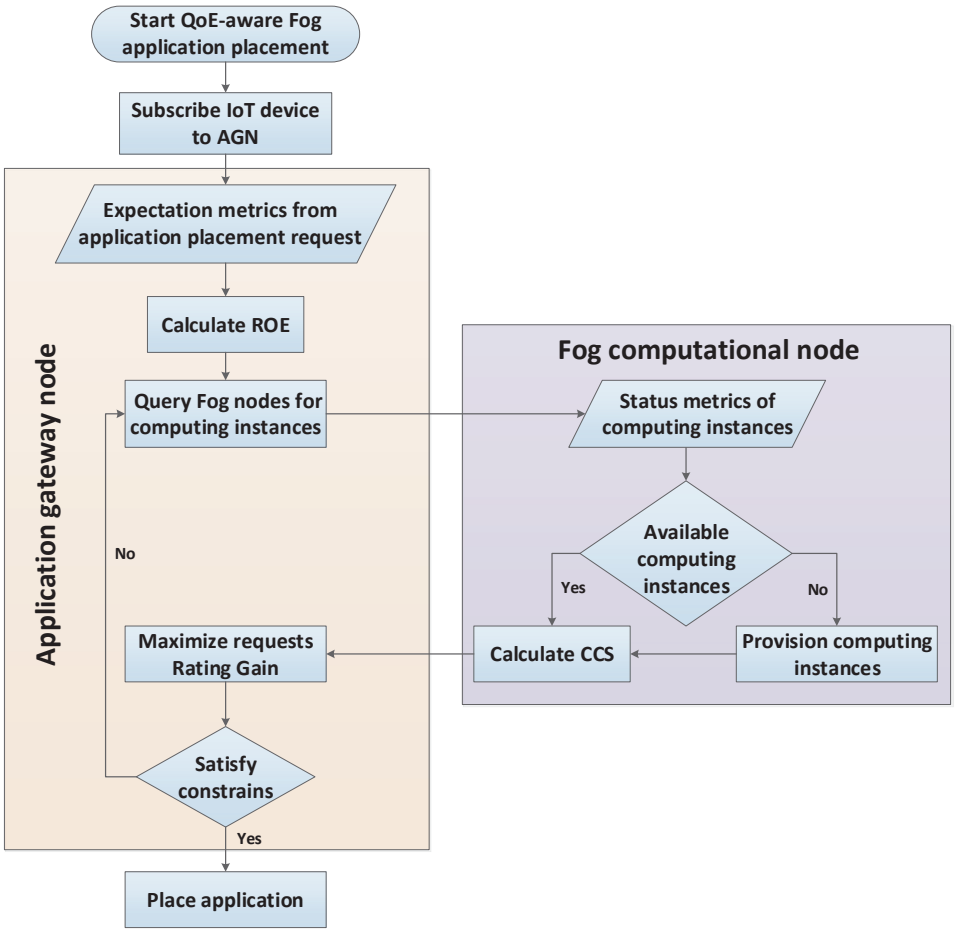

Figure 6.4: Flowchart for QoE-aware management

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 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

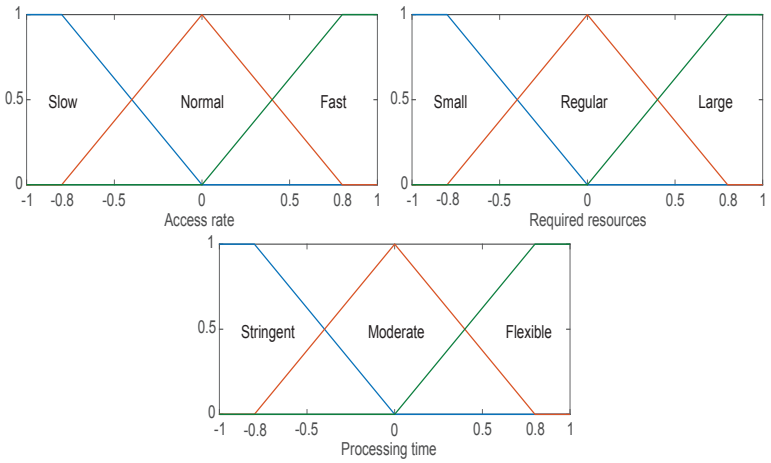

Figure 6.5: Membership function for expectation metrics

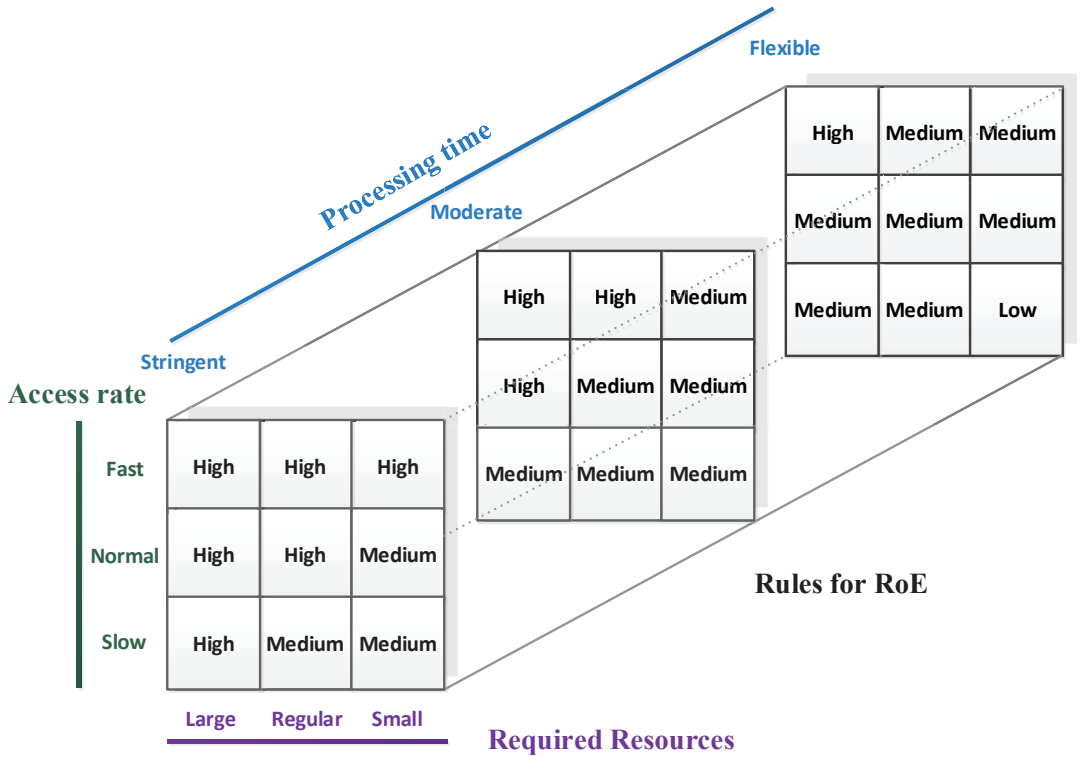

Figure 6.6: Fuzzy rules for RoE calculation

degree of fuzzy output, $\mu_r(f_{a_m})$ is determined using Eq. 6.2:

$$\mu_r(f_{a_m}) = max(\mu_\omega(\overline{U_\omega^{a_m}}), \mu_\gamma(\overline{U_\gamma^{a_m}}), \mu_\lambda(\overline{U_\lambda^{a_m}})) \tag{6.2}$$

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. 6.3.

$$\eta_{a_m} = \frac{\sum_{k=1}^{k=j} \mu_r(f_{a_m}^k) \times \phi_k^{f_{a_m}}}{\sum_{k=1}^{k=j} \mu_r(f_{a_m}^k)} \tag{6.3}$$

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.

6.6.2 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 an 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. 6.4 different parameter y of S_{i_n} have been normalized to $[-1, 1]$.

$$\overline{V_y^{i_n}} = 2\left(\frac{V_y^{i_n} - \alpha'_y}{\beta'_y - \alpha'_y}\right) - 1 \tag{6.4}$$

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,

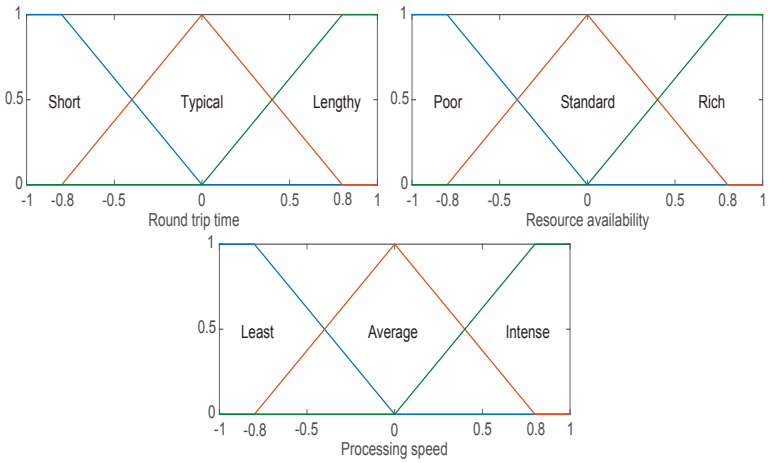

Figure 6.7: Membership function for status parameters

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. 6.7.

The fuzzy rules applied for determining the fuzzy output for CCS calculation is represented in Fig. 6.8. 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

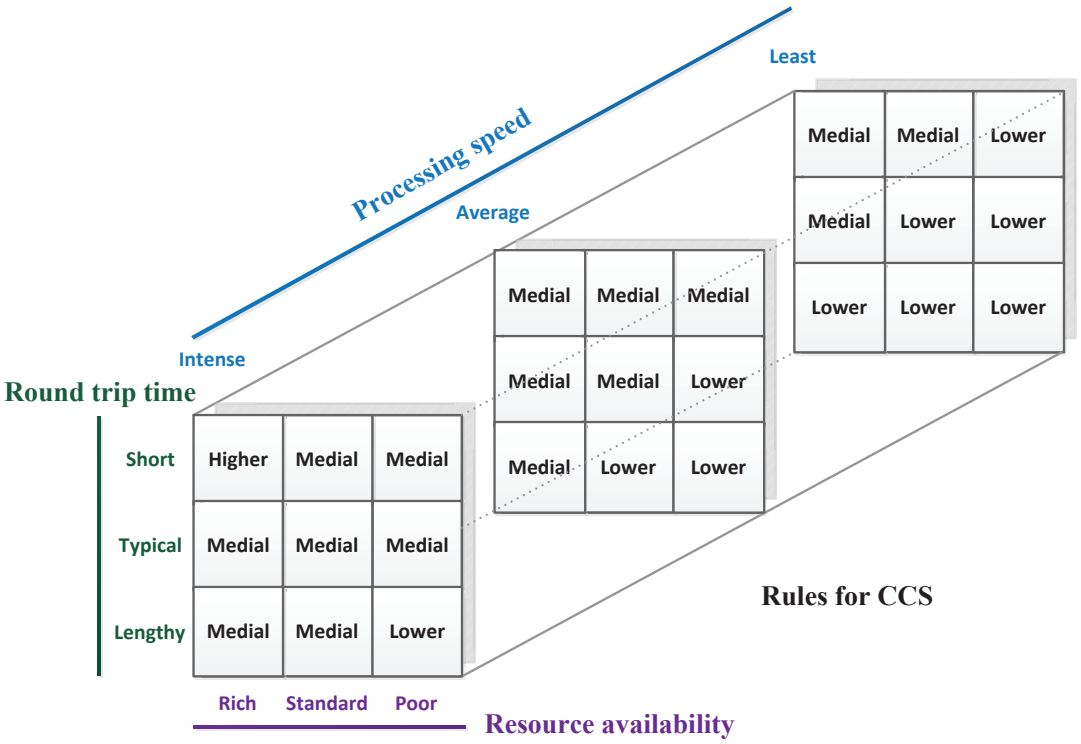

Figure 6.8: Fuzzy rules for CCS calculation

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 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. 6.5:

$$\mu'_c(f'_{i_n}) = min(\mu'_\Omega(\overline{V_\Omega^{i_n}}), \mu'_\Gamma(\overline{V_\Gamma^{i_n}}), \mu'_\Lambda(\overline{V_\Lambda^{i_n}})) \tag{6.5}$$

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. 6.6.

$$\tau_{i_n} = \frac{\sum_{k=1}^{k=j} \mu'_c(f'^k_{i_n}) \times \Phi_k^{f'_{i_n}}}{\sum_{k=1}^{k=j} \mu'_c(f'^k_{i_n})} \tag{6.6}$$

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.

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

$$\max \sum_{\forall a_m \in A_m} \sum_{\forall n \in N} \sum_{\forall i_n \in I_n} z_{i_n}^{a_m} (\eta_{a_m} \times \tau_{i_n}) \tag{6.7}$$

subject to,

$$\sum_{a_m \in A_m} z_{i_n}^{a_m} = 1; \forall n \in N, \forall i_n \in I_n \tag{6.8}$$

$$\delta_{a_m} \leq Q_\delta \tag{6.9}$$

$$\zeta_{a_m} \leq Q_\zeta \tag{6.10}$$

$$\rho_{a_m} \leq Q_\rho \tag{6.11}$$

The objective function in Eq. 6.7 maximizes the Rating Gain for all application placement requests received by the FGN that subsequently enhances the overall user QoE. The constraint in Eq. 6.8 ensures one to one mapping between applications and instances. Besides, constraints in Eqs. 6.9, 6.10 and 6.11 maintains the QoS of the application in terms of service delivery time, service cost and packet loss rate respectively. If an 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 an FGN, the optimization problem is solved and placement of the applications are conducted. By using any integer programming solver e.g. SCIP [215], 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 localized operation, the probability of receiving large number of application placement requests by an FGN on a particular time is low. Therefore, the solution to this optimization problem can be found in a reasonable amount of time.

6.6.4 Rationality of the Applied Techniques

In this chapter, 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 [293]. 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 [294].

After determining RoE and CCS of application placement request and Fog instances, 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 [215].

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 [295]. 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 [296]. Considering these challenges of multi-objective optimization, we rather preferred to apply Fuzzy logic and single objective optimization in our proposed policy.

6.7 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. 6.9.

In this Fog environment, at any time t, the FGN m receives five application placement requests with $v_{a_m} = 1000 \sim 2000$ instructions ($\forall a_m \in A_m$). The scope in the Fog for different expectation parameters of application placement requests is represented in Table. 6.3

The exact expectation parameters of the requests along with normalized values, degree of membership to different fuzzy sets and RoE are shown in Table. 6.4. Here, the

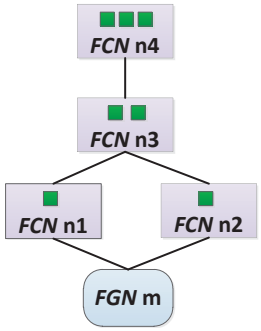

Figure 6.9: Illustrative Fog environment

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

Table 6.3: Scope of expectation parameters

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 6.6.1.

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. 6.5.

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. 6.6. 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 6.6.2.

By applying Eq. 6.7 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. 6.7. Here the constraints (Q_δ = 250∼750 ms, Q_ζ = 0.1∼0.15 $ per min, Q_ρ = 3∼5 % data signals) are as-

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

Table 6.4: Parameters of application placement requests

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

Table 6.5: Scope of status parameters

sumed to be met. Based on the optimization solution, after exploring the expectation and the status parameters (from Tables. 6.4 and 6.6) 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

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

Table 6.6: Parameters of computing instances

literature [297][298]. The values explicitly 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

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

Table 6.7: Solution of the optimization problem

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.

6.8 Performance Evaluation

The proposed QoE-aware application placement policy is compared with different QoS and QoE-aware policies. The QoS-aware application placement policy in [267] meets execution deadline of the applications. Among the QoE-aware policies, Cloud-Fog [270] optimizes service coverage, response time and network congestion whereas MeFoRE [271] 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.

6.8.1 Simulation Environment

To evaluate the proposed policy, a Fog environment is simulated using iFogSim [237]. iFogSim is an extention of *CloudSim* [238] 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

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

Table 6.8: Simulation parameters for QoE-aware management

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 6.8 represents the details of workload and system parameters.

6.8.2 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. 6.12:

$$avg(NRR_m) = \frac{1}{|A_m^t|} \sum_{\forall a_m \in A_m} \frac{2}{U_\omega^{a_m} \times V_\Omega^{i_n}} \tag{6.12}$$

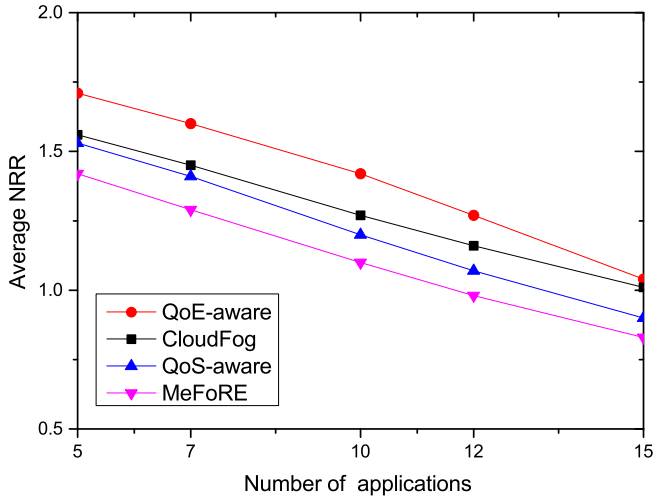

Figure 6.10: Network relaxation for varying number of applications

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

Any value of $NRR > 1$ for an application refers to less possibility of network congestion. For example, an application with $U_{\omega}^{a_m} = 2$ 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 $V_{\Omega}^{i_n} = 0.3$ 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. 6.10). As the number of applications increase, the data transmission load over the network increases and $avg(NRR)$ declines, which is natural. However, compared 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.

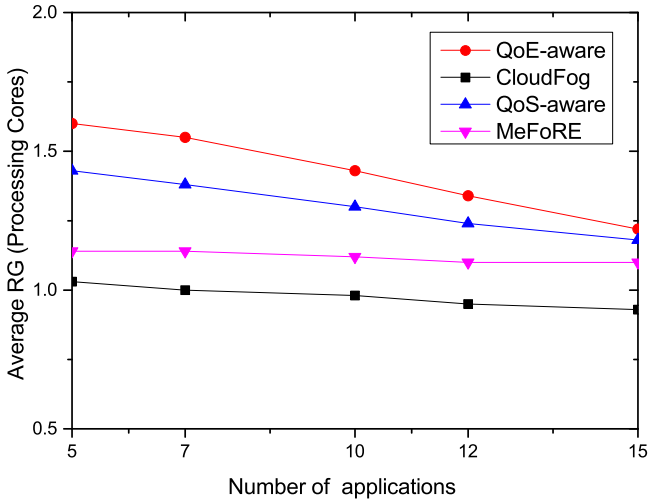

Figure 6.11: Resource gain for varying number of applications

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. 6.13:

$$avg(RG_m) = \frac{1}{|A_m^t|} \sum_{\forall a_m \in A_m} \frac{V_\Gamma^{i_n}}{U_\gamma^{a_m}} \tag{6.13}$$

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. 6.11). 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.

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. 6.14:

$$avg(PTRR_m) = \frac{1}{|A_m^t|} \sum_{\forall a_m \in A_m} \frac{U_\lambda^{a_m} \times V_\Lambda^{i_n}}{v_{a_m}} \tag{6.14}$$

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 $v_{a_m} = 1000$ 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. 6.12). 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.

Fig. 6.13 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 optimization problem for increasing number of 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

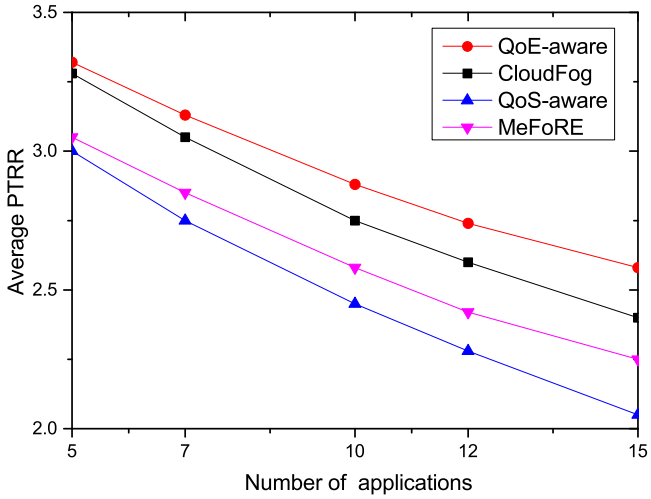

Figure 6.12: Processing time reduction for varying number of applications

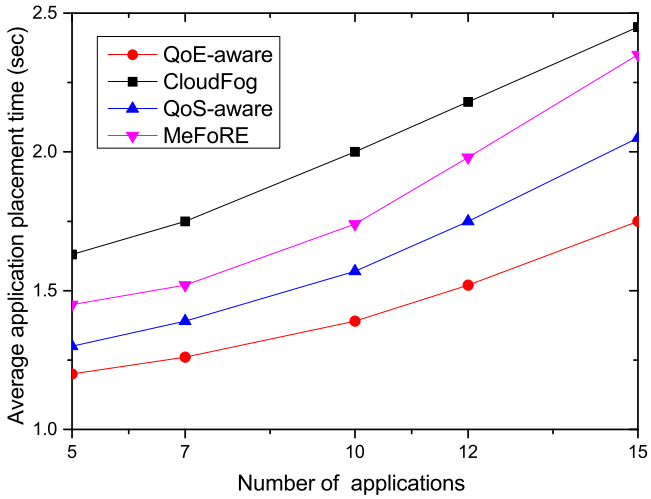

Figure 6.13: Application placement time for varying number of applications

placing the applications (Fig. 6.14). In QoS-aware policy only deadline has been considered as the QoS metric. In that policy, for many data signals the deadline satisfied QoS cannot 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.

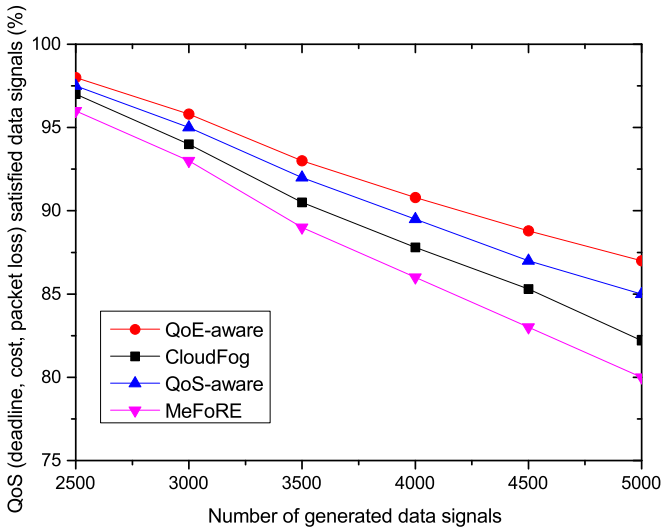

Figure 6.14: Percentage of QoS satisfaction

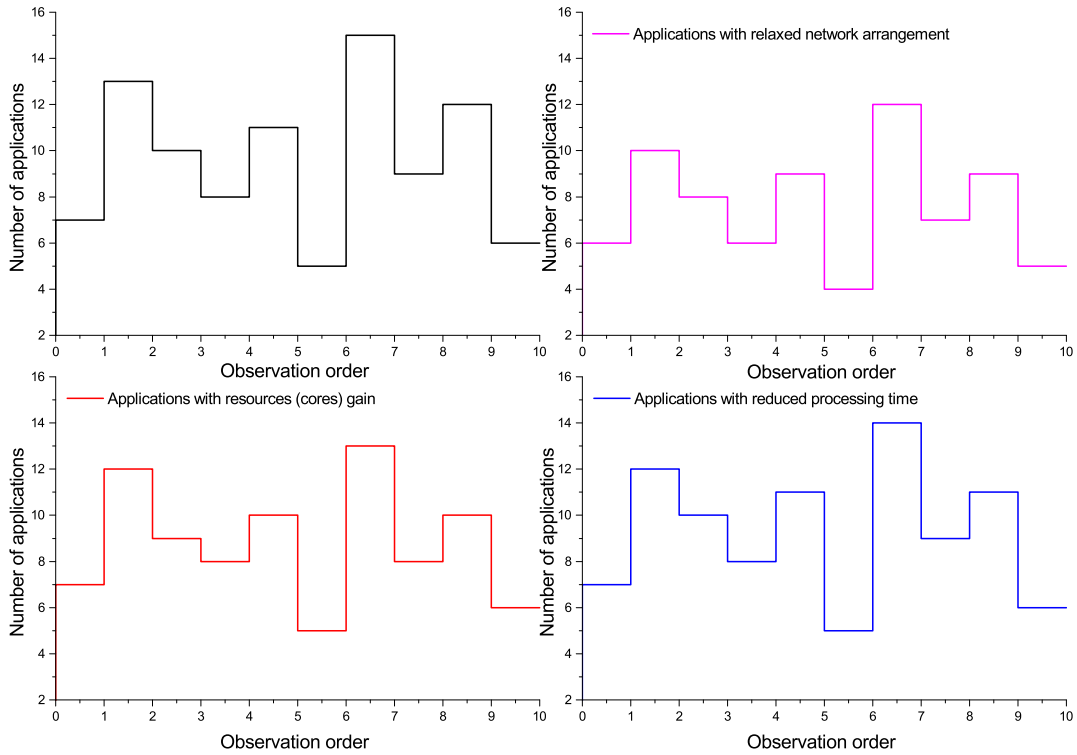

Figure 6.15: QoE Gain of applications from different service aspects

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 application. Fig.6.15 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 and NRR, this percentage belongs to 88% and 80% respectively.

6.9 Summary

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.

Chapter 7

Profit-aware Application Management

The integration of Fog and Cloud paradigm aims at harnessing both edge device and remote datacentre-based computing resources to meet the Quality of Service (QoS) requirements of various smart systems. Due to lack of instance pricing and revenue maximizing techniques, it becomes difficult for service providers to make comprehensive profit from such integration. Conversely, the rigid revenue maximizing intention of providers affects user's budget and system's service quality. To address these issues, we propose a profit-aware application placement policy for integrated Fog-Cloud environments. It simultaneously enhances profit and ensures QoS. Furthermore, it provides compensation to users for any violation of Service Level Agreement (SLA) and sets the price of instances according to their ability of reducing service delivery time. The performance of proposed policy is evaluated in a simulated Fog-Cloud environment using iFogSim and the results demonstrate that it outperforms other placement policies in increasing provider's profit and user's QoS satisfaction rate.

7.1 Introduction

Fog nodes have less computational capabilities than Cloud datacentres that resist accommodation of every IoT application at the edge level [55]. Therefore, different Cloud providers such as Amazon, Microsoft and Google initiate integrating Fog and Cloud infrastructure to offer extensive placement options for IoT applications [13]. The in-

This chapter is derived from:

- **Redowan Mahmud**, Satish Narayana Srirama, Kotagiri Ramamohanarao, and Rajkumar Buyya, "Profit-aware Application Placement for Integrated Fog-Cloud Computing Environments", *Journal of Parallel and Distributed Computing (JPDC)*, Volume 135, Pages: 177-190, ISSN: 0743-7315, Elsevier Press, Amsterdam, The Netherlands, January 2020.

clusion of Fog computing to current Cloud-centric IoT model is expected to add US$ 203.48 million more in their combined marketplace by 2022 [299]. It will also increase the operational cost in computing environments for consuming additional energy, deploying Fog infrastructure and utilizing more network bandwidth [300]. In this case, without revenue maximization, it will be difficult for providers to make profit from integrated environments. Contrariwise, firm intention of maximizing revenue often instigates providers to compromise application Quality of Service (QoS) that increases Service Level Agreement (SLA) violations. The imprecise price of Fog instances that is set for revenue maximization can also add overhead to user's budget [301]. Hence, it becomes challenging to enhance provider's profit in integrated environments as it urges to make a balance among expectations of users, expenses of providers and performance of Fog-Cloud infrastructure. Failure to ensure such balance inhibits providers and users to realize the potential of integrated computation [302].

In integrated environments, placement of applications on suitable instances is very crucial to enhance profit of providers and meet application QoS for users. Although different application placement policies for Fog computing are proposed prioritizing deadline, completion time and revenue [207] [177] [303], it is critical for these policies to attain the aforementioned objectives individually for integrated environment. Diversified affordability level of users, uneven expenses of operating heterogeneous instances and commitment of providing compensation to users for service failure further intensify the complexity of such application placement problem [2]. Therefore, it is demanding to develop an application placement policy for integrated Fog-Cloud environments that can comply with their economic and performance-based attributes simultaneously.

In Internet economics, providers are encouraged to charge users more for improved services [212]. Since Fog instances upgrade application service delivery time, it creates a scope for providers to charge users an extra amount for these instances on top of their actual Cloud-based price. To users, providers can advertise this additional charge as the price for extending the instance from Cloud to Fog infrastructure. However, it should be justified with the scale of performance improvement and user budget constraint. It is also required for clarifying the impreciseness of instance pricing and assisting users to identify how much they need to pay for executing applications in Fog. Additionally, to

attain loyalty, providers can offer compensation to users on SLA violations. With such instance pricing model and compensation method, an application placement policy in integrated environments can boost the revenue and arouse the necessity of meeting application QoS that will consequently enhance provider's profit. However, in existing works such policy has not been explored yet. Therefore, we propose a profit-aware application placement policy for integrated Fog-Cloud environments that increases revenue and reduces their number of failures in meeting application's service delivery deadline. It also sets price of Fog instances in accordance with their capabilities of improving service quality and provides compensation to users based on SLA violation rate of computing environments.

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

- Proposes an application placement policy for integrated Fog-Cloud environments based on an *Integer Linear Programming (ILP)* model that enhances provider's profit and meets application's QoS simultaneously.
- Presents a pricing model for Fog instances which increases provider's revenue by incorporating their Cloud-based pricing with the service delivery time improvement ratio of applications placed on those instances.
- Develops a user compensation method that supports both user's and provider's interest through inverse relationship between compensation amount and performance of the computing environments in observing SLA requirements.
- Demonstrates the performance of proposed policy in enhancing profit, satisfying QoS and managing waiting time via simulation on *iFogSim* [219] and compares them with the outcomes of existing policies.

The rest of the chapter is organized as follows. Section 7.2 highlights several relevant works form literature. Section 7.3 provides the architecture of integrated environments along with revenue estimation, pricing model and compensation method. The proposed application placement policy and its illustrative example are presented in Section 7.4 and 7.5 respectively. Section 7.6 presents the simulation environment and performance evaluation of the proposed policy. Finally, Section 7.7 concludes the chapter.

7.2 Related Work

Provider's profit and cost maintenance have already been studied extensively in Cloud computing [304] [305]. However, Fog computing is different from Cloud as it is more distributed and composed of numerous resource-constrained and heterogeneous Fog nodes. Service expectations of users from Fog-based applications, their anticipated runtime and budget for execution are also diversified compared to that of Cloud-based applications. Therefore, it is very complicated to develop interoperable resource and application management policies for both Fog and Cloud computing and tedious to customize any existing Cloud policy for Fog computing [2]. Nevertheless, there exists several works that discuss about financial aspects of integrated Fog-Cloud environments. Nan et al. [207] provided an online solution that minimizes task completion time and provider's cost in integrated environments. It also reduces overall response time by discarding infeasible applications directly from the queue. A trade-off is made between power consumption and transmission delay in [131]. It solves the placement problem distributively and allocate resources at the Fog to complement Cloud for improving performance. Moreover, Pham et al. [306] conducted a trade-off between execution time and cost of Cloud-based processing while placing applications in integrated environment. Their Cost-Makespan-aware placement policy meets application deadline constraints. Yu at al. [199] also focused on reducing processing cost in Cloud by placing applications in Fog. Their policy saves bandwidth cost by serving users with Fog resources and compensates Fog providers for processing data on behalf of Cloud.

The efficacy of Fog has also been extended to other computing paradigms. Lin et al. [252] minimized the expenses in Fog assisted Cyber Physical System (CPS) considering instance deployment, data uploading and inter-nodal communication cost. To overcome high complexity, their policy linearizes the cost-minimization problem then solves it through a two-phase linear program-based heuristic algorithm. Likewise, Yang et al. [307] explored cost-efficient service placement and load distribution in Fog enabled Mobile Cloud Computing (MCC) environments. Their algorithms make trade-off between the average response delay and the expenses of providers by considering mobility and service access pattern of users. Yao et al. [308] considered instance deployment

Work	Decentralised Decision	Provides Compensation	Considers Budget	Maintains QoS	Enhances Profit	Models Price	Targets Cost	
							Fog	Cloud
[207]	✓			✓			✓	✓
[131]	✓			✓			✓	✓
[306]				✓				✓
[199]	✓	✓						✓
[300]	✓			✓			✓	
[307]				✓			✓	
[308]				✓			✓	
[309]			✓		✓		✓	
[177]	✓			✓	✓		✓	
[303]	✓	✓	✓		✓	✓	✓	
[200]	✓	✓			✓	✓	✓	
[310]	✓		✓	✓			✓	
Profit-aware (This work)	✓	✓	✓	✓	✓	✓	✓	✓

Table 7.1: Summary of related work for profit-aware management

cost and diverse mobility pattern of the users while placing applications on heterogeneous Fog nodes (Cloudlets). Their greedy solution, at first, generates candidate set of Cloudlets that meets user's requirements, then selects a Cloudlet from the candidate set to place the applications with minimum deployment cost. Kiani et al. [309] proposed an auction-based profit maximization policy for Fog enabled Mobile Edge Computing (MEC) environment. Their policy is developed on a binary linear programming model and incorporates a two-time scale technique while allocating both the computing and communications resources to the mobile users.

In literature, profit and budget-aware resource estimation for Fog computing are also studied. Fan et al. [177] discussed deadline-aware application placement that enhances provider's profit and user's QoS satisfaction. It exploits provider's owned Fog resources and rented Cloud instances. Neetu et al. [303] explored the competition between Fog providers in setting service price and minimizing their cost through Equilibrium Con-

straints model. It aims at enhancing the profit and balancing the service requirements between providers and users by facilitating incentivization. A dynamic resource estimation and pricing policy for Fog computing was developed in [200]. While allocating resources and charging services, their policy considers user's behavior, provider's profit and mobility pattern of IoT devices. Ni et al. [310] proposed a resource provisioning policy that enables users to meet demand from a set of reserved resources. It considers cost and deadline along with user's affordability and features of Fog nodes during resource allocation.

Table 7.1 provides a summary of different application placement policies that investigate monetary issues and service objectives of integrated environments. In these works, enhancement of profit and maintenance of QoS are not simultaneously ensured during application placement. They barely apply performance-driven instance pricing model and compensation method to promote provider's revenue and subsidize user's losses. In comparison, the proposed policy contains two important features; a). sets price of Fog instances based on their competency of improving application's service time and b). offers compensation according to the SLA-violation rate of computing environments. Both jointly offer a systematic way to support user's and provider's interests. The profit-aware application placement problem is also formulated as a function of application's service delivery deadline that resists their QoS degradation. These aspects form the core innovation part of the policy that helps to overcome the limitation of existing placement policies. Additionally, the proposed policy deals with various Fog and Cloud-based costs and works in decentralized manner so that it can be synthesized with distributed Fog nodes.

7.3 System Overview

7.3.1 Features of Integrated Fog-Cloud environments

As a supplement to IoT, Fog computing executes latency-sensitive applications in proximate of data sources to offer services in real time. Conversely, as an extension of Cloud computing, Fog conducts IoT-data pre-processing so that communication and compu-

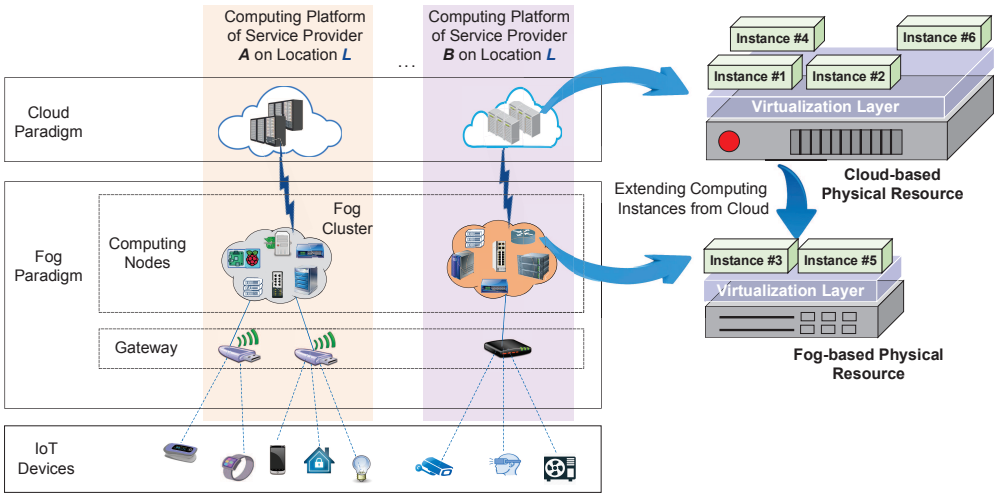

Figure 7.1: Integrated Fog-Cloud environments

tation overhead from Cloud datacentres can be reduced. Thus, Fog computing maintains an intermediate layer between IoT and Cloud computing [207] [199]. Based on this concept, the *Computing Platforms* for IoT applications are considered to be expanded across the Fog and Cloud infrastructure of integrated environments. Different providers deploy such platforms on various locations with their owned physical resources (Fog nodes and Cloud datacentres), for example, Fig. 7.1 shows the Computing Platforms deployed by provider *A* and *B* on location *L*. At the Fog part of each platform, provider-specific Gateways are deployed, and their number can be scalable as per the load of external connections with the platform. When multiple Gateways are associated to a Computing Platform, their operations are synchronized, and monetary calculations are performed in collective manner. Within a Fog cluster, the communication is maintained by faster Constrained Application Protocol and Simple Network Management Protocol. Since Gateways and Fog clusters are localized, their data exchange delay is considered negligible. In Fog clusters, cybersecurity frameworks are used to identify and monitor malicious Fog nodes that defend the Fog part of platforms from uncertain attacks in future [311]. Fog clusters can extend different types of virtualized instances (virtual machines and containers) from Cloud datacentres for application execution [199] [148]. While making application placement decisions in Fog infrastructure, the Cloud-based attributes of extended instances such as their configurations, price and cost model are

used extensively along with other performance parameters [312] [313].

Moreover, it is regarded that most of the IoT devices are cramped of performing large-scale data processing and maintaining direct communication with Cloud because of their resource scarcity and bandwidth limitations. In such circumstance, Bottom-up interaction among IoT, Fog and Cloud computing plays an important role where IoT devices at first communicate with Fog infrastructure and notify their service requirements. Fog infrastructure tries to meet these requirements with its available resources. If it is infeasible, Fog infrastructure asks Cloud to deal with the issue [306] [200]. To enable such interaction in the devised integrated environments, IoT devices are configured with the Gateways of any accessible Computing Platforms and the reliable links between Fog and Cloud part of Computing Platforms help the Gateways to reach out Cloud-based services via Fog clusters on behalf of the IoT devices. However, due to mobility of IoT devices, their associated Gateway and Computing Platform can change with the course of time. Therefore, to maintain connectivity and deal with data traffic, related network Service Function Chains (SFCs) are transferred from one place to another. In integrated environments, efficient SFC migration approaches are applied to support this operation with reduced reconfiguration cost [314].

While configuring an IoT device with a Gateway, the placement request for corresponding application is narrated. A placement request comprises specifications of the application including its number of instructions, input packet size, data receiving frequency, expected service delivery time limit and user's budget for its execution. However, IoT device and user-driven contexts can vary from time to time. Therefore, integrated environments endorse time series analytic frameworks for dependable data extraction so that varying contexts of these entities can be tracked and placement requests for the applications can be updated [315]. After assimilation of an application placement request, the associated Gateway grasps status such as processing speed, bandwidth, per unit time costs and price of available instances [200]. Based on these parameters, the Gateway finds suitable placement option for the application. If any instance satisfies minimum resource requirements of an application, its deployment time on that instance becomes trivial.

Unremitting application placement requests received by a Gateway can intensify its

management overhead. Therefore, Gateways conduct placement of applications after a certain interval, for example 0.100 seconds. It helps to manage the overhead of Gateways, simplifies their synchronization with a Computing Platform and resists unnecessary reporting. However, it can increase resource wastage and redundancy to some extent. Their effect can be mitigated by setting the interval between two placement rounds to a minimum value or dynamically tuning it according to the average runtime of applications. In devised integrated environment, providers can follow any of these approaches so that placement round intervals neither burden the Gateways nor decrease resource utilization [240]. Within this interval, Gateways receive new placement requests and instances execute the applications placed at previous round. Before initiating a placement round k, available instances C_k and requested applications R_k for that round are identified. Later, the Gateway estimates profit of the platform provider for executing each application. During placement, a single instance can host at most one application. A placement request is successful if the application is mapped to a computing instance, and its service is ensured to be delivered within the deadline. For k^{th} placement round, the set of successful applications is noted as R_k^{χ}. If an application is not scheduled in a placement round, it is considered for scheduling in the next round along with newly received placement requests. This process continues unless the application is placed, or its estimated service delivery time surpasses the deadline. During a billing period, a Gateway can run numerous placement rounds targeting the Computing Platform. However, after a billing period, compensation for users based on the SLA violation rate of corresponding platform is determined and total profit of its provider is calculated. Relevant notations for these calculations are shown in Table 7.2.

7.3.2 Gross Profit Estimation for Providers

Before placing an application $r \in R_k$ on instance $c \in C_k$, *Gross profit* e_c^r of provider for executing the application is estimated. Usually, Gross profit is calculated by deducting the cost of production from the revenue. Here, the revenue refers to the service charge of instance that is collected by provider from user to execute the application. Conversely, the cost of production is the operating cost of instance that is paid by provider to others for

Symbol	Definition
K	Total number of placement rounds per billing period on a Computing Platform.
Y	Total Gross profit of providers from a Computing Platform per billing period.
I_k	Gross profit of providers from a Computing Platform during k^{th} placement round.
C_k	Set of all computational instances during k^{th} placement round on a Computing Platform.
R_k	Set of all requested applications during k^{th} placement round on a Computing Platform.
R_k^{χ}	Set of successful applications during k^{th} placement round on a Computing Platform. $R_k^{\chi} \subseteq R_k$.
e_c^r	Estimated Gross profit for executing the application $r \in R_k$ on instance $c \in C_k$.
m_c^r	Profit Merit (PM) of an application $r \in R_k$ on any instance $c \in C_k$.
l^r	Input packet size for the application $r \in R_k$.
s^r	Number of instructions in the application $r \in R_k$.
z^r	Minimum resources required for hosting the application $r \in R_k$, $z \in \{\text{processing cores, memory}\}$.
ψ^r	Users budget for executing the application $r \in R_k$.
δ^r	User expected service delivery time limit for the application $r \in R_k$.
ξ^r	Latency sensitivity index for the application $r \in R_k$.
μ^c	Processing speed of a computing instance $c \in C_k$.
λ^c	Network bandwidth of a computing instance $c \in C_k$.
Z^c	Available resources on a computing instance $c \in C_k$, $Z \in \{\text{processing cores, memory}\}$.
ω^c	Cloud-based price of a computing instance $c \in C_k$ for per unit time.
α^c	Cost of computing instance $c \in C_k$ for processing resource consumption per unit time.
β^c	Cost of computing instance $c \in C_k$ for network resource consumption per unit time.
ϵ^c	Additional price of a computing instance $c \in C_k$ for per unit time.
τ_a^r	Arrival time stamp of placement request for application $r \in R_k$.
τ_ϑ^r	Placement time stamp of application $r \in R_k$.
τ	Current time stamp.
P	Net profit for the provider per billing period.
ρ	Compensation given for SLA violation per billing period on a Computing Platform.
Φ	Set of all requested applications per billing period on a Computing Platform. $R_k \subset \Phi$
ϕ	Set of all QoS-satisfied applications per billing period on a Computing Platform. $R_k^{\chi} \subset \phi$
φ	Set of all SLA-violated applications per billing period on a Computing Platform.
t_{rc}^p	Input processing time on computing instance $c \in C_k$ for application $r \in R_k$.
t_{rc}^n	Input propagation time to computing instance $c \in C_k$ for application $r \in R_k$.
t_{rc}	Total time required to complete the execution of application $r \in R_k$ on instance $c \in C_k$.
v_{rc}	Performance improvement grade of application $r \in R_k$ for extending instance $c \in C_k$ form Cloud to Fog.
Ω_{rc}	Service charge to users for executing the application $r \in R_k$ on instance $c \in C_k$.
Γ_{rc}	Operational cost of providers for executing the application $r \in R_k$ on instance $c \in C_k$.
$\eta_c \in \{0,1\}$	Equals to 1 if computing instance $c \in C_k$ is running in remote Cloud, 0 otherwise.
$x_{rc} \in \{0,1\}$	Equals to 1 if the application $r \in R_k$ is mapped to $c \in C_k$, 0 otherwise.

Table 7.2: Notations for profit-aware management

application execution. For Gross profit estimation, input packet size l^r and number of instructions in the application s^r are extracted from the placement request. Processing speed μ^c and network bandwidth λ^c of the instance are also considered. Input processing time t^p_{rc} and input propagation time t^n_{rc} are calculated for the application using Eqs.(7.1) and (7.2);

$$t^p_{rc} = \frac{s^r}{\mu^c}, \tag{7.1}$$

$$t^n_{rc} = \frac{l^r}{\lambda^c}. \tag{7.2}$$

If the instance is deployed in Cloud part, service charge of the instance for executing the application depends on its per unit time price ω^c and the summation of input processing time t^p_{rc} and input propagation time t^n_{rc}. Its operating cost also relies on t^p_{rc} and t^n_{rc} along with its per unit time processing cost α^c and networking cost β^c. In α^c and β^c, providers encapsulate certain portion of various expenses such as deployment, migration, energy and security management costs separately. The Gross profit for executing the application in Cloud-based instance is estimated using Eq. (7.3);

$$e^r_{c \in Cloud} = \omega^c(t^p_{rc} + t^n_{rc}) - (t^p_{rc}\alpha^c + t^n_{rc}\beta^c). \tag{7.3}$$

However, if the instance resides in Fog part, input propagation time t^n_{rc} becomes negligible. Therefore, its impact on service charge and operational cost are omitted while estimating the Gross profit. To align with the characteristics of Internet economy [212], providers can also charge users ϵ^c price per unit time on top of ω^c for ensuring improved service through Fog-based placement of applications. It is usually advertised to users as the price for extending the instance from Cloud to Fog. Hence, the Gross profit for application execution on Fog-based instance is estimated by Eq. (7.4);

$$e^r_{c \in Fog} = t^p_{rc}(\omega^c + \epsilon^c) - t^p_{rc}\alpha^c. \tag{7.4}$$

Combining Eqs. (7.3) and (7.4), a general narration of Gross profit for executing an application is shown in Eq. (7.5);

$$e^r_c = (t^p_{rc} + \eta_c t^n_{rc})\{\omega^c + (1 - \eta_c)\epsilon^c\} - (t^p_{rc}\alpha^c + \eta_c t^n_{rc}\beta^c). \tag{7.5}$$

Here, the binary variable η_c tracks whether the instance is deployed in Cloud or extended to Fog part of the Computing Platform. Based on Eq. (7.5), Gross profit of provider per placement round and per billing period from a Computing Platform in respect of a Gateway is given by Eqs. (7.6) and (7.7);

$$I_k = \sum_{r \in R_k^{\chi}} e_c^r, \tag{7.6}$$

$$\Upsilon = \sum_{k=1}^{K} I_k. \tag{7.7}$$

7.3.3 Pricing Model for Fog Instances

To increase provider's Gross profit from Fog-based placement of applications in integrated environment, the following condition needs to be satisfied;

$$e^r_{c \in Fog} > e^r_{c \in Cloud}.$$

One of the possible ways to satisfy this condition is to raise provider's revenue from the Fog instance. It can be achieved by setting a higher ϵ^c value while charging users for the instance. Providers can set this value as per their interest with no guarantee of user acceptance. To attain user's acknowledgement, ϵ^c should reflect the value of improving performance for placing applications on Fog instances. Therefore, in defining ϵ^c, the performance improvement grade v_{rc} of application $r \in R_k$ is used that denotes per unit time improvement in service delivery of the application for extending its assigned instance $c \in C_k$ from Cloud datacentre to Fog cluster.

When instance c remains in Cloud, service delivery time of application r is the summation of input processing time t^p_{rc} and input propagation time t^n_{rc}. However, if the instance is extended to Fog, service delivery time of application r becomes dependent to input processing time t^p_{rc} and its rough improvement is equivalent to t^n_{rc} compared to Cloud-based placement. Hence, the performance improvement grade v_{rc} of application r can be narrated as Eq. (7.8);

$$v_{rc} = \frac{t^n_{rc}}{t^p_{rc}}. \tag{7.8}$$

Moreover, providers save a larger portion of networking cost when the application is executed in Fog [9]. It is also observed in assessing the value of ϵ^c. Considering performance improvement and cost saving attributes, to boost the revenue from Fog-based application placement, providers should set the value of ϵ^c satisfying the following condition;

$$\epsilon^c > v_{rc}(\omega^c - \beta^c).$$

This condition can also be certified with the help of Eqs. (7.3) and (7.4). In proposed profit-aware application placement policy, it is applied by adding of a very small charge ∂ per unit time, for example, 0.005 \$/s, with ϵ^c as shown in Eq. (7.9);

$$\epsilon^c = v_{rc}(\omega^c - \beta^c) + \partial. \tag{7.9}$$

7.3.4 Compensation Method and Net Profit Calculation

SLA of an application $r \in R_k$ violates when the Computing Platform fails to assist it in meeting the service delivery deadline. This deadline is determined by adding the user's expected service delivery time limit δ^r with the request's arrival time stamp τ_a^r. If service of the application is delivered within deadline, its QoS to users is satisfied. In a Computing Platform, per billing period users are only charged for the set of QoS satisfied applications ϕ and compensated for the set of SLA-violated applications φ. The compensation is given as a percentage of average Gross profit of providers that is accumulated from QoS-satisfied applications [316]. It is calculated using the ratio of SLA-violated and total number of requested applications $|\Phi|$ per billing period; where $|\Phi| = |\phi| + |\varphi|$. Hence, the total amount of compensation ρ given by the provider is shown in Eq. (7.10);

$$\rho = |\varphi| \times \frac{\Upsilon}{|\phi|} \times \frac{|\varphi|}{|\Phi|}. \tag{7.10}$$

This compensation method works as per the performance of Computing Platform. If the Computing Platform assists to increase the number of QoS satisfied applications, the total compensation reduces. Conversely, if its performance degrades, increased amount of compensation helps to retain the user's loyalty. This inverse relationship balances the

financial support of Computing Platform for both users and providers [317]

After determining the compensation, *Net profit P* of provider from the Computing Platform for a billing period is assessed. Net profit is calculated by deducting the non-operational cost from the total Gross profit. Here, the non-operational cost of providers refers to the amount of compensation that is repaid to the users. Repaying the compensation ρ by applying Eq. (7.11), the residual portion of Gross profit Y per billing period is regarded as the provider's Net profit P;

$$P = \mathrm{Y} - \rho. \tag{7.11}$$

7.4 Profit-aware Application Placement

7.4.1 Problem Formulation

According to Eq. (7.11), provider's Net profit P from a Computing Platform enhances if the Gross profit Y per billing period increases and the amount of compensation ρ decreases. To support these conditions during placement rounds, the proposed Profit-aware Application Placement policy prioritizes each application $r \in R_k$ in terms of estimated Gross profit e_c^r for execution on any instance $c \in C_k$ and latency sensitivity index ξ^r. On current time stamp τ , ξ^r refers to the remaining time from application's service delivery deadline as shown in Eq. (7.12);

$$\xi^r = \tau_a^r + \delta^r - \tau. \tag{7.12}$$

In the proposed policy, based on e_c^r and ξ^r, *Profit Merit (PM)* m_c^r of application $r \in R_k$ is calculated using Eq. (7.13);

$$m_c^r = \frac{e_c^r}{\xi^r}. \tag{7.13}$$

On an instance $c \in C_k$, if the estimated Gross profit e_c^r remains same $\forall r \in R_k$, the application having stringent deadline will have the higher PM value. Conversely, if two applications have identical latency sensitivity index, the application estimating elevated Gross profit for execution will have the higher PM value. According to these two cases,

in other scenarios, the PM value of an application r on any instance c signifies the relative weight of estimated Gross profit for execution and latency sensitivity index. Additionally, latency sensitivity index ξ^r of an application decreases with the course of time. As a result, if an application is placed at k^{th} round, its PM value on particular instance increases by $k + 1^{th}$ round. Based on these features of PM, the objective function of profit-aware application placement for any placement round k is formulated as Eq. (7.14), where a binary decision x_{rc} helps to identify optimal mapping of an application $r \in R_k$ to an instance $c \in C_k$;

$$\max \sum_{r \in R_k} x_{rc} m_c^r. \tag{7.14}$$

subject to,

$$\sum x_{rc} \leq 1; \forall r \in R_k. \tag{7.15}$$

$$x_{rc} z^r \leq Z^c; \forall r \in R_k, \forall Z, \forall z. \tag{7.16}$$

$$(t_{rc}^p + \eta_c t_{rc}^n) \leq \xi^r; \forall r \in R_k. \tag{7.17}$$

$$\Omega_{rc} \leq \psi^r; \forall r \in R_k, \tag{7.18}$$

where,

$$\Omega_{rc} = (t_{rc}^p + \eta_c t_{rc}^n)\{\omega^c + (1 - \eta_c)\epsilon^c\}. \tag{7.19}$$

Eq. (7.14) is a constrained ILP model that maximizes the total PM of applications during k^{th} placement round and Eqs. (7.15), (7.16), (7.17) and (7.18) specify its constraints. This objective function is required to solve at the beginning of each placement round. It can be solved with any ILP solver such as SCIP [215]. The constraints of Eq. (7.14) are discussed in the following subsections.

Placement Constraint

To deliver uninterrupted services, an application $r \in R_k$ requires exclusive access to the assigned instance $c \in C_k$. The constraint presented in Eq. (7.15) supports this condi-

tion by compelling a computing instance to host at most one application per placement round.

Resource Constraint

A computing instance $c \in C_k$ can host the application $r \in R_k$, if its available resources Z^c such as processing cores and memory meet minimum resource requirements z^c of the application. Eq. (7.16) enforces this constraint signifying that minimum resources to host an application is always available on its assigned computing instance.

QoS Constraint

Placement of the application $r \in R_k$ on an instance $c \in C_k$ will not be successful unless its QoS satisfaction is ensured. QoS of the application is satisfied when the propagation and processing of input data are completed within the remaining time from service delivery deadline. Eq. (7.17) imposes this constraint to the proposed application placement policy.

Budget Constraint

Total service charge Ω_{rc} of executing the application in $r \in R_k$ on an instance $c \in C_k$ should be within the affordability of the user. If user's budget is not sufficient compared to the total service charge, execution of that application will trigger negative gearing for the provider. Eq. (7.18) defines this constraint during application placement.

7.4.2 Enhancement of Profit

Complexity of solving the optimization problem noted in Eq. (7.14) using an ILP solver is very high. Through this method, it is not feasible to identify the application-instance mapping within stringent time frame for profit-aware application placement. Therefore, a heuristic-method to solve the placement problem is proposed. The heuristic is immanent in the *EnhanceProfit* procedure presented in Algorithm 6. It identifies the best-fit

Algorithm 6 Profit enhancement algorithm

1: **procedure** ENHANCEPROFIT(τ, C_k, R_k)
2: **for** $c := C_k$ **do**
3: **if** !$c.allocated$ **then**
4: $M_c \leftarrow -\infty$
5: $X_c \leftarrow null$
6: **for** $r := R_k$ **do**
7: **if** !$r.placed$ **then**
8: $t^p_{rc} \leftarrow \frac{s^r}{\mu_c}$
9: $t^n_{rc} \leftarrow \frac{l^r}{\lambda_c}$
10: $\upsilon_{rc} \leftarrow \frac{t^n_{rc}}{t^p_{rc}}$
11: $\zeta^r \leftarrow \tau^r_a + \delta^r - \tau$
12: $t_{rc} \leftarrow t^p_{rc} + \eta_c t^n_{rc}$
13: $\Omega_{rc} \leftarrow t_{rc}[\omega^c + (1-\eta_c)\{\upsilon_{rc}(\omega^c - \beta^c) + \partial\}]$
14: $\Gamma_{rc} \leftarrow t^p_{rc}\alpha^c + \eta_c t^n_{rc}\beta^c$
15: $e^r_c \leftarrow \Omega_{rc} - \Gamma_{rc}$
16: $m^r_c \leftarrow \frac{e^r_c}{\zeta^r}$
17: **if** $z^r \leq Z^c$ **then**
18: **if** $t_{rc} \leq \xi^r$ **then**
19: **if** $\Omega_{rc} \leq \psi^r$ **then**
20: **if** $M_c < m^r_c$ **then**
21: $M_c \leftarrow m^r_c$
22: $X_c \leftarrow r$
23: **if** $X_c \neq null$ **then**
24: $c.assignedApplication \leftarrow X_c$
25: $X_c.placed \leftarrow true$
26: $c.allocated \leftarrow true$
27: $\tau^{X_c}_{\vartheta} \leftarrow \tau$
28: $\phi.add(X_c)$

solution in terms of Profit Merit (PM) for placing applications to instances. Details of the EnhanceProfit procedure is described as follows.

To determine the application-instance mapping for the k^{th} placement round, EnhanceProfit procedure takes the current time stamp τ, set of available instances C_k and set of requested applications R_k as arguments. While identifying the mapping, at first Fog-based instances and later, the Cloud-based instances are considered. As shown in Algorithm 6, EnhanceProfit procedure consists of 4 steps:

1) For each instance $c \in C_k$, firstly, it is checked whether the instance has already been allocated to any application or not (line 2-3). If the instance is not allocated, two variables M_c and X_c are initialized (line 4-5). M_c tracks the maximum PM value associate with the instance c and X_c stores the application which is responsible for the value in M_c.

2) For the instance c, the placement request of each application $r \in R_k$ is exploited

(line 6). If the application has not been placed yet (line 7), the following parameters are determined for its placement in respect of instance c.

i. input data processing time t^p_{rc} through Eq. (7.1) (line 8).

ii. input data propagation time t^n_{rc} applying Eq. (7.2) (line 9).

iii. performance improvement grade v_{rc} by Eq. (7.8) (line 10).

iv. latency sensitivity index ξ^r according to Eq. (7.12) (line 11).

v. required time t_{rc} to complete the application execution based on t^p_{rc} and t^n_{rc}(line 12) .

vi. service charge Ω_{rc} for users to execute the application using Eq. (7.19) (line 13). The value of ϵ^c is derived from Eq. (7.9).

vii. operational cost Γ_{rc} for executing the application considering its input data propagation time t^n_{rc} and processing time t^p_{rc} along with per unit time networking cost β^c and processing cost α^c (line 14).

viii. Gross profit e^r_c for application execution by deducting operational cost Γ_{rc} from service charge Ω_{rc} (line 15).

ix. PM m^r_c by applying Eq. (7.13) (line 16).

3) Based on the calculation of step 2, resource, QoS and budget constraint for the placement are explored (line 17-19). The estimated m^r_c is also compared with M_c provided that the constraints are satisfied (line 20). If m^r_c is higher than M_c, then M_c is updated with the value of m^r_c and X_c is set to r (line 21-22). The intuition for performing these operations is to select an application r for placing on instance c which meets all the imposed constraints and has the maximum relative weight of estimated Gross Profit and latency sensitivity index on c. It not only increases the proportion of Gross Profit for executing application r on instance c but also reduces the scope of SLA violation for r. Consequently, it enhances the Net profit of providers.

4) If X_c refers to any application (line 23), that application is assigned to instance c (line 24). To ensure the placement constraint, $X_c.placed$ and $c.allocated$ are set true (line 25-26). The placement time $\tau^{X_c}_{\vartheta}$ of X_c is also set to current time stamp τ and X_c is added to the set of QoS satisfied application placement requests ϕ (line 27-28).

For each placement round of a billing period, EnhanceProfit procedure is required to be executed. However, from lines 2 to 28 in Algorithm 6, there are $\mathcal{O}(|C_k| \cdot |R_k|)$ iterations, where $|C_k|$ is the number of available computing instance and $|R_k|$ denotes the

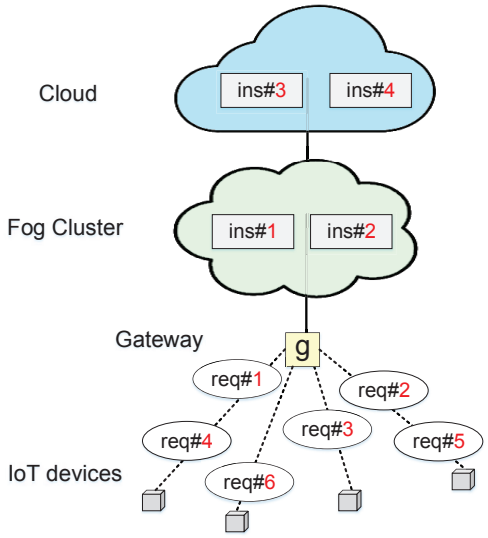

Figure 7.2: Illustrative integrated Fog-Cloud environments

number of received application placement requests during k^{th} placement round. Therefore, while identifying application-instance mapping per placement round, Algorithm 6 functions with polynomial time complexity. Theoretically, it also takes less amount of time to operate than ILP solvers. In addition, the proposed heuristic-method simultaneously enhances the Net profit of providers, ensures the QoS satisfaction of applications and meets the budget constraint of users which makes the method more effective for profit-aware application placement.

7.5 Illustrative Example

To numerically illustrate the basic steps of proposed profit-aware application placement policy, we have considered an integrated Fog-Cloud environment as depicted in Fig. 7.2.

Here, the Computing Platform offers 4 instances: two instances (ins#1, ins#2) are extended from Cloud to Fog part and two instances (ins#3, ins#4) remain at Cloud part. Configuration of the instances are summarized in Table 7.3. Here, Kilo byte per second (KB/s) and Thousand instruction per second (TI/s) refers to the unit of network bandwidth and processing speed for the instances respectively.

Instances	λ^c(KB/s)	μ^c(TI/s)	ω^c(\$/s)	α^c(\$/s)	β^c(\$/s)
ins#1	840.00	190.00	0.0380	0.0085	0.0065
ins#2	824.00	167.00	0.0364	0.0064	0.0050
ins#3	820.00	162.00	0.0360	0.0060	0.0047
ins#4	845.00	193.00	0.0386	0.0088	0.0068

Table 7.3: Parameters of computing instances

Requests	l^r(KB)	s^r (TI)	ψ^r (\$)	δ^r (sec)	τ_a^r
app#1	110.00	27.00	0.9054	0.5838	0.0170
app#2	80.00	23.00	0.8062	0.5115	0.0190
app#3	140.00	30.00	0.8915	0.6115	0.0330
app#4	90.00	26.00	0.9797	0.6075	0.0340
app#5	100.00	28.00	0.8384	0.6719	0.0360
app#6	130.00	21.00	0.9210	0.5448	0.0410

Table 7.4: Parameters of applications

At time $\tau = 0.0$ second, the Gateway g starts receiving placement requests and the placement round interval is set to 0.100 seconds. Thus, the first placement round occurs at $\tau = 0.100$ second. Details of requested applications before the first placement round are given in Table 7.4.

For the first placement round, entries of Table 7.3 and 7.4 are denoted as C_1 and R_1, and as part of EnhanceProfit procedure, the input data processing time t^p_{rc} and propagation time t^n_{rc}, $\forall r \in R_1$ on each $c \in C_1$ are determined. The value of t^p_{rc} and t^n_{rc} for this round are shown in Table 7.5. Later, $\forall r \in R_1$, the performance improvement grade v_{rc} are determined. They are figured out in respect of ins#1 and ins#2 those are extended from Cloud to Fog part of the platform. Since ins#3 and ins#4, remain in Cloud part, according to the proposed policy, performance improvement grade of applications regarding them are irrelevant. Moreover, execution service charge Ω_{rc} and operational cost Γ_{rc} of applications on each $c \in C_1$ are estimated. Here, ∂ is set to 0.005 \$/s. These estimations are presented in Table 7.6 and 7.7. Additionally, the PM values of each $r \in R_1$ on different $c \in C_1$ are calculated and they are listed in Table 7.8.

The proposed policy selects that instance for placing an application which ensures

Input data processing time t^p_{rc}				
Instances → Applications ↓	ins#1	ins#2	ins#3	ins#4
app#1	0.1421	0.1617	0.1667	0.1399
app#2	0.1211	0.1377	0.1420	0.1192
app#3	0.1579	0.1796	0.1852	0.1554
app#4	0.1368	0.1557	0.1605	0.1347
app#5	0.1474	0.1677	0.1728	0.1451
app#6	0.1105	0.1257	0.1296	0.1088

Input data propagation time t^n_{rc}				
Instances → Applications ↓	ins#1	ins#2	ins#3	ins#4
app#1	0.1310	0.1335	0.1341	0.1302
app#2	0.0952	0.0971	0.0976	0.0947
app#3	0.1667	0.1699	0.1707	0.1657
app#4	0.1071	0.1092	0.1098	0.1065
app#5	0.1190	0.1214	0.1220	0.1183
app#6	0.1548	0.1578	0.1585	0.1538

Table 7.5: Input data processing and propagation time

Instances → Applications ↓	ins#1	ins#2
app#1	0.9215	0.8257
app#2	0.7867	0.7049
app#3	1.0556	0.9458
app#4	0.7830	0.7015
app#5	0.8078	0.7238
app#6	1.4002	1.2546

Table 7.6: Performance improvement grade v_{rc} for Fog instances

Total service charge Ω_{rc}				
Instances → Applications ↓	ins#1	ins#2	ins#3	ins#4
app#1	0.0102	0.0109	0.0108	0.0104
app#2	0.0082	0.0088	0.0086	0.0083
app#3	0.0120	0.0128	0.0128	0.0124
app#4	0.0093	0.0099	0.0097	0.0093
app#5	0.0101	0.0108	0.0106	0.0102
app#6	0.0096	0.0102	0.0104	0.0101

Total operational cost Γ_{rc}				
Instances → Applications ↓	ins#1	ins#2	ins#3	ins#4
app#1	0.0012	0.0010	0.0016	0.0021
app#2	0.0010	0.0009	0.0013	0.0017
app#3	0.0013	0.0011	0.0019	0.0025
app#4	0.0012	0.0010	0.0015	0.0019
app#5	0.0013	0.0011	0.0016	0.0021
app#6	0.0009	0.0008	0.0015	0.0020

Table 7.7: Total service charge and operational cost

Instances → Applications ↓	ins#1	ins#2	ins#3	ins#4
app#1	0.0180	0.0197	~~0.0184~~	~~0.0166~~
app#2	0.0167	0.0183	0.0170	0.0152
app#3	0.0196	~~0.0213~~	~~0.0200~~	~~0.0182~~
app#4	0.0150	0.0164	0.0152	0.0137
app#5	0.0145	0.0159	0.0148	0.0133
app#6	0.0179	0.0193	0.0182	~~0.0167~~

Table 7.8: PM of applications for first placement round

Instances	Applications
ins#1	app#3
ins#2	app#1
ins#3	app#6
ins#4	app#2

Table 7.9: Placement map for first round

Instances → Applications ↓	ins#1	ins#2	ins#3	ins#4
app#4	0.0237	~~0.0260~~	~~0.0242~~	~~0.0217~~
app#5	0.0217	0.0237	~~0.0221~~	~~0.0198~~

Table 7.10: PM of applications for second placement round

maximum PM value (in red color on Table 7.8) for the application satisfying all constraints. The placement map for the first round is shown in Table 7.9.

In this example, the second placement round is supposed to occur at $\tau = 0.200$ second. Since all instances were busy on that time in executing previously placed applications, the second round occurs at $\tau = 0.300$ second. By this time all instances become available for C_2 and due to not receiving new requests, R_2 encapsulates only app#4 and app#5. The PM values of each $r \in R_2$ on $\forall c \in C_2$ are shown in Table 7.10. According to them, the placement map for the second round is shown in Table 7.11.

The illustrative example shows that the PM value of applications waiting for longer period of time gradually increases. However, the aforementioned operations for each placement round are conducted on Gateway g of the Computing Platform. We implement Gateway g with *Intel(R) Core(TM)2 Duo CPU E6550 @ 2.33GHz 2GB DDR2 RAM*. On this configuration, Gateway g takes 0.008 seconds on average to identify placement map for each round.

Instances	Applications
ins#1	~~app#4~~
ins#2	app#5

Table 7.11: Placement map for second round

7.6 Performance Evaluation

In this section, performance of the proposed profit-aware application placement policy is compared with the basic concept of *Completion time* [207], *Deadline* [177] and *Revenue-prioritized placement* policies [303]. In Deadline-prioritized placement policy, deadline-critical applications are placed on computationally powerful instances in higher precedence whereas in Revenue-prioritized placement policy, it is done for economically beneficial applications. Alternatively, through Completion time-prioritized placement policy, applications are placed on those instances which collectively ensure minimized application execution time. The profit-aware application placement problem for the proposed policy is also solved by following two approaches; in *Optimized Profit-aware placement*, the gateway runs SCIP [215] to find solution for Eq. (7.14), and in *Heuristic Profit-aware placement* the gateway executes EnhanceProfit procedure from Algorithm 6 to identify application-instance map during each placement round.

7.6.1 Simulation Environment

The experiments for performance evaluation of proposed policy are conducted in a simulated Fog-Cloud environment using iFogSim [219]. Instances of this environment are containerized and their specifications such as network bandwidth, processing speed and expenses are extracted from real-world references [318]. Since the instances offer short-term services, their per unit time price is comperatively higher than the instances provisioned for long-term services [319]. Additionally, to model the placement requests, synthetic workload is used since real-world workload for large number of different IoT applications are not currently available. The arrival pattern of these requests is Poisson distributed and their parametric standards are congruent with the configuration of instances. Numerical values of simulation attributes are determined from their given range in Table 7.12 through discrete uniform distribution.

Parameter	Numerical Specification
Simulation Duration	500 sec
Interval between placement rounds	0.020 – 0.180 sec
Total number of instances	50
Frequency of request arrival	10 – 50 applications/s
Instance:	
Network bandwidth	700 – 1000 KBPS
Processing speed	120 – 220 TIPS
Price	0.025 – 0.06 $/s
Cost of processing	0.005 – 0.01 $/s
Cost of networking	0.002 – 0.007 $/s
Application:	
Input packet size	80 – 140 KB
Number of instructions	20 – 30 TI
User's budget	0.80 – 1.00 $ per application
Service delivery time limit	0.500 – 0.700 sec

Table 7.12: Simulation parameters for profit-aware management

7.6.2 Performance Metrics

In experiments, the following metrics are used to evaluate the performance of proposed application placement policy.

1. Percentage of QoS-satisfied applications: The percentage of QoS satisfied applications $\phi^{\wp}$ is calculated as the ratio between number of QoS-satisfied applications $|\phi|$ and total number of requested application $|\Phi|$ per billing period using Eq. (7.20);

$$\phi^{\wp} = \frac{|\phi|}{|\Phi|} \times 100\%. \tag{7.20}$$

The higher percentage denotes the improved performance of integrated environment.

2. Average waiting time of applications: Waiting time of an application is defined as the interval between its requesting and placement time. The average waiting time $\bar{\tau^{w}}$ of

QoS-satisfied applications per billing period is calculated using Eq. (7.21);

$$\bar{\tau^{w}} = \frac{1}{|\phi|} \sum_{\forall r \in \phi} \tau^{r}_{\vartheta} - \tau^{r}_{a}. \tag{7.21}$$

The lower the waiting time signifies the higher the performance of integrated environment.

3. Total Gross profit of providers from a Computing Platform: Per billing period, this metric is calculated using Eq. (7.7). The increased total Gross profit refers to higher balance between service charge and operational cost. It also reflects the efficiency of providers in setting price of the instances.

4. Net profit of providers from a Computing Platform: This metric is calculated by Eq. (7.11) after each billing period. Enhanced Net profit signifies improved performance of the platform in reducing SLA violation and compensation.

Besides, *Percentage of compensation to Gross profit* is calculated in the experiments by Eq. (7.22);

$$\rho^{\wp} = \frac{\rho}{\Upsilon} \times 100\%. \tag{7.22}$$

Proportional relation between $\rho^{\wp}$ and the rate of SLA violation helps to support financial aspects of both users and providers. Moreover, *Average application completion time* and *Average service charge* for Fog and Cloud instances are analysed during experiments to demonstrate the improvement in application's service delivery and justify the applicability of proposed pricing model. *Average time to identify placement map* is also used as an performance metric. Reduced time to identify the placement map denotes the higher feasibility of applying corresponding approach for solving the placement problem.

The aforementioned performance metrics can have a significant impact on any real-world IoT-enabled system. For example, a remote health management system can request an integrated computing environment to place various IoT applications for measuring heartbeat, determining oxygen saturation level, monitoring electrocardiogram pattern and analysing sleep apnea data of different patients [2]. In such circumstance, high percentage of QoS satisfied applications is required for ensuring that the computing environment can offer services for most of the requested applications within their deadline. Similarly, the computing environment should guarantee lower waiting time

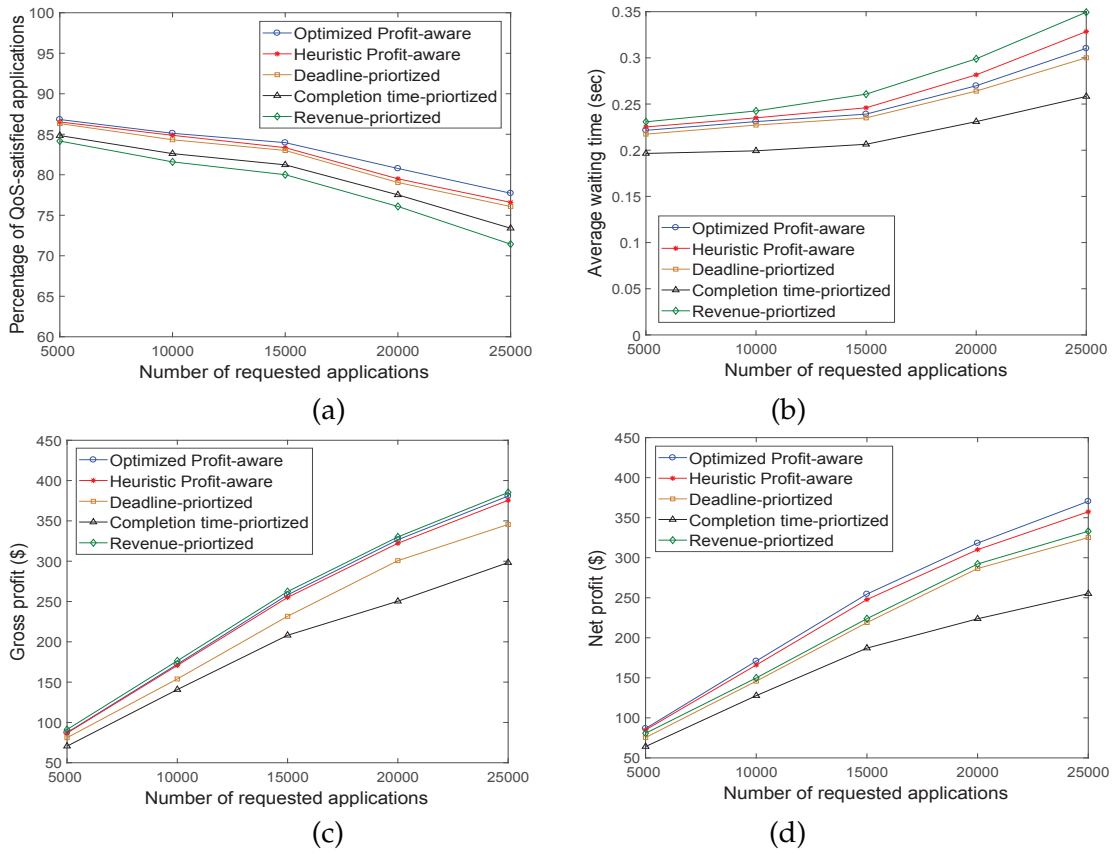

Figure 7.3: Impact of varying number of applications on (a) percentage of QoS satisfaction (b) waiting time (c) Gross profit (d) Net Profit

for all applications so that their execution can initiate in quicker time. Both will help the IoT-enabled system to deal with emergency situations of critical patients. However, the computing environment would need to manage the applications for remote health management system in such a way that can maximize Gross and Net profit of providers. Otherwise, the computing environment will be beneficial only for the IoT-system operators and patients, and there will be no financial support for the providers. Apart from these issues, the high percentage of compensation for increased number of SLA violations will assist the computing environment to attain loyalty of other IoT-enabled systems that can eventually work in favour of the providers.

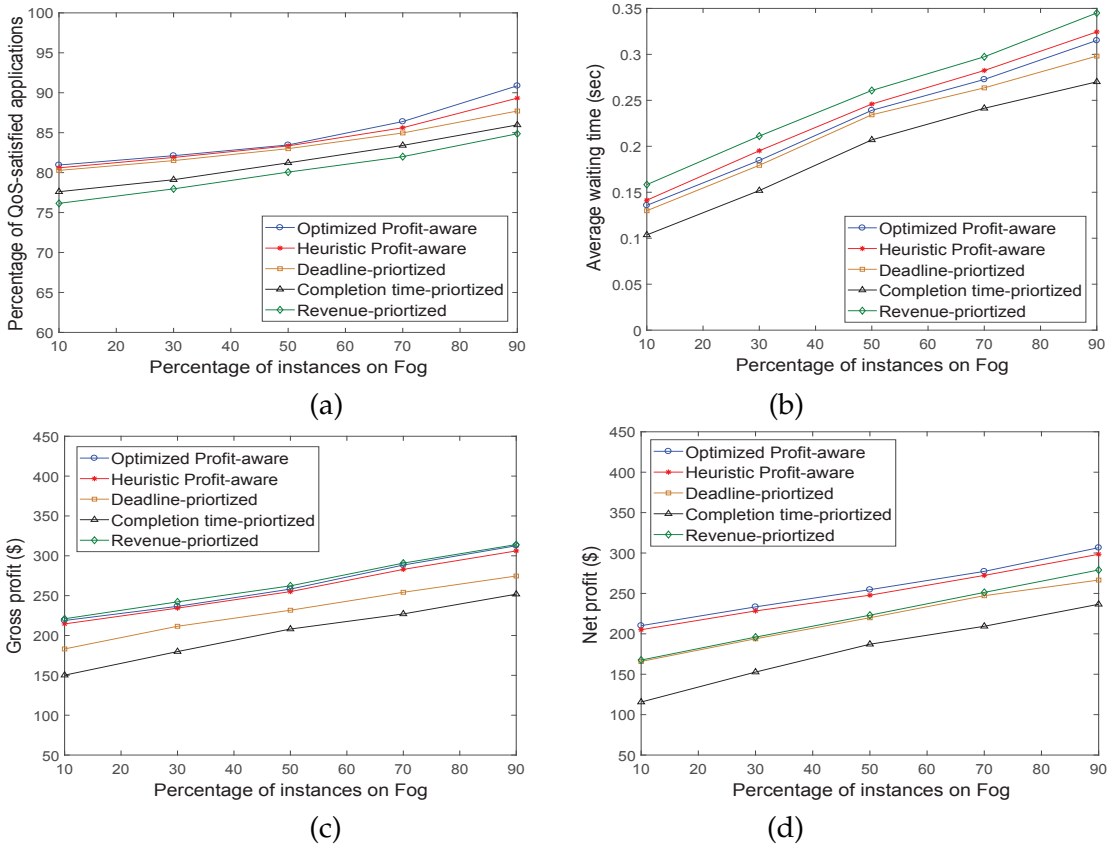

Figure 7.4: Impact of varying percentage of Fog instances on (a) percentage of QoS satisfaction (b) waiting time (c) Gross profit (d) Net profit

7.6.3 Experimental Results

The experiments for performance evaluation are conducted by varying the number of requested applications, the number of instances, the percentage of Fog instances and the interval between placement rounds separately. For each variation of these parameters, simulation is run for 500 seconds and the performance metrics are calculated only when the simulation is over. For simplicity, results of all homogeneous variations for a specific metric are combined in a single two-dimensional graph. These graphs are described in the following subsections.

Impact of varying number of applications

Due to receiving placement requests at higher frequency compared to the available rate of instances, the percentage of QoS satisfied applications decreases (Fig. 7.3.a). In Profit-aware application placement, this down trend is slower and closer to the Deadline-prioritized placement policy. The proposed policy schedules applications in precedence of their service delivery deadline. It raises the QoS satisfaction percentage compared to the Completion time and the Revenue-prioritized application placement where priority of applications largely depends on their program size and prospect of earning revenue. Moreover, average waiting time of placed applications prolongs as the number of requested application increases (Fig. 7.3.b). In this experiment, the proposed policy performs better than the Deadline-prioritized and the Revenue-prioritized policy since it increases the PM value of applications per placement round. However, it awaits latency-tolerant and less economical applications for a longer period of time. Conversely, the Completion time-prioritized placement policy releases instances quickly that helps to place more applications in next rounds and reduces their waiting time.

Since Net profit is accumulated from Gross profit, it always remains greater than Net profit. However, the instance pricing model of the proposed policy helps providers to increase revenue from Fog-based placement of applications that consequently boosts their Gross profit. Hence, the amount of Gross profit with the increasing of applications becomes higher for the proposed policy compared to Completion time and Deadline-prioritized placement policy (Fig. 7.3.c). Moreover, the proposed policy ensures QoS satisfaction for significant percentage of applications that resists SLA violations and assists providers to pay less compensation. For this reason, provider's Net profit elevates at higher rate in favour of the proposed policy compared to others as the number of applications increases (Fig. 7.3.d). Lower mount of Gross profit also results in reduced Net profit for Completion time and Deadline-prioritized placement policy. Nevertheless, the Revenue-prioritized placement policy performs almost similar to the proposed policy in increasing provider's Gross profit. Since the percentage of SLA violation rises for the Revenue-prioritized placement policy with the increasing of applications, it leads provider's to pay high compensation. Therefore, despite of elevating Gross profit, it offers less Net profit than the proposed policy.

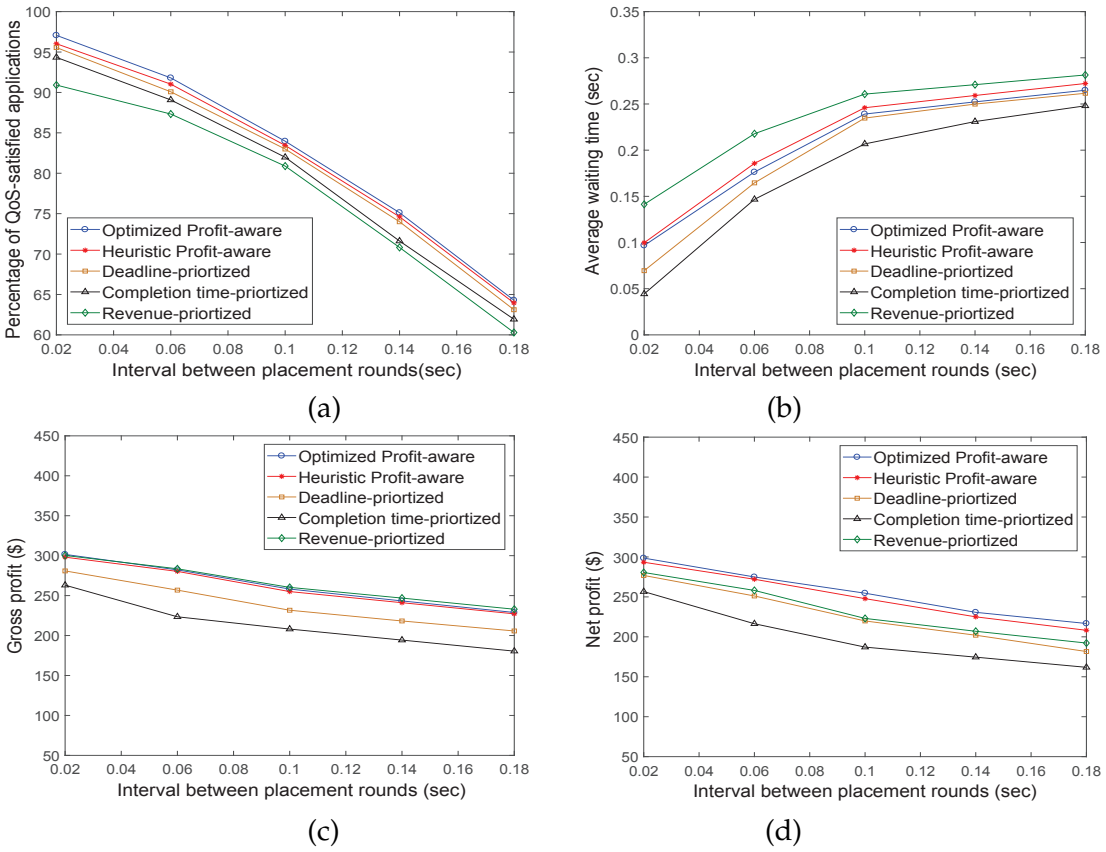

Figure 7.5: Impact of varying placement round interval on (a) percentage of QoS satisfaction (b) waiting time (c) Gross profit (d) Net Profit

The impact of varying number of requested applications on different performance metrics signify that placement request receiving frequency of a Computing Platform should be congruent with the availability rate of instances. It helps to maintain acceptable level of provider's profit and waiting time of applications with higher QoS satisfaction of users.

Impact of varying percentage of Fog instances

In a Computing Platform, if the percentage of Fog instances rises, the percentage of QoS satisfied application increases (Fig. 7.4.a). The higher number of Fog instances assist more applications to reduce their input propagation delay and to meet QoS. Besides, for

explicitly dealing with the service delivery deadline, the proposed and the Deadline-prioritized placement policy performs better in this case compared to the others.

Similarly, as the number of Fog instances increases, average waiting time of applications increases (Fig. 7.4.b). Due to charging additional price for using Fog instances, Cloud instances remain as the only option to place low-budget applications. When the number of Cloud instances becomes less, these applications wait for a longer period of time. In this case, compared to other policies, the Completion time-prioritized placement policy performs better as it releases all kinds of instances quickly. Moreover, increased number of Fog instances elevates both Gross and Net profit for providers (Fig. 7.4.c and 7.4.d). In these experiments, for concurrently maximizing revenue and reducing SLA violations, the proposed policy performs better than others.

However, results of the aforementioned experiments signify that a balanced ratio of Fog and Cloud instances assists both profit enhancement and waiting time management.

Impact of varying placement round interval

As the interval between two placement rounds increases, the percentage of QoS satisfaction decreases and waiting time of applications increases (Fig. 7.5.a and 7.5.b). This interval halts placement of applications even when the instances are already available and resists applications to meet service delivery deadline. However, for considering deadline during application placement, the proposed policy performs better in this experiment than others specially in terms of QoS satisfaction. Moreover, lower QoS satisfaction occurred form increased placement round interval decrease both Gross and Net profit of providers from a Computing Platform (Fig. 7.5.c and 7.5.d). Since the proposed policy offers compensation as a variable percentage of provider's total Gross profit, it ensures moderate Net profit despite of higher SLA violation than others.

Outcomes of these experiments recommend tuning the placement round interval in such way that neither creates overhead on gateway by invoking placement process frequently nor degrades the percentage of QoS satisfied requests.

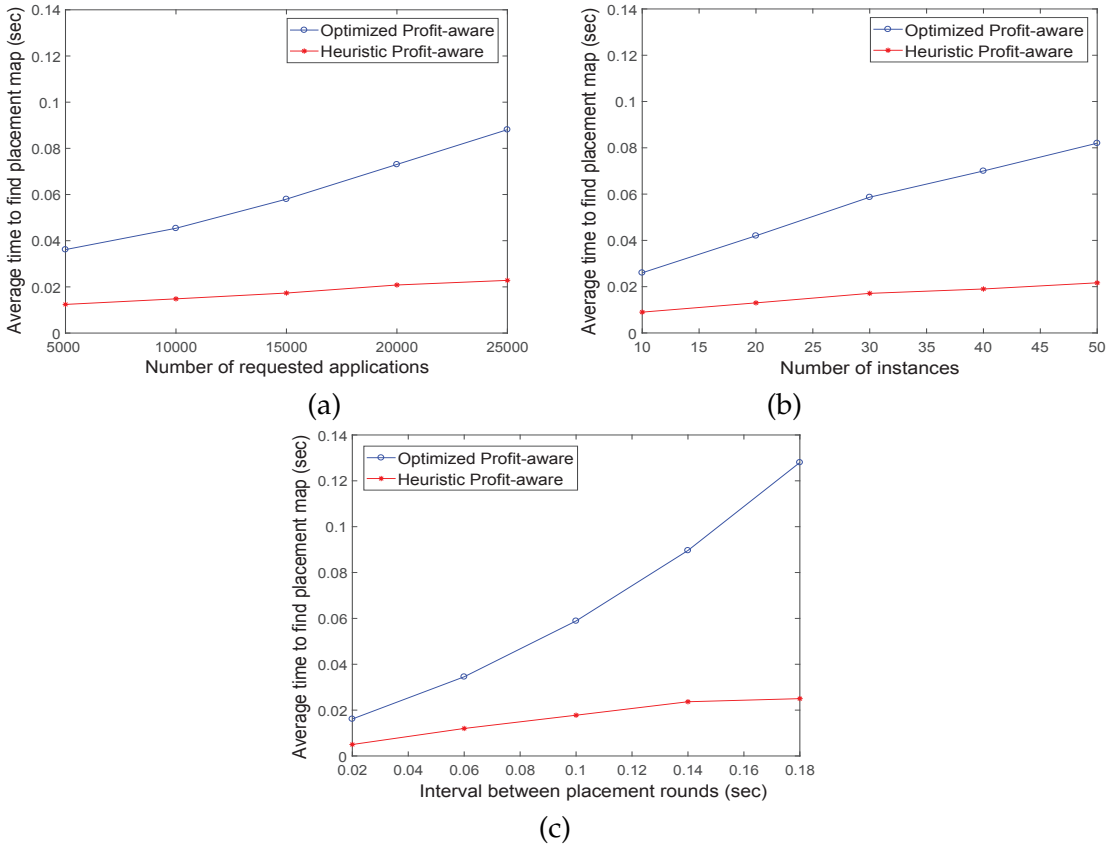

Figure 7.6: Required time to find placement map varying (a) number of requests (b) number of instances (c) placement round interval

Feasibility of placement problem solving approaches

In aforementioned results, the Optimized Profit-aware placement performs slightly better than the Heuristic Profit-aware placement. However, as the number of requested applications increases or the number of instances increases, or both happens due to increasing the placement round interval, the Optimized Profit-aware placement takes longer period of time to identify the application-instance map than the heuristic approach (Fig 7.6.a, 7.6.b and 7.6.c). Thus, the Optimized Profit-aware placement increases overhead on gateways and degrades their performance. Since the outcomes of both Optimized and Heuristic Profit-aware placement do not vary significantly, for in-time performance, it is feasible to apply the heuristic approach instead of the optimized one

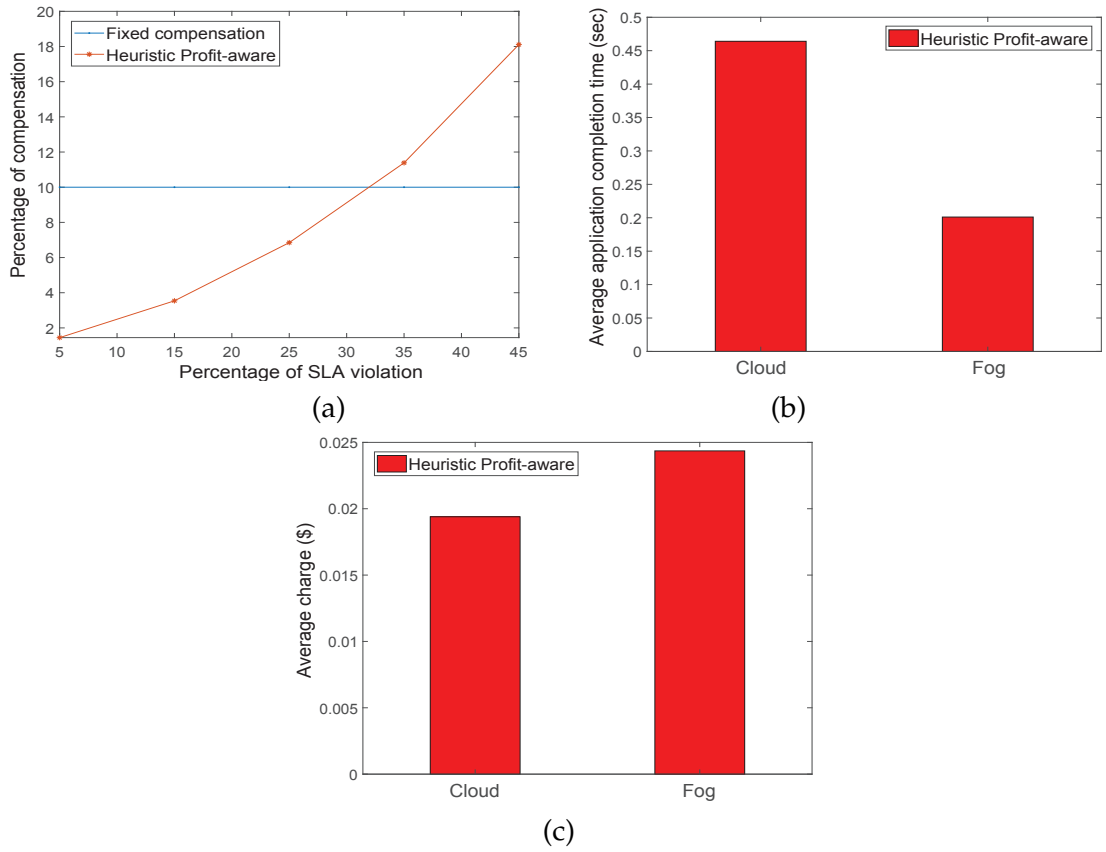

Figure 7.7: Comparison between (a) fixed and variable compensation (b) average application completion time (c) average charge for Fog and Cloud instances

to implement the proposed policy.

Justification for compensation and instance pricing

The proposed policy facilitates compensation according to the performance of Computing Platform. It ensures high compensation for higher percentage of SLA violation and vice versa (Fig. 7.7.a). Conversely, in fixed rate compensation method (10% of total revenue), the percentage of compensation to Gross profit remain static despite of performance improvement or degradation [320]. As a result, the Computing Platform fails to show its financial support to both providers and users. Besides, in such method, the distribution of compensation amount is uniform among the affected users on different

SLA violation rate. Therefore, the variable rate compensation method of the proposed policy is more conducive to build a financially supportive computing environments for both users and providers than the fixed rate compensation method.

Moreover, it is already proven that Fog-based placement of an application deliver services in reduced time compared to its Cloud-based placement. It happens due to less or negligible input propagation delay while executing an application in Fog instance. The experiment result shown in Fig. 7.7.b also certifies this fact with almost 55% improved completion time of applications for their Fog-based placement in context of the devised computing environment and proposed placement policy. For this experiment, a candidate set of applications is placed through the proposed policy on fixed number of Fog and Cloud instances separately. Identical configuration is maintained for each instances and workload of the applications are kept unchanged during the experiment. Thus, its fairness and validity are ensured. On same experimental setup, it is also observed that Fog instances charge additional 20% on average in contrast to Cloud instances (Fig. 7.7.c), which is quite less compared to the scale of performance improvement. Hence, it is realized that instance pricing model applied in the policy is justified and reflects an acceptable value for improving performance.

7.7 Summary

A profit-aware application placement policy for integrated Fog-Cloud environments is proposed in this work. The policy simultaneously increases provider's Gross and Net profit by placing applications on suitable instances without violating their deadline constraint. It incorporates a pricing model that tunes the service charge of Fog instances according to their capability of reducing application service delivery time. The policy follows a compensation method to mitigate the effect of SLA violation on users. The compensation method depends on the performance of computing environments and supportive for both providers and users. The proposed policy can identify application-instance map by using any ILP solver or best-fit heuristic approach. To demonstrate the efficacy of proposed policy, we applied the heuristic approach in an iFogSim-simulated environment and compared its performance with several existing application placement

policies. The experimental results show improvements in Gross and Net profit, waiting time and QoS satisfaction rate for the proposed policy. They also manifested that heuristic implementation of the policy finds closer to optimal solution within minimal time, and pricing of the Fog instances are justified to their performance.

Chapter 8
Conclusions and Future Directions

This chapter concludes the thesis and discusses a summary of works and key contributions. Then, it highlights several future research directions for further improvement of Fog computing concepts.

8.1 Summary of Contributions

In recent years, the number of Internet of Things (IoT) devices has increased to a great extent. Fog computing has been introduced to support the computational demand of real-time latency-sensitive applications of geographically distributed IoT devices. Fog computing environments reside closer to the IoT devices and extend the Cloud-based computing, storage and networking facilities to the users. It has already drawn significant attention from both industry and academia. However, most of the Fog nodes are not resource enriched. Therefore, large scale application development in resource-constrained nodes are not quite easy compared to traditional datacentres. Additionally, the SLA between users and service providers in Fog computing explicitly depends on the QoS requirements of the applications, the expectations of the users and profit of the providers. Therefore, in a particular scenario, it is quite difficult to specify the application service management metrics and the corresponding Service Level Objectives (SLOs). Moreover, the management of applications in Fog environments is subjected to dynamic scalability, interoperability and reliability-related issues. In Fog environments, these issues can be resolved by identifying suitable placement options for the applications. In this thesis, we investigated the placement of applications over resource-constrained, heterogeneous and distributed Fog nodes.

Chapter 1 presented the basic concept of Fog computing and highlighted its chal-

lenges. It also outlined the research questions addressed in this thesis. Then, Chapter 2 investigated the existing application management approaches in Fog computing from the perspectives of architecture, placement and maintenance. Later, it presented a taxonomy and reviewed the literature based on the taxonomy. It also highlighted a perspective model for application management in Fog environments.

Chapter 3 discussed an Edge affinity-based application management policy for Fog environments. Edge affinity denotes the relative intensity of user-defined deadline, amount of data per input and sensing frequency of IoT devices that defines the application characteristics. These characteristics help in identifying the necessity of an application for Fog-based placement to meet its QoS requirements. This policy is applicable for different types of applications including latency-sensitive, data-intensive and streaming applications. When a user has multiple applications and limited Fog resources to allocate them, our policy assists in classifying and selecting the applications as per their Edge affinity at the gateway level. Additionally, it determines the placement of an application to that Fog node which ensures the minimum application service delivery time.

Chapter 4 presented a latency-aware application management policy for Fog environments. The policy facilitates the placement of distributed applications which can be decomposed as a collection of inter-dependent Application Modules. Fog nodes are often organized in a hierarchical order. Our proposed policy places the latency-sensitive applications at the lower levels. It also forwards the latency-tolerant application to the higher levels based on the inter-module data dependency delay the inter-nodal communication delay of the Fog nodes. Additionally, the proposed policy consolidates the number of active Fog nodes through a module forwarding technique. It schedules the execution of modules on Fog nodes as per their input receiving frequency. Consequently, it assists in managing energy usage and application's QoS in the Fog environments.

Chapter 5 proposed a context-aware application management policy for Fog computing environments. The policy exploits the sensing frequency of IoT devices and the size of their generated data to determine the suitable placement options for the applications. The novel contribution of this policy is to support multi-tenancy on bare-metal without congesting the network and increasing the computational overhead significantly. It assists different streaming applications to leverage the capabilities of their host

Fog nodes. At the same time, it ensures that streaming applications are not required to alter their input processing destinations very frequently. Thus, the proposed policy helps in meeting the QoS of the applications in Fog environments. In this thesis, we discussed this policy from the perspective of Industry 4.0.

Chapter 6 discussed QoE-aware application management in Fog computing environments. The policy places the applications according to the user expectations. It is widely accepted that placement of applications based on user expectations helps to enhance the QoE of the user. This chapter explicitly distinguishes QoS and QoE from the user's perspective. The novelty of our proposed QoE-aware policy is to quantify the subjective measurement of user expectations. Moreover, this chapter incorporates an illustrative example to depict the numerical operations of the proposed policy and calculate the compound QoE gain of the users.

Chapter 7 investigated a profit-aware application management policy for Fog computing environments. It includes a pricing model for executing applications in Fog environments according to the level of performance improvement. Consequently, it enhances the profit of service providers form the Fog-based placement of the applications. Additionally, it facilitates compensation to users for the SLA violations. The compensation amount maintains an inverse relation with the QoS satisfaction percentage that observes the economic benefit for both the service providers and users.

The chapters mentioned above collectively present an IoT application management system for Fog computing environments that simultaneously improves application's QoS, user's QoE and provider's profit. It is indeed a timely contribution to the state-of-the-art.

8.2 Future Research Directions

In this thesis, we addressed several challenges of application management in Fog computing environments. However, there exist some other issues that require to be investigated more comprehensively. This section gives some insights into these challenges for future work in this area.

8.2.1 Trade-off between energy and accuracy

The accuracy level of an application largely depends on the sensing rate of the IoT devices especially during smart surveillance. Conversely, it elevates their energy consumption significantly. When the renewable energy sources power the IoT devices, energy management becomes even more complicated. To deal with such cases, the dynamic tuning of the accuracy level and the sensing frequency of the IoT devices can be a potential solution.

8.2.2 User demand-driven application management

Users' service demand can change during application execution. It is also difficult to grasp and predict the users' dynamics in real-time. If they are overlooked while executing the applications, the QoE can degrade. A dynamic demand-driven application management policy from the user's perspective can be helpful to address this issue.

8.2.3 Artificial intelligence-based application management

Currently, artificial intelligence is receiving significant attention due its ability of solving complex problems. A huge amount of training data set is required to build an artificial intelligence-based system, which is very easy to accumulate in Fog. It will also help to predict the future resource requirements, context variation and nodal failures more precisely, and manage the applications accordingly.

8.2.4 Pricing and detailed estimation of Fog resources

The Cloud-based pricing models for subscription-oriented services cannot be directly applied to Fog computing due to the localized demand and distributed deployment of the IoT-enabled systems. For the same reasons, resource over-provisioning can also occur in Fog computing environments. Therefore, detailed estimation of resources in Fog computing is needed considering the number of IoT devices within the CPS and their application service requirements simultaneously. It will also help to develop an

efficient business model for the Fog computing environments without relying on the Cloud-based pricing attributes.

8.2.5 Trusted service orchestration in Fog

The Fog infrastructure can be private or public. The publicly available Fog infrastructure is highly exposed to security threats. On the other hand, service of private-owned Fog infrastructure is subjected to lack of transparency. In this case, a trusted service orchestration policy is required to ensure the collaboration and reliability between different types of Fog computing infrastructure.

8.2.6 Fog node consolidation and scaling

Fog nodes are resource-constrained. Inclusion of more Fog nodes can help to alleviate this limitation to some extent. However, it also increases the deployment cost, communication interference and energy consumption at the edge network. In this situation, dynamic consolidation and scaling of Fog nodes as per the demand can be helpful.

8.2.7 Lightweight security features

Due to the distributed nature of blockchain, it is regarded as one of the crucial elements to enable security in Fog computing. However, the blockchain feature in Fog itself introduces significant computation overhead to resource-constrained Fog nodes. Therefore, it is required to develop a lightweight blockchain technique for Fog environments.

8.2.8 Application-specific management

Fog computing is intended to support various sensitive IoT applications from different domains including smart healthcare, transport, city, agriculture and industry. These IoT applications have specific requirements and need specialized management. Application-specific management can be helpful in dealing with sensitive applications in Fog.

8.3 Final Remarks

Fog computing has already attracted significant attention because of its feasibility for IoT-driven use cases. However, the QoS-aware management of applications is a major concern for Fog computing. In this thesis, we investigated how the service quality of IoT applications can be enhanced by placing them efficiently over resource-constrained, heterogeneous and distributed Fog nodes. The algorithms, mathematical models, and system architectures proposed in this thesis optimizes the service delivery time of applications, enhances the QoE of users, and increases the profit of providers. These research outcomes also provide opportunities for further innovation and development in the domain of IoT and Fog computing.

Bibliography

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," Future Generation Computer Systems, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] R. Mahmud, F. L. Koch, and R. Buyya, "Cloud-fog interoperability in iot-enabled healthcare solutions," in Proceedings of the 19th International Conference on Distributed Computing and Networking, ser. ICDCN '18. New York, NY, USA: ACM, 2018, pp. 32:1–32:10.

[3] A. Chowdhury, G. Karmakar, and J. Kamruzzaman, "The co-evolution of cloud and iot applications: Recent and future trends," in Handbook of Research on the IoT, Cloud Computing, and Wireless Network Optimization. IGI Global, 2019, pp. 213–234.

[4] L. Belli, S. Cirani, L. Davoli, A. Gorrieri, M. Mancin, M. Picone, and G. Ferrari, "Design and Deployment of an IoT Application-Oriented Testbed," Computer, vol. 48, no. 9, pp. 32–40, Sep. 2015.

[5] B. Martinez, M. Montón, I. Vilajosana, and J. D. Prades, "The power of models: Modeling power consumption for iot devices," IEEE Sensors Journal, vol. 15, no. 10, pp. 5777–5789, Oct 2015.

[6] M. Yannuzzi, R. Milito, R. Serral-Gracià, D. Montero, and M. Nemirovsky, "Key ingredients in an iot recipe: Fog computing, cloud computing, and more fog computing," in 2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), Dec 2014, pp. 325–329.

[7] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," Future Generation Computer Systems, vol. 25, no. 6, pp. 599 – 616, 2009.

[8] H. Madsen, B. Burtschy, G. Albeanu, and F. Popentiu-Vladicescu, "Reliability in the utility computing era: Towards reliable Fog computing," in 2013 20th International Conference on Systems, Signals and Image Processing (IWSSIP), July 2013, pp. 43–46.

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

[10] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," Journal of Systems Architecture, vol. 98, pp. 289 – 330, 2019.

[11] J. Yao and N. Ansari, "QoS-Aware Fog Resource Provisioning and Mobile Device Power Control in IoT Networks," IEEE Transactions on Network and Service Management, vol. 16, no. 1, pp. 167–175, March 2019.

[12] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, and O. Rana, "Fog computing for the internet of things: A Survey," ACM Transactions on Internet Technology (TOIT), vol. 19, no. 2, p. 18, 2019.

[13] IoT For All, "The Big Three Make a Play for the Fog," https://www.iotforall.com/big-three-make-play-fog/, 2018, [Online; accessed 06-October-2018].

[14] Industrial Internet Consortium, "The Industrial Internet Consortium And Openfog Consortium Join Forces," https://www.iiconsortium.org/press-room/01-31-19.htm, 2019, [Online; accessed 19-December-2019].

[15] Cisco, "IOx and Fog Applications," https://www.cisco.com/c/en_in/solutions/internet-of-things/iot-fog-applications.html, 2016, [Online; accessed 19-December-2019].

[16] SBWire, "Fog Computing Market to Grow at 65% CAGR Till 2024 : Cisco, Dell, Nebbiolo, IBM, Intel, Microsoft and 15 Other Companies Profile," http://www.digitaljournal.com/pr/3997363, 2018, [Online; accessed 19-December-2019].

[17] B. Cheng, G. Solmaz, F. Cirillo, E. Kovacs, K. Terasawa, and A. Kitazawa, "FogFlow: Easy Programming of IoT Services Over Cloud and Edges for Smart Cities," IEEE Internet of Things Journal, vol. 5, no. 2, pp. 696–707, April 2018.

[18] M. H. Syed, E. B. Fernandez, and M. Ilyas, "A Pattern for Fog Computing," in Proceedings of the 10th Travelling Conference on Pattern Languages of Programs, ser. VikingPLoP '16. New York, NY, USA: ACM, 2016, pp. 13:1–13:10.

[19] M. Maksimović, "Implementation of Fog computing in IoT-based healthcare system," Jita-Journal Of Information Technology And Aplications, vol. 14, no. 2, 2018.

[20] A. W. G. OpenFog Consortium, "Openfog reference architecture for fog computing," OPFRA001, vol. 20817, p. 162, 2017.

[21] A. Dastjerdi, H. Gupta, R. Calheiros, S. Ghosh, and R. Buyya, "Fog Computing: principles, architectures, and applications," in Internet of Things: Principles and Paradigms, R. Buyya and A. V. Dastjerdi, Eds. Morgan Kaufmann, 2016, pp. 61 – 75.

[22] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the Suitability of Fog Computing in the Context of Internet of Things," IEEE Transactions on Cloud Computing, vol. PP, no. 99, pp. 1–1, 2015.

[23] C. Hong and B. Varghese, "Resource Management in Fog/Edge Computing: A Survey," CoRR, vol. abs/1810.00305, 2018. [Online]. Available: http://arxiv.org/abs/1810.00305

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

[25] P. Varshney and Y. Simmhan, "Demystifying fog computing: Characterizing architectures, applications and abstractions," in 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC), 2017, pp. 115–124.

[26] A. R. Benamer, H. Teyeb, and N. Ben Hadj-Alouane, "Latency-Aware Placement Heuristic in Fog Computing Environment," in On the Move to Meaningful

Internet Systems. OTM 2018 Conferences, H. Panetto, C. Debruyne, H. A. Proper, C. A. Ardagna, D. Roman, and R. Meersman, Eds. Cham: Springer International Publishing, 2018, pp. 241–257.

[27] C. Fiandrino, N. Allio, D. Kliazovich, P. Giaccone, and P. Bouvry, "Profiling Performance of Application Partitioning for Wearable Devices in Mobile Cloud and Fog Computing," IEEE Access, vol. 7, pp. 12 156–12 166, 2019.

[28] S. Prabavathy, K. Sundarakantham, S. M. Shalinie, and K. N. Mallikarjunan, "Fog Computing-Based Autonomic Security Approach to Internet of Things Applications," in Computational Intelligence: Theories, Applications and Future Directions - Volume II, N. K. Verma and A. K. Ghosh, Eds. Singapore: Springer Singapore, 2019, pp. 3–14.

[29] J. Mass, C. Chang, and S. N. Srirama, "Context-aware Edge Process Management for Mobile Thing-to-fog Environment," in Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings, ser. ECSA '18. New York, NY, USA: ACM, 2018, pp. 44:1–44:7.

[30] P. Wiener, P. Zehnder, and D. Riemer, "Towards Context-Aware and Dynamic Management of Stream Processing Pipelines for Fog Computing," in 2019 IEEE 3rd International Conference on Fog and Edge Computing (ICFEC), May 2019, pp. 1–6.

[31] J. Luo, L. Yin, J. Hu, C. Wang, X. Liu, X. Fan, and H. Luo, "Container-based fog computing architecture and energy-balancing scheduling algorithm for energy IoT," Future Generation Computer Systems, vol. 97, pp. 50 – 60, 2019.

[32] P. Gazori, D. Rahbari, and M. Nickray, "Saving time and cost on the scheduling of fog-based IoT applications using deep reinforcement learning approach," Future Generation Computer Systems, 2019.

[33] T. Huang, W. Lin, Y. Li, L. He, and S. Peng, "A Latency-Aware Multiple Data Replicas Placement Strategy for Fog Computing," Journal of Signal Processing Systems, vol. 91, no. 10, pp. 1191–1204, Oct 2019.

[34] T. Wang, J. Zhou, A. Liu, M. Z. A. Bhuiyan, G. Wang, and W. Jia, "Fog-Based Computing and Storage Offloading for Data Synchronization in IoT," IEEE Internet of Things Journal, vol. 6, no. 3, pp. 4272–4282, June 2019.

[35] C. Anglano, M. Canonico, P. Castagno, M. Guazzone, and M. Sereno, "Profit-aware coalition formation in fog computing providers: A game-theoretic approach," Concurrency and Computation: Practice and Experience, vol. 0, no. 0, p. e5220, 2019.

[36] S. Filiposka, A. Mishev, and K. Gilly, "Mobile-aware dynamic resource management for edge computing," Transactions on Emerging Telecommunications Technologies, vol. 30, no. 6, p. e3626, 2019.

[37] H. Noura, O. Salman, A. Chehab, and R. Couturier, "Preserving data security in distributed fog computing," Ad Hoc Networks, vol. 94, p. 101937, 2019.

[38] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, and R. Ranjan, "Fog Computing: Survey of trends, architectures, requirements, and research directions," IEEE access, vol. 6, pp. 47 980–48 009, 2018.

[39] A. Barak and O. La'adan, "The mosix multicomputer operating system for high performance cluster computing," Future Generation Computer Systems, vol. 13, no. 4, pp. 361 – 372, 1998, hPCN '97.

[40] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, "Economic models for resource management and scheduling in grid computing," Concurrency and Computation: Practice and Experience, vol. 14, no. 13-15, pp. 1507–1542, 2002.

[41] A. Celesti, F. Tusa, M. Villari, and A. Puliafito, "How to enhance cloud architectures to enable cross-federation," in 2010 IEEE 3rd International Conference on Cloud Computing, July 2010, pp. 337–345.

[42] K. Kumar and Y. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" Computer, vol. 43, no. 04, pp. 51–56, apr 2010.

[43] Z. Zhu, P. Gupta, Q. Wang, S. Kalyanaraman, Y. Lin, H. Franke, and S. Sarangi, "Virtual base station pool: Towards a wireless network cloud for radio access networks," in Proceedings of the 8th ACM International Conference on Computing Frontiers, ser. CF '11. New York, NY, USA: Association for Computing Machinery, 2011.

[44] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in 2012 IEEE Symposium on Computers and Communications (ISCC), July 2012, pp. 000 059–000 066.

[45] A. Voellmy and J. Wang, "Scalable software defined network controllers," SIGCOMM Comput. Commun. Rev., vol. 42, no. 4, p. 289–290, Aug. 2012.

[46] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5g," ETSI white paper, vol. 11, no. 11, pp. 1–16, 2015.

[47] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," IEEE Communications Surveys Tutorials, vol. 18, no. 1, pp. 236–262, Firstquarter 2016.

[48] S. Park, O. Simeone, and S. Shamai Shitz, "Joint optimization of cloud and edge processing for fog radio access networks," IEEE Transactions on Wireless Communications, vol. 15, no. 11, pp. 7621–7632, Nov 2016.

[49] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637–646, Oct 2016.

[50] J. S. Preden, K. Tammemäe, A. Jantsch, M. Leier, A. Riid, and E. Calis, "The Benefits of Self-Awareness and Attention in Fog and Mist Computing," Computer, vol. 48, no. 7, pp. 37–45, July 2015.

[51] R. K. Barik, A. C. Dubey, A. Tripathi, T. Pratik, S. Sasane, R. K. Lenka, H. Dubey, K. Mankodiya, and V. Kumar, "Mist Data: Leveraging Mist Computing for Secure and Scalable Architecture for Smart and Connected Health," Procedia Computer

Science, vol. 125, pp. 647 – 653, 2018, the 6th International Conference on Smart Computing and Communications.

[52] W. Shi and S. Dustdar, "The Promise of Edge Computing," Computer, vol. 49, no. 5, pp. 78–81, May 2016.

[53] M. Uehara, Mist Computing: Linking Cloudlet to Fogs. Cham: Springer International Publishing, 2018, pp. 201–213.

[54] M. Satyanarayanan, "The Emergence of Edge Computing," Computer, vol. 50, no. 1, pp. 30–39, Jan 2017.

[55] R. Mahmud, R. Kotagiri, and R. Buyya, "Fog computing: A taxonomy, survey and future directions," in Internet of everything. Springer, 2018, pp. 103–130.

[56] F. Chiti, R. Fantacci, and B. Picano, "A matching game for tasks offloading in integrated edge-fog computing systems," Transactions on Emerging Telecommunications Technologies, vol. 0, no. 0, p. e3718, 2019.

[57] G. Cristescu, R. Dobrescu, O. Chenaru, and G. Florea, "DEW: A New Edge Computing Component for Distributed Dynamic Networks," in 2019 22nd International Conference on Control Systems and Computer Science (CSCS), May 2019, pp. 547–551.

[58] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: architecture, key technologies, applications and open issues," Journal of network and computer applications, vol. 98, pp. 27–42, 2017.

[59] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," IEEE Communications Surveys & Tutorials, vol. 20, no. 1, pp. 416–464, 2017.

[60] C. Li, Y. Xue, J. Wang, W. Zhang, and T. Li, "Edge-oriented computing paradigms: A survey on architecture design and system management," ACM Computing Surveys (CSUR), vol. 51, no. 2, p. 39, 2018.

[61] M. Ghobaei-Arani, A. Souri, and A. A. Rahmanian, "Resource management approaches in fog computing: a comprehensive review," Journal of Grid Computing, pp. 1–42, 2019.

[62] O. Osanaiye, S. Chen, Z. Yan, R. Lu, K.-K. R. Choo, and M. Dlodlo, "From cloud to fog computing: A review and a conceptual live VM migration framework," IEEE Access, vol. 5, pp. 8284–8300, 2017.

[63] E. Baccarelli, P. G. V. Naranjo, M. Scarpiniti, M. Shojafar, and J. H. Abawajy, "Fog of everything: Energy-efficient networked computing architectures, research challenges, and a case study," IEEE access, vol. 5, pp. 9882–9910, 2017.

[64] P. Bellavista, J. Berrocal, A. Corradi, S. K. Das, L. Foschini, and A. Zanni, "A survey on fog computing for the Internet of Things," Pervasive and Mobile Computing, vol. 52, pp. 71 – 99, 2019.

[65] S. B. Nath, H. Gupta, S. Chakraborty, and S. K. Ghosh, "A Survey of Fog Computing and Communication: Current Researches and Future Directions," CoRR, vol. abs/1804.04365, 2018. [Online]. Available: http://arxiv.org/abs/1804.04365

[66] M. Aazam, S. Zeadally, and K. A. Harras, "Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities," Future Generation Computer Systems, vol. 87, pp. 278–289, 2018.

[67] F. A. Kraemer, A. E. Braten, N. Tamkittikhun, and D. Palma, "Fog computing in healthcare–a review and discussion," IEEE Access, vol. 5, pp. 9206–9222, 2017.

[68] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," IEEE Communications Surveys & Tutorials, vol. 20, no. 3, pp. 1826–1857, 2018.

[69] C. Perera, Y. Qin, J. C. Estrella, S. Reiff-Marganiec, and A. V. Vasilakos, "Fog computing for sustainable smart cities: A survey," ACM Computing Surveys (CSUR), vol. 50, no. 3, p. 32, 2017.

[70] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," Future Generation Computer Systems, vol. 78, pp. 680–698, 2018.

[71] S. N. Shirazi, A. Gouglidis, A. Farshad, and D. Hutchison, "The extended cloud: Review and analysis of mobile edge computing and fog from a security and resilience perspective," IEEE Journal on Selected Areas in Communications, vol. 35, no. 11, pp. 2586–2595, 2017.

[72] P. Zhang, M. Zhou, and G. Fortino, "Security and trust issues in Fog computing: A survey," Future Generation Computer Systems, vol. 88, pp. 16–27, 2018.

[73] S. H. Abbasi, N. Javaid, M. H. Ashraf, M. Mehmood, M. Naeem, and M. Rehman, "Load Stabilizing in Fog Computing Environment Using Load Balancing Algorithm," in Advances on Broadband and Wireless Computing, Communication and Applications, L. Barolli, F.-Y. Leu, T. Enokido, and H.-C. Chen, Eds. Cham: Springer International Publishing, 2019, pp. 737–750.

[74] H. R. Arkian, A. Diyanat, and A. Pourkhalili, "MIST: Fog-based data analytics scheme with cost-efficient resource provisioning for IoT crowdsensing applications," Journal of Network and Computer Applications, vol. 82, pp. 152 – 165, 2017.

[75] O. Fadahunsi and M. Maheswaran, "Locality sensitive request distribution for fog and cloud servers," Service Oriented Computing and Applications, vol. 13, no. 2, pp. 127–140, Jun 2019.

[76] H. T. T. Binh, T. T. Anh, D. B. Son, P. A. Duc, and B. M. Nguyen, "An Evolutionary Algorithm for Solving Task Scheduling Problem in Cloud-Fog Computing Environment," in Proceedings of the Ninth International Symposium on Information and Communication Technology, ser. SoICT 2018. New York, NY, USA: ACM, 2018, pp. 397–404.

[77] K. Fizza, N. Auluck, O. Rana, and L. Bittencourt, "PASHE: Privacy Aware Scheduling in a Heterogeneous Fog Environment," in 2018 IEEE 6th International

Conference on Future Internet of Things and Cloud (FiCloud), Aug 2018, pp. 333–340.

[78] J. He, J. Wei, K. Chen, Z. Tang, Y. Zhou, and Y. Zhang, "Multitier Fog Computing With Large-Scale IoT Data Analytics for Smart Cities," IEEE Internet of Things Journal, vol. 5, no. 2, pp. 677–686, April 2018.

[79] F. Karatas and I. Korpeoglu, "Fog-Based Data Distribution Service (F-DAD) for Internet of Things (IoT) applications," Future Generation Computer Systems, vol. 93, pp. 156 – 169, 2019.

[80] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On Reducing IoT Service Delay via Fog Offloading," IEEE Internet of Things Journal, vol. 5, no. 2, pp. 998–1010, April 2018.

[81] E. AbdElhalim, M. Obayya, and S. Kishk, "Distributed Fog-to-Cloud computing system: A minority game approach," Concurrency and Computation: Practice and Experience, vol. 31, no. 15, p. e5162, 2019.

[82] A. Anzanpour, H. Rashid, A. M. Rahmani, A. Jantsch, N. Dutt, and P. Liljeberg, "Energy-efficient and Reliable Wearable Internet-of-Things through Fog-Assisted Dynamic Goal Management," Procedia Computer Science, vol. 151, pp. 493 – 500, 2019, the 10th International Conference on Ambient Systems, Networks and Technologies (ANT 2019) / The 2nd International Conference on Emerging Data and Industry 4.0 (EDI40 2019) / Affiliated Workshops.

[83] R. Ding, X. Li, X. Liu, and J. Xu, "A Cost-Effective Time-Constrained Multi-workflow Scheduling Strategy in Fog Computing," in Service-Oriented Computing – ICSOC 2018 Workshops, X. Liu, M. Mrissa, L. Zhang, D. Benslimane, A. Ghose, Z. Wang, A. Bucchiarone, W. Zhang, Y. Zou, and Q. Yu, Eds. Cham: Springer International Publishing, 2019, pp. 194–207.

[84] T. Djemai, P. Stolf, T. Monteil, and J. Pierson, "A Discrete Particle Swarm Optimization Approach for Energy-Efficient IoT Services Placement Over Fog In-

frastructures," in 2019 18th International Symposium on Parallel and Distributed Computing (ISPDC), June 2019, pp. 32–40.

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

[86] F. Concone, G. L. Re, and M. Morana, "A Fog-Based Application for Human Activity Recognition Using Personal Smart Devices," ACM Trans. Internet Technol., vol. 19, no. 2, pp. 20:1–20:20, Mar. 2019.

[87] S. Dehnavi, H. R. Faragardi, M. Kargahi, and T. Fahringer, "A reliability-aware resource provisioning scheme for real-time industrial applications in a Fog-integrated smart factory," Microprocessors and Microsystems, vol. 70, pp. 1 – 14, 2019.

[88] P. Kayal and J. Liebeherr, "Autonomic Service Placement in Fog Computing," in 2019 IEEE 20th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM), June 2019, pp. 1–9.

[89] I. Lera, C. Guerrero, and C. Juiz, "Availability-Aware Service Placement Policy in Fog Computing Based on Graph Partitions," IEEE Internet of Things Journal, vol. 6, no. 2, pp. 3641–3651, April 2019.

[90] N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "Towards Fault Tolerant Fog Computing for IoT-Based Smart City Applications," in 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), Jan 2019, pp. 0752–0757.

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

[92] J. Santos, T. Wauters, B. Volckaert, and F. De Turck, "Towards Network-Aware

Resource Provisioning in Kubernetes for Fog Computing Applications," in 2019 IEEE Conference on Network Softwarization (NetSoft), June 2019, pp. 351–359.

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

[94] D. Zeng, L. Gu, and H. Yao, "Towards energy efficient service composition in green energy powered Cyber–Physical Fog Systems," Future Generation Computer Systems, 2018.

[95] C. Guerrero, I. Lera, and C. Juiz, "A lightweight decentralized service placement policy for performance optimization in fog computing," Journal of Ambient Intelligence and Humanized Computing, vol. 10, no. 6, pp. 2435–2452, Jun 2019.

[96] C. Zhu, J. Tao, G. Pastor, Y. Xiao, Y. Ji, Q. Zhou, Y. Li, and A. Ylä-Jääski, "Folo: Latency and Quality Optimized Task Allocation in Vehicular Fog Computing," IEEE Internet of Things Journal, vol. 6, no. 3, pp. 4150–4161, June 2019.

[97] C. Anglano, M. Canonico, and M. Guazzone, "Online User-driven Task Scheduling for FemtoClouds," in 2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC), June 2019, pp. 5–12.

[98] M. Al-khafajiy, T. Baker, H. Al-Libawy, Z. Maamar, M. Aloqaily, and Y. Jararweh, "Improving fog computing performance via Fog-2-Fog collaboration," Future Generation Computer Systems, vol. 100, pp. 266 – 280, 2019.

[99] Z. Liu, X. Yang, Y. Yang, K. Wang, and G. Mao, "DATS: Dispersive Stable Task Scheduling in Heterogeneous Fog Networks," IEEE Internet of Things Journal, vol. 6, no. 2, pp. 3423–3436, April 2019.

[100] C. Li, H. Zhuang, Q. Wang, and X. Zhou, "SSLB: Self-Similarity-Based Load Balancing for Large-Scale Fog Computing," Arabian Journal for Science and Engineering, vol. 43, no. 12, pp. 7487–7498, Dec 2018.

[101] D. Tychalas and H. Karatza, "A Scheduling Algorithm for a Fog Computing System with Bag-of-Tasks Jobs: Simulation and Performance Evaluation," Simulation Modelling Practice and Theory, vol. 98, p. 101982, 2020.

[102] M. Nasir, K. Muhammad, J. Lloret, A. K. Sangaiah, and M. Sajjad, "Fog computing enabled cost-effective distributed summarization of surveillance videos for smart cities," Journal of Parallel and Distributed Computing, vol. 126, pp. 161 – 170, 2019.

[103] J. Yue, M. Xiao, and Z. Pang, "Distributed Fog Computing Based on Batched Sparse Codes for Industrial Control," IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4683–4691, Oct 2018.

[104] S. Li, M. A. Maddah-Ali, and A. S. Avestimehr, "Coding for distributed fog computing," IEEE Communications Magazine, vol. 55, no. 4, pp. 34–40, April 2017.

[105] T. Jeong, J. Chung, J. W. Hong, and S. Ha, "Towards a distributed computing framework for fog," in 2017 IEEE Fog World Congress (FWC), Oct 2017, pp. 1–6.

[106] X. Pang, Z. Bie, and X. Lin, "Access point decoding coded mapreduce for tree fog network," in 2018 International Conference on Network Infrastructure and Digital Content (IC-NIDC), Aug 2018, pp. 384–388.

[107] S. Imai, C. A. Varela, and S. Patterson, "A performance study of geo-distributed iot data aggregation for fog computing," in 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion), Dec 2018, pp. 278–283.

[108] L. Dang, M. Dong, K. Ota, J. Wu, J. Li, and G. Li, "Resource-efficient secure data sharing for information centric e-health system using fog computing," in 2018 IEEE International Conference on Communications (ICC), May 2018, pp. 1–6.

[109] M. I. NAAS, L. Lemarchand, J. Boukhobza, and P. Raipin, "A Graph Partitioning-based Heuristic for Runtime IoT Data Placement Strategies in a Fog Infras-

tructure," in Proceedings of the 33rd Annual ACM Symposium on Applied Computing, ser. SAC '18. New York, NY, USA: ACM, 2018, pp. 767–774.

[110] R. Oma, S. Nakamura, D. Duolikun, T. Enokido, and M. Takizawa, "Fault-Tolerant Fog Computing Models in the IoT," in Advances on P2P, Parallel, Grid, Cloud and Internet Computing, F. Xhafa, F.-Y. Leu, M. Ficco, and C.-T. Yang, Eds. Cham: Springer International Publishing, 2019, pp. 14–25.

[111] U. Ozeer, X. Etchevers, L. Letondeur, F.-G. Ottogalli, G. Salaün, and J.-M. Vincent, "Resilience of Stateful IoT Applications in a Dynamic Fog Environment," in Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, ser. MobiQuitous '18. New York, NY, USA: ACM, 2018, pp. 332–341.

[112] C. Mouradian, S. Kianpisheh, M. Abu-Lebdeh, F. Ebrahimnezhad, N. T. Jahromi, and R. H. Glitho, "Application Component Placement in NFV-Based Hybrid Cloud/Fog Systems With Mobile Fog Nodes," IEEE Journal on Selected Areas in Communications, vol. 37, no. 5, pp. 1130–1143, May 2019.

[113] M. Suter, R. Eidenbenz, Y. Pignolet, and A. Singla, "Fog Application Allocation for Automation Systems," in 2019 IEEE International Conference on Fog Computing (ICFC), June 2019, pp. 97–106.

[114] C. Wu and L. Wang, "A Deadline-Aware Estimation of Distribution Algorithm for Resource Scheduling in Fog Computing Systems," in 2019 IEEE Congress on Evolutionary Computation (CEC), June 2019, pp. 660–666.

[115] M. S. Elbamby, M. Bennis, W. Saad, M. Latva-aho, and C. S. Hong, "Proactive edge computing in fog networks with latency and reliability guarantees," EURASIP Journal on Wireless Communications and Networking, vol. 2018, no. 1, p. 209, Aug 2018.

[116] T. Choudhari, M. Moh, and T.-S. Moh, "Prioritized Task Scheduling in Fog Computing," in Proceedings of the ACMSE 2018 Conference, ser. ACMSE '18. New York, NY, USA: ACM, 2018, pp. 22:1–22:8.

[117] N. K. Giang, R. Lea, and V. C. Leung, "Developing applications in large scale, dynamic fog computing: A case study," Software: Practice and Experience, vol. 0, no. 0, 2019.

[118] L. Yin, J. Luo, and H. Luo, "Tasks Scheduling and Resource Allocation in Fog Computing Based on Containers for Smart Manufacturing," IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4712–4721, Oct 2018.

[119] B. Kim, S. Heo, G. Lee, S. Song, J. Kim, and H. Kim, "Spinal Code: Automatic Code Extraction for Near-user Computation in Fogs," in Proceedings of the 28th International Conference on Compiler Construction, ser. CC 2019. New York, NY, USA: ACM, 2019, pp. 87–98.

[120] P. G. Vinueza Naranjo, E. Baccarelli, and M. Scarpiniti, "Design and energy-efficient resource management of virtualized networked Fog architectures for the real-time support of IoT applications," The Journal of Supercomputing, vol. 74, no. 6, pp. 2470–2507, Jun 2018.

[121] D. Wang, Z. Liu, X. Wang, and Y. Lan, "Mobility-Aware Task Offloading and Migration Schemes in Fog Computing Networks," IEEE Access, vol. 7, pp. 43 356–43 368, 2019.

[122] Y. Wang, K. Wang, H. Huang, T. Miyazaki, and S. Guo, "Traffic and Computation Co-Offloading With Reinforcement Learning in Fog Computing for Industrial Applications," IEEE Transactions on Industrial Informatics, vol. 15, no. 2, pp. 976–986, Feb 2019.

[123] J. Zhou, J. Fan, J. Wang, and J. Jia, "Dynamic service deployment for budget-constrained mobile edge computing," Concurrency and Computation: Practice and Experience, vol. 0, no. 0, p. e5436, 2019.

[124] Y. Nan, W. Li, W. Bao, F. C. Delicato, P. F. Pires, Y. Dou, and A. Y. Zomaya, "Adaptive Energy-Aware Computation Offloading for Cloud of Things Systems," IEEE Access, vol. 5, pp. 23 947–23 957, 2017.

[125] T. Nazar, N. Javaid, M. Waheed, A. Fatima, H. Bano, and N. Ahmed, "Modified Shortest Job First for Load Balancing in Cloud-Fog Computing," in Advances on Broadband and Wireless Computing, Communication and Applications, L. Barolli, F.-Y. Leu, T. Enokido, and H.-C. Chen, Eds. Cham: Springer International Publishing, 2019, pp. 63–76.

[126] Z. Su, Q. Xu, J. Luo, H. Pu, Y. Peng, and R. Lu, "A secure content caching scheme for disaster backup in fog computing enabled mobile social networks," IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4579–4589, Oct 2018.

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

[128] D. Arya and M. Dave, "Priority Based Service Broker Policy for Fog Computing Environment," in Advanced Informatics for Computing Research, D. Singh, B. Raman, A. K. Luhach, and P. Lingras, Eds. Singapore: Springer Singapore, 2017, pp. 84–93.

[129] M. R. Ramli, P. T. Daely, J. Lee, and D. Kim, "Bio-inspired Service Provisioning Scheme for Fog-based Industrial Internet of Things," in 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Sep. 2019, pp. 1661–1664.

[130] M. Avgeris, D. Dechouniotis, N. Athanasopoulos, and S. Papavassiliou, "Adaptive resource allocation for computation offloading: A control-theoretic approach," ACM Trans. Internet Technol., vol. 19, no. 2, pp. 23:1–23:20, Apr. 2019.

[131] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal Workload Allocation in Fog-Cloud Computing Toward Balanced Delay and Power Consumption," IEEE Internet of Things Journal, vol. 3, no. 6, pp. 1171–1181, Dec 2016.

[132] A. A. Gad-Elrab and A. Y. Noaman, "A two-tier bipartite graph task allocation approach based on fuzzy clustering in cloud–fog environment," Future Generation Computer Systems, vol. 103, pp. 79 – 90, 2020.

[133] A. Karamoozian, A. Hafid, and E. M. Aboulhamid, "On the Fog-Cloud Cooperation: How Fog Computing can address latency concerns of IoT applications," in 2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC), June 2019, pp. 166–172.

[134] Liqing Liu, Zheng Chang, Xijuan Guo, and T. Ristaniemi, "Multi-objective optimization for computation offloading in mobile-edge computing," in 2017 IEEE Symposium on Computers and Communications (ISCC), July 2017, pp. 832–837.

[135] M. M. Mahmoud, J. J. Rodrigues, K. Saleem, J. Al-Muhtadi, N. Kumar, and V. Korotaev, "Towards energy-aware fog-enabled cloud of things for healthcare," Computers & Electrical Engineering, vol. 67, pp. 58 – 69, 2018.

[136] H. Mazouzi, N. Achir, and K. Boussetta, "DM2-ECOP: An Efficient Computation Offloading Policy for Multi-user Multi-cloudlet Mobile Edge Computing Environment," ACM Trans. Internet Technol., vol. 19, no. 2, pp. 24:1–24:24, Apr. 2019.

[137] B. Nair and M. S. B. Somasundaram, "Overload prediction and avoidance for maintaining optimal working condition in a fog node," Computers & Electrical Engineering, vol. 77, pp. 147 – 162, 2019.

[138] N. Auluck, A. Azim, and K. Fizza, "Improving the Schedulability of Real-Time Tasks using Fog Computing," IEEE Transactions on Services Computing, pp. 1–1, 2019.

[139] M. Bhatia, S. K. Sood, and S. Kaur, "Quantum-based predictive fog scheduler for IoT applications," Computers in Industry, vol. 111, pp. 51 – 67, 2019.

[140] W. Fang, W. Zhang, W. Chen, Y. Liu, and C. Tang, "TMSRS: trust management-based secure routing scheme in industrial wireless sensor network with fog computing," Wireless Networks, Sep 2019.

[141] G. Zhang, F. Shen, Z. Liu, Y. Yang, K. Wang, and M. Zhou, "Femto: Fair and energy-minimized task offloading for fog-enabled iot networks," IEEE Internet of Things Journal, vol. 6, no. 3, pp. 4388–4400, June 2019.

[142] M. A. Benblidia, B. Brik, L. Merghem-Boulahia, and M. Esseghir, "Ranking fog nodes for tasks scheduling in fog-cloud environments: A fuzzy logic approach," in 2019 15th International Wireless Communications Mobile Computing Conference (IWCMC), June 2019, pp. 1451–1457.

[143] W.-S. Kim and S.-H. Chung, "User incentive model and its optimization scheme in user-participatory fog computing environment," Computer Networks, vol. 145, pp. 76 – 88, 2018.

[144] H. Shah-Mansouri and V. W. S. Wong, "Hierarchical Fog-Cloud Computing for IoT Systems: A Computation Offloading Game," IEEE Internet of Things Journal, vol. 5, no. 4, pp. 3246–3257, Aug 2018.

[145] D.-N. Vu, N.-N. Dao, Y. Jang, W. Na, Y.-B. Kwon, H. Kang, J. J. Jung, and S. Cho, "Joint energy and latency optimization for upstream IoT offloading services in fog radio access networks," Transactions on Emerging Telecommunications Technologies, vol. 30, no. 4, p. e3497, 2019.

[146] M. Wazid, A. K. Das, N. Kumar, and A. V. Vasilakos, "Design of secure key management and user authentication scheme for fog computing services," Future Generation Computer Systems, vol. 91, pp. 475 – 492, 2019.

[147] S. Ahn, M. Gorlatova, P. Naghizadeh, M. Chiang, and P. Mittal, "Adaptive Fog-Based Output Security for Augmented Reality," in Proceedings of the 2018 Morning Workshop on Virtual Reality and Augmented Reality Network, ser. VR/AR Network '18. New York, NY, USA: ACM, 2018, pp. 1–6.

[148] A. Alrawais, A. Alhothaily, C. Hu, X. Xing, and X. Cheng, "An Attribute-Based Encryption Scheme to Secure Fog Communications," IEEE Access, vol. 5, pp. 9131–9138, 2017.

[149] H. Li, K. Ota, and M. Dong, "Deep Reinforcement Scheduling for Mobile Crowdsensing in Fog Computing," ACM Trans. Internet Technol., vol. 19, no. 2, pp. 21:1–21:18, Apr. 2019.

[150] Y. Yao, X. Chang, J. Mišić, and V. Mišić, "Reliable and Secure Vehicular Fog Service Provision," IEEE Internet of Things Journal, vol. 6, no. 1, pp. 734–743, Feb 2019.

[151] S. Zhao, Y. Yang, Z. Shao, X. Yang, H. Qian, and C. Wang, "FEMOS: Fog-Enabled Multitier Operations Scheduling in Dynamic Wireless Networks," IEEE Internet of Things Journal, vol. 5, no. 2, pp. 1169–1183, April 2018.

[152] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, and J. Rodriguez, "Computation Resource Allocation and Task Assignment Optimization in Vehicular Fog Computing: A Contract-Matching Approach," IEEE Transactions on Vehicular Technology, vol. 68, no. 4, pp. 3113–3125, April 2019.

[153] Y. Xiao and M. Krunz, "Distributed Optimization for Energy-Efficient Fog Computing in the Tactile Internet," IEEE Journal on Selected Areas in Communications, vol. 36, no. 11, pp. 2390–2400, Nov 2018.

[154] M. Zeng, Y. Li, K. Zhang, M. Waqas, and D. Jin, "Incentive Mechanism Design for Computation Offloading in Heterogeneous Fog Computing: A Contract-Based Approach," in IEEE International Conf. on Communications, May 2018, pp. 1–6.

[155] H. Zheng, K. Xiong, P. Fan, Z. Zhong, and K. B. Letaief, "Fog-Assisted Multiuser SWIPT Networks: Local Computing or Offloading," IEEE Internet of Things Journal, vol. 6, no. 3, pp. 5246–5264, June 2019.

[156] M. Ali, N. Riaz, M. I. Ashraf, S. Qaisar, and M. Naeem, "Joint Cloudlet Selection and Latency Minimization in Fog Networks," IEEE Transactions on Industrial Informatics, vol. 14, no. 9, pp. 4055–4063, Sep. 2018.

[157] A. A. Alli and M. M. Alam, "SecOFF-FCIoT: Machine learning based secure offloading in Fog-Cloud of things for smart city applications," Internet of Things, vol. 7, p. 100070, 2019.

[158] A. Brogi and S. Forti, "QoS-Aware Deployment of IoT Applications Through the Fog," IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1185–1192, Oct 2017.

[159] G. Lee, W. Saad, and M. Bennis, "An Online Optimization Framework for Distributed Fog Network Formation With Minimal Latency," IEEE Transactions on Wireless Communications, vol. 18, no. 4, pp. 2244–2258, April 2019.

[160] M. Adhikari and H. Gianey, "Energy efficient offloading strategy in fog-cloud environment for IoT applications," Internet of Things, vol. 6, p. 100053, 2019.

[161] A. Alnoman and A. Anpalagan, "A Dynamic Priority Service Provision Scheme for Delay-Sensitive Applications in Fog Computing," in 2018 29th Biennial Symposium on Communications (BSC). IEEE, June 2018, pp. 1–5.

[162] J. Du, L. Zhao, J. Feng, and X. Chu, "Computation Offloading and Resource Allocation in Mixed Fog/Cloud Computing Systems With Min-Max Fairness Guarantee," IEEE Transactions on Communications, vol. 66, no. 4, pp. 1594–1608, April 2018.

[163] A. J. Fahs and G. Pierre, "Proximity-Aware Traffic Routing in Distributed Fog Computing Platforms," in 2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), May 2019, pp. 478–487.

[164] S. K. Mishra, D. Puthal, J. J. P. C. Rodrigues, B. Sahoo, and E. Dutkiewicz, "Sustainable Service Allocation Using a Metaheuristic Technique in a Fog Server for Industrial Applications," IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4497–4506, Oct 2018.

[165] Y. Niu, Y. Liu, Y. Li, Z. Zhong, B. Ai, and P. Hui, "Mobility-Aware Caching Scheduling for Fog Computing in mmWave Band," IEEE Access, vol. 6, pp. 69 358–69 370, 2018.

[166] C. Puliafito, E. Mingozzi, C. Vallati, F. Longo, and G. Merlino, "Companion Fog Computing: Supporting Things Mobility Through Container Migration at the Edge," in 2018 IEEE International Conference on Smart Computing (SMARTCOMP), June 2018, pp. 97–105.

[167] L. Li, Q. Guan, L. Jin, and M. Guo, "Resource Allocation and Task Offloading for

Heterogeneous Real-Time Tasks With Uncertain Duration Time in a Fog Queueing System," IEEE Access, vol. 7, pp. 9912–9925, 2019.

[168] E. Melnik, A. Klimenko, and V. Klimenko, "A Recovery Technique for the Fog-Computing-Based Information and Control Systems," in Intelligent Systems in Cybernetics and Automation Control Theory, R. Silhavy, P. Silhavy, and Z. Prokopova, Eds. Cham: Springer International Publishing, 2019, pp. 216–227.

[169] D. Rahbari and M. Nickray, "Task offloading in mobile fog computing by classification and regression tree," Peer-to-Peer Networking and Applications, vol. 13, no. 1, pp. 104–122, 2020.

[170] M. Chen, W. Li, G. Fortino, Y. Hao, L. Hu, and I. Humar, "A Dynamic Service Migration Mechanism in Edge Cognitive Computing," ACM Trans. Internet Technol., vol. 19, no. 2, pp. 30:1–30:15, Apr. 2019.

[171] Y. Jiang, Y. Chen, S. Yang, and C. Wu, "Energy-Efficient Task Offloading for Time-Sensitive Applications in Fog Computing," IEEE Systems Journal, vol. 13, no. 3, pp. 2930–2941, Sep. 2019.

[172] H. Sun, H. Yu, G. Fan, and L. Chen, "Energy and time efficient task offloading and resource allocation on the generic iot-fog-cloud architecture," Peer-to-Peer Networking and Applications, vol. 13, no. 2, pp. 548–563, 2020.

[173] M. B. Kamal, N. Javaid, S. A. A. Naqvi, H. Butt, T. Saif, and M. D. Kamal, "Heuristic Min-conflicts Optimizing Technique for Load Balancing on Fog Computing," in Advances in Intelligent Networking and Collaborative Systems, F. Xhafa, L. Barolli, and M. Greguš, Eds. Cham: Springer International Publishing, 2019, pp. 207–219.

[174] G. Zhang, F. Shen, N. Chen, P. Zhu, X. Dai, and Y. Yang, "Dots: Delay-optimal task scheduling among voluntary nodes in fog networks," IEEE Internet of Things Journal, vol. 6, no. 2, pp. 3533–3544, April 2019.

[175] D. Zhao, G. Sun, D. Liao, S. Xu, and V. Chang, "Mobile-aware service function

chain migration in cloud–fog computing," Future Generation Computer Systems, vol. 96, pp. 591 – 604, 2019.

[176] S. Jošilo and G. Dán, "Decentralized Algorithm for Randomized Task Allocation in Fog Computing Systems," IEEE/ACM Transactions on Networking, vol. 27, no. 1, pp. 85–97, Feb 2019.

[177] J. Fan, X. Wei, T. Wang, T. Lan, and S. Subramaniam, "Deadline-Aware Task Scheduling in a Tiered IoT Infrastructure," in GLOBECOM 2017 - 2017 IEEE Global Communications Conference, Dec 2017, pp. 1–7.

[178] X. Wang, L. Wang, Y. Li, and K. Gai, "Privacy-Aware Efficient Fine-Grained Data Access Control in Internet of Medical Things Based Fog Computing," IEEE Access, vol. 6, pp. 47 657–47 665, 2018.

[179] A. Singh and N. Auluck, "Load balancing aware scheduling algorithms for fog networks," Software: Practice and Experience, vol. 0, no. 0, 2019.

[180] W. Wang, G. Wu, Z. Guo, L. Qian, L. Ding, and F. Yang, "Data Scheduling and Resource Optimization for Fog Computing Architecture in Industrial IoT," in Distributed Computing and Internet Technology, G. Fahrnberger, S. Gopinathan, and L. Parida, Eds. Cham: Springer International Publishing, 2019, pp. 141–149.

[181] D. Charântola, A. C. Mestre, R. Zane, and L. F. Bittencourt, "Component-based Scheduling for Fog Computing," in Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion, ser. UCC '19 Companion. New York, NY, USA: ACM, 2019, pp. 3–8.

[182] L. Shooshtarian, D. Lan, and A. Taherkordi, "A Clustering-Based Approach to Efficient Resource Allocation in Fog Computing," in Pervasive Systems, Algorithms and Networks, C. Esposito, J. Hong, and K.-K. R. Choo, Eds. Cham: Springer International Publishing, 2019, pp. 207–224.

[183] B. Jamil, M. Shojafar, I. Ahmed, A. Ullah, K. Munir, and H. Ijaz, "A job scheduling algorithm for delay and performance optimization in fog computing,"

Concurrency and Computation: Practice and Experience, vol. n/a, no. n/a, p. e5581, 2019.

[184] R. K. Behera, K. H. K. Reddy, and D. S. Roy, "A Novel Context Migration Model for Fog-Enabled Cross-Vertical IoT Applications," in International Conference on Innovative Computing and Communications, A. Khanna, D. Gupta, S. Bhattacharyya, V. Snasel, J. Platos, and A. E. Hassanien, Eds. Singapore: Springer Singapore, 2020, pp. 287–295.

[185] M. Mtshali, H. Kobo, S. Dlamini, M. Adigun, and P. Mudali, "Multi-Objective Optimization Approach for Task Scheduling in Fog Computing," in 2019 International Conference on Advances in Big Data, Computing and Data Communication Systems (icABCD), Aug 2019, pp. 1–6.

[186] P. Farhat, H. Sami, and A. Mourad, "Reinforcement R-learning model for time scheduling of on-demand fog placement," The Journal of Supercomputing, Oct 2019.

[187] S. Sharma and H. Saini, "A novel four-tier architecture for delay aware scheduling and load balancing in fog environment," Sustainable Computing: Informatics and Systems, vol. 24, p. 100355, 2019.

[188] S. Meixner, D. Schall, F. Li, V. Karagiannis, S. Schulte, and K. Plakidas, "Automatic application placement and adaptation in cloud-edge environments," in 2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Sep. 2019, pp. 1001–1008.

[189] M. Mishra, S. K. Roy, A. Mukherjee, D. De, S. K. Ghosh, and R. Buyya, "An energy-aware multi-sensor geo-fog paradigm for mission critical applications," Journal of Ambient Intelligence and Humanized Computing, Sep 2019.

[190] J. L. de Souza Toniolli and B. Jaumard, "Resource allocation for multiple workflows in cloud-fog computing systems," in Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion, ser. UCC '19 Companion. New York, NY, USA: ACM, 2019, pp. 77–84.

[191] Q. Zheng, J. Jin, T. Zhang, J. Li, L. Gao, and Y. Xiang, "Energy-Sustainable Fog System for Mobile Web Services in Infrastructure-Less Environments," IEEE Access, vol. 7, pp. 161 318–161 328, 2019.

[192] P. Bellavista, A. Corradi, L. Foschini, and D. Scotece, "Differentiated service/data migration for edge services leveraging container characteristics," IEEE Access, vol. 7, pp. 139 746–139 758, 2019.

[193] M. Adhikari, M. Mukherjee, and S. N. Srirama, "DPTO: A Deadline and Priority-aware Task Offloading in Fog Computing Framework Leveraging Multi-level Feedback Queueing," IEEE Internet of Things Journal, pp. 1–1, 2019.

[194] S. Ghosh, A. Mukherjee, S. K. Ghosh, and R. Buyya, "Mobi-IoST: Mobility-aware Cloud-Fog-Edge-IoT Collaborative Framework for Time-Critical Applications," IEEE Transactions on Network Science and Engineering, pp. 1–1, 2019.

[195] S. P. Singh, A. Sharma, and R. Kumar, "Design and exploration of load balancers for fog computing using fuzzy logic," Simulation Modelling Practice and Theory, p. 102017, 2019.

[196] X. He, Z. Tu, X. Xu, and Z. Wang, "Re-deploying Microservices in Edge and Cloud Environment for the Optimization of User-Perceived Service Quality," in Service-Oriented Computing, S. Yangui, I. Bouassida Rodriguez, K. Drira, and Z. Tari, Eds. Cham: Springer International Publishing, 2019, pp. 555–560.

[197] G. Li, J. Yan, L. Chen, J. Wu, Q. Lin, and Y. Zhang, "Energy Consumption Optimization With a Delay Threshold in Cloud-Fog Cooperation Computing," IEEE Access, vol. 7, pp. 159 688–159 697, 2019.

[198] S. K. Battula, S. Garg, R. K. Naha, P. Thulasiraman, and R. Thulasiram, "A Micro-Level Compensation-Based Cost Model for Resource Allocation in a Fog Environment," Sensors, vol. 19, no. 13, 2019.

[199] L. Yu, T. Jiang, and Y. Zou, "Fog-assisted Operational Cost Reduction for Cloud Data Centers," IEEE Access, vol. 5, pp. 13 578–13 586, 2017.

[200] M. Aazam and E.-N. Huh, "Fog Computing Micro Datacenter Based Dynamic Resource Estimation and Pricing Model for IoT," in Advanced Information Networking and Applications (AINA), 2015 IEEE 29th International Conference on. IEEE, 2015, pp. 687–694.

[201] C. Huang and K. Xu, "Reliable realtime streaming in vehicular cloud-fog computing networks," in 2016 IEEE/CIC International Conference on Communications in China (ICCC), July 2016, pp. 1–6.

[202] W. Yánez, R. Mahmud, R. Bahsoon, Y. Zhang, and R. Buyya, "Data allocation mechanism for internet of things systems with blockchain," IEEE Internet of Things Journal, pp. 1–1, 2020.

[203] A. da Silva Veith, F. R. de Souza, M. D. de Assunção, L. Lefèvre, and J. C. S. dos Anjos, "Multi-Objective Reinforcement Learning for Reconfiguring Data Stream Analytics on Edge Computing," in 48th International Conference on Parallel Processing. ACM, 2019, pp. 106:1–106:10.

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

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

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

[207] Y. Nan, W. Li, W. Bao, F. C. Delicato, P. F. Pires, and A. Y. Zomaya, "A Dynamic Tradeoff Data Processing Framework for Delay-sensitive Applications in Cloud of Things Systems," Journal of Parallel and Distributed Computing, vol. 112, pp. 53–66, 2018.

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

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

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

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

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

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

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

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

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

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

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

[219] R. Mahmud and R. Buyya, Modeling and Simulation of Fog and Edge Computing Environments Using iFogSim Toolkit. John Wiley & Sons, Ltd, 2019, ch. 17, pp. 433–465.

[220] S. Yangui, P. Ravindran, O. Bibani, R. H. Glitho, N. B. Hadj-Alouane, M. J. Morrow, and P. A. Polakos, "A platform as-a-service for hybrid cloud/fog environments," in IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN). IEEE, 2016, pp. 1–7.

[221] Y. Kang, Z. Zheng, and M. R. Lyu, "A latency-aware co-deployment mechanism for cloud-based services," in 2012 IEEE Fifth International Conference on Cloud Computing, June 2012, pp. 630–637.

[222] T. Nishio, R. Shinkuma, T. Takahashi, and N. B. Mandayam, "Service-oriented heterogeneous resource sharing for optimizing service latency in mobile cloud," in Proceedings of the First International Workshop on Mobile Cloud Computing & Networking, ser. MobileCloud '13. New York, NY, USA: ACM, 2013, pp. 19–26.

[223] B. Ottenwälder, B. Koldehofe, K. Rothermel, and U. Ramachandran, "Migcep: Operator migration for mobility driven distributed complex event processing," in Proceedings of the 7th ACM International Conference on Distributed Event-based Systems, ser. DEBS '13. New York, NY, USA: ACM, 2013, pp. 183–194.

[224] I. Takouna, R. Rojas-Cessa, K. Sachs, and C. Meinel, "Communication-aware and energy-efficient scheduling for parallel applications in virtualized data centers," in 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing, Dec 2013, pp. 251–255.

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

[226] J. Famaey, W. De Cock, T. Wauters, F. De Turck, B. Dhoedt, and P. Demeester, "A latency-aware algorithm for dynamic service placement in large-scale overlays," in Integrated Network Management, 2009. IM'09. IFIP/IEEE International Symposium on. IEEE, 2009, pp. 414–421.

[227] I. Gupta, M. S. Kumar, and P. K. Jana, "Transfer time-aware workflow scheduling for multi-cloud environment," in Computing, Communication and Automation (ICCCA), 2016 International Conference on. IEEE, 2016, pp. 732–737.

[228] Y. Fan, J. Chen, L. Wang, and Z. Cao, Energy-Efficient and Latency-Aware Data Placement for Geo-Distributed Cloud Data Centers. Cham: Springer International Publishing, 2018, pp. 465–474.

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

[230] V. Chamola, C.-K. Tham, and G. S. Chalapathi, "Latency aware mobile task assignment and load balancing for edge cloudlets," in Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on. IEEE, 2017, pp. 587–592.

[231] C. Xu, S. Gamage, P. N. Rao, A. Kangarlou, R. R. Kompella, and D. Xu, "vslicer: Latency-aware virtual machine scheduling via differentiated-frequency cpu slicing," in Proceedings of the 21st International Symposium on High-Performance Parallel and Distributed Computing, ser. HPDC '12. New York, NY, USA: ACM, 2012, pp. 3–14.

[232] P. S. Addison, J. N. Watson, M. L. Mestek, J. P. Ochs, A. A. Uribe, and S. D. Bergese,

"Pulse oximetry-derived respiratory rate in general care floor patients," Journal of Clinical Monitoring and Computing, vol. 29, no. 1, pp. 113–120, Feb 2015.

[233] M. Sahani, C. Nanda, A. K. Sahu, and B. Pattnaik, "Web-based online embedded door access control and home security system based on face recognition," in International Conference onCircuit, Power and Computing Technologies (ICCPCT). IEEE, 2015, pp. 1–6.

[234] T. H. Ashrafi, M. A. Hossain, S. E. Arefin, K. D. J. Das, and A. Chakrabarty, IoT Infrastructure: Fog Computing Surpasses Cloud Computing. Springer Singapore, 2018, pp. 43–55.

[235] M. Slabicki and K. Grochla, "Performance evaluation of coap, snmp and netconf protocols in fog computing architecture," in NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium, April 2016, pp. 1315–1319.

[236] J. Kempf, J. Arkko, N. Beheshti, and K. Yedavalli, "Thoughts on reliability in the internet of things," in Interconnecting smart objects with the Internet workshop, vol. 1, 2011, pp. 1–4.

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

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

[239] W. Lin, C. Liang, J. Z. Wang, and R. Buyya, "Bandwidth-aware divisible task scheduling for cloud computing," Software: Practice and Experience, vol. 44, no. 2, pp. 163–174, 2014.

[240] R. Mahmud, M. Afrin, M. A. Razzaque, M. M. Hassan, A. Alelaiwi, and M. Alrubaian, "Maximizing quality of experience through context-aware mobile application

scheduling in cloudlet infrastructure," Software: Practice and Experience, vol. 46, no. 11, pp. 1525–1545, 2016, spe.2392.

[241] H. Lasi, P. Fettke, H.-G. Kemper, T. Feld, and M. Hoffmann, "Industry 4.0," Business & Information Systems Engineering, vol. 6, no. 4, pp. 239–242, Aug 2014.

[242] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, Opportunities, and Directions," IEEE Transactions on Industrial Informatics, vol. 14, no. 11, pp. 4724–4734, Nov 2018.

[243] B. Chen, J. Wan, L. Shu, P. Li, M. Mukherjee, and B. Yin, "Smart Factory of Industry 4.0: Key Technologies, Application Case, and Challenges," IEEE Access, vol. 6, pp. 6505–6519, Dec 2018.

[244] M. Aazam, S. Zeadally, and K. A. Harras, "Deploying Fog Computing in Industrial Internet of Things and Industry 4.0," IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4674–4682, Oct 2018.

[245] K. Sato and S. Azuma, "Secure Real-Time Control Through Fog Computation," IEEE Transactions on Industrial Informatics, vol. 15, no. 2, pp. 1017–1026, Feb 2019.

[246] A. Singhvi, S. Banerjee, Y. Harchol, A. Akella, M. Peek, and P. Rydin, "Granular Computing and Network Intensive Applications: Friends or Foes?" in 16th ACM Workshop on Hot Topics in Networks, Palo Alto, CA, USA, 2017, pp. 157–163.

[247] M. Haferkamp, B. Sliwa, C. Ide, and C. Wietfeld, "Payload-Size and Deadline-Aware scheduling for time-critical Cyber Physical Systems," in Wireless Days, Porto, Portugal, Mar 2017, pp. 4–7.

[248] Q. T. Minh, E. Kamioka, and S. Yamada, "CFC-ITS: Context-Aware Fog Computing for Intelligent Transportation Systems," IT Professional, vol. 20, no. 6, pp. 35–45, Nov 2018.

[249] J. Lee, K. Lee, E. Jeong, J. Jo, and N. B. Shroff, "CAS: Context-Aware Background Application Scheduling in Interactive Mobile Systems," IEEE Journal on Selected Areas in Communications, vol. 35, no. 5, pp. 1013–1029, May 2017.

[250] B. Gu, X. Wang, Y. Qu, J. Jin, Y. Xiang, and L. Gao, "Context-Aware Privacy Preservation in a Hierarchical Fog Computing System," in IEEE International Conference on Communications, Shanghai, China, May 2019, pp. 1–6.

[251] X. Yao, H. Kong, H. Liu, T. Qiu, and H. Ning, "An Attribute Credential Based Public Key Scheme for Fog Computing in Digital Manufacturing," IEEE Transactions on Industrial Informatics, vol. 15, no. 4, pp. 2297–2307, Apr 2019.

[252] C. Lin and J. Yang, "Cost-Efficient Deployment of Fog Computing Systems at Logistics Centers in Industry 4.0," IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4603–4611, Oct 2018.

[253] D. A. Chekired, L. Khoukhi, and H. T. Mouftah, "Industrial IoT Data Scheduling Based on Hierarchical Fog Computing: A Key for Enabling Smart Factory," IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4590–4602, Oct 2018.

[254] J. Wan, B. Chen, S. Wang, M. Xia, D. Li, and C. Liu, "Fog Computing for Energy-Aware Load Balancing and Scheduling in Smart Factory," IEEE Transactions on Industrial Informatics, vol. 14, no. 10, pp. 4548–4556, Oct 2018.

[255] A. Kumari, S. Tanwar, S. Tyagi, and N. Kumar, "Fog computing for Healthcare 4.0 environment: Opportunities and challenges," Computers & Electrical Engineering, vol. 72, pp. 1 – 13, Nov 2018.

[256] N. Ahmed, D. De, and I. Hussain, "Internet of Things (IoT) for Smart Precision Agriculture and Farming in Rural Areas," IEEE Internet of Things Journal, vol. 5, no. 6, pp. 4890–4899, Dec 2018.

[257] S. Pešić, M. Tošić, O. Iković, M. Ivanović, M. Radovanović, and D. Bošković, "Context Aware Resource and Service Provisioning Management in Fog Computing Systems," in International Symposium on Intelligent and Distributed Computing, Belgrade, Serbia, 2017, pp. 213–223.

[258] P. Moore and H. Van Pham, "FOG Computing and Low Latency Context-Aware Health Monitoring in Smart Interconnected Environments," in International

Conference on Emerging Internetworking, Data & Web Technologies, Tirana, Albania, 2018, pp. 29–40.

[259] B. Afzal, S. A. Alvi, G. A. Shah, and W. Mahmood, "Energy efficient context aware traffic scheduling for iot applications," Ad Hoc Networks, vol. 62, pp. 101 – 115, Jul 2017.

[260] B. Gu, Y. Chen, H. Liao, Z. Zhou, and D. Zhang, "A Distributed and Context-Aware Task Assignment Mechanism for Collaborative Mobile Edge Computing," Sensors, vol. 18, no. 8, p. 2423, Jul 2018.

[261] S. Garg, A. Singh, K. Kaur, G. S. Aujla, S. Batra, N. Kumar, and M. S. Obaidat, "Edge Computing-Based Security Framework for Big Data Analytics in VANETs," IEEE Network, vol. 33, no. 2, pp. 72–81, Mar 2019.

[262] M. M. Tajiki, M. Shojafar, B. Akbari, S. Salsano, and M. Conti, "Software defined service function chaining with failure consideration for fog computing," Concurrency and Computation: Practice and Experience, vol. 31, no. 8, p. e4953, Apr 2019.

[263] R. Oma, S. Nakamura, D. Duolikun, T. Enokido, and M. Takizawa, "Energy-Efficient Recovery Algorithm in the Fault-Tolerant Tree-Based Fog Computing (FTBFC) Model," in International Conference on Advanced Information Networking and Applications, Matsue, Japan, 2019, pp. 132–143.

[264] M. Afrin, J. Jin, A. Rahman, Y.-C. Tian, and A. Kulkarni, "Multi-objective resource allocation for Edge Cloud based robotic workflow in smart factory," Future Generation Computer Systems, vol. 97, pp. 119 – 130, Aug 2019.

[265] A. K. Vishwakarma and A. Mishra, "Color Image Enhancement Techniques: A Critical Review," Indian Journal of Computer Science and Engineering, vol. 3, no. 1, pp. 39–45, Mar 2012.

[266] R. Longbottom, "Roy Longbottom's PC Benchmark Collection." [Online]. Available: http://www.roylongbottom.org.uk

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

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

[269] K. u. R. Laghari, N. Crespi, B. Molina, and 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, ser. WAINA '11. IEEE, March 2011, pp. 837–842.

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

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

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

[273] L. Li, M. Rong, and 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, ser. ICCSE '15. IEEE, 2015, pp. 925–928.

[274] M. Fiedler, T. Hossfeld, and P. Tran-Gia, "A Generic Quantitative Relationship between QoE and QoS," IEEE Network, vol. 24, no. 2, pp. 36–41, March 2010.

[275] X. Zhou, M. Sun, Y. Wang, and 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, ser. QSHINE '15. IEEE, 2015, pp. 122–126.

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

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

[278] X. Lin, W. Wu, Y. Zhu, T. Qiu, and 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, ser. ICUWB '16. IEEE, 2016, pp. 1–4.

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

[280] N. Iotti, M. Picone, S. Cirani, and G. Ferrari, "Improving Quality of Experience in Future Wireless Access Networks through Fog Computing," IEEE Internet Computing, vol. 21, no. 2, pp. 26–33, Mar 2017.

[281] E. Recommendation, "Definitions of Terms Related to QoS," 2008.

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

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

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

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

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

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

[288] M. Taneja and 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, ser. IM '15. IEEE, 2017, pp. 1222–1228.

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

[290] K. Hong, D. Lillethun, U. Ramachandran, B. Ottenwälder, and B. Koldehofe, "Mobile fog: A programming model for large-scale applications on the internet of things," in Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing. ACM, 2013, pp. 15–20.

[291] C. Chang, S. N. Srirama, and R. Buyya, "Indie Fog: An Efficient Fog-Computing Infrastructure for the IoT," Computer, vol. 50, no. 9, pp. 92–98, 2017.

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

[293] E. Mamdani and S. Assilian, "An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller," International Journal of Man-Machine Studies, vol. 7, no. 1, pp. 1 – 13, 1975.

[294] A. Ansari and 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, ser. ICAIET '14. IEEE, 2014, pp. 31–36.

[295] G. Chiandussi, M. Codegone, S. Ferrero, and F. E. Varesio, "Comparison of Multi-objective Optimization Methodologies for Engineering Applications," Computers & Mathematics with Applications, vol. 63, no. 5, pp. 912–942, 2012.

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

[297] Y. Elkhatib, B. Porter, H. B. Ribeiro, M. F. Zhani, J. Qadir, and E. Rivière, "On Using Micro-clouds to Deliver the Fog," IEEE Internet Computing, vol. 21, no. 2, pp. 8–15, 2017.

[298] P. Bellavista and 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.

[299] F. M. Insights, "Fog Computing Market: Global Industry Analysis and Opportunity Assessment 2017-2027," https://www.futuremarketinsights.com/reports/fog-computingmarket, 2017, [Online; accessed 23-June-2018].

[300] L. Gu, D. Zeng, S. Guo, A. Barnawi, and Y. Xiang, "Cost Efficient Resource Management in Fog Computing Supported Medical Cyber-physical System," IEEE Transactions on Emerging Topics in Computing, vol. 5, no. 1, pp. 108–119, 2017.

[301] M. Afrin, M. A. Razzaque, I. Anjum, M. M. Hassan, and A. Alamri, "Tradeoff between user quality-of-experience and service provider profit in 5G cloud radio access network," Sustainability, vol. 9, no. 11, p. 2127, 2017.

[302] Y. Lin and H. Shen, "CloudFog: Leveraging fog to extend cloud gaming for thin-client MMOG with high quality of service," IEEE Transactions on Parallel & Distributed Systems, vol. 28, no. 2, pp. 431–445, 2017.

[303] N. Raveendran, H. Zhang, Z. Zheng, L. Song, and Z. Han, "Large-scale Fog Computing Optimization Using Equilibrium Problem with Equilibrium Constraints," in GLOBECOM 2017-2017 IEEE Global Communications Conference. IEEE, 2017, pp. 1–6.

[304] C. Liu, K. Li, C. Xu, and K. Li, "Strategy configurations of multiple users competition for cloud service reservation," IEEE Transactions on Parallel and Distributed Systems, vol. 27, no. 2, pp. 508–520, 2015.

[305] C. Liu, K. Li, and K. Li, "Minimal cost server configuration for meeting time-varying resource demands in cloud centers," IEEE Transactions on Parallel and Distributed Systems, vol. 29, no. 11, pp. 2503–2513, 2018.

[306] X.-Q. Pham, N. D. Man, N. D. T. Tri, N. Q. Thai, and E.-N. Huh, "A Cost-and Performance-effective Approach for Task Scheduling based on Collaboration between Cloud and Fog Computing," International Journal of Distributed Sensor Networks, vol. 13(11), 2017.

[307] L. Yang, J. Cao, G. Liang, and X. Han, "Cost aware Service Placement and Load Dispatching in Mobile Cloud Systems," IEEE Transactions on Computers, vol. 65, no. 5, pp. 1440–1452, 2016.

[308] H. Yao, C. Bai, M. Xiong, D. Zeng, and Z. Fu, "Heterogeneous Cloudlet Deployment and User-cloudlet Association Toward Cost Effective Fog Computing," Concurrency and Computation: Practice and Experience, vol. 29, no. 16, 2017.

[309] A. Kiani and N. Ansari, "Toward Hierarchical Mobile Edge Computing: An

Auction-Based Profit Maximization Approach," IEEE Internet of Things Journal, vol. 4, no. 6, pp. 2082–2091, 2017.

[310] L. Ni, J. Zhang, C. Jiang, C. Yan, and K. Yu, "Resource allocation strategy in fog computing based on priced timed petri nets," IEEE Internet of Things Journal, vol. 4, no. 5, pp. 1216–1228, 2017.

[311] A. S. Sohal, R. Sandhu, S. K. Sood, and V. Chang, "A cybersecurity framework to identify malicious edge device in fog computing and cloud-of-things environments," Computers & Security, vol. 74, pp. 340–354, 2018.

[312] D. Rosário, M. Schimuneck, J. Camargo, J. Nobre, C. Both, J. Rochol, and M. Gerla, "Service migration from cloud to multi-tier fog nodes for multimedia dissemination with QoE support," Sensors, vol. 18, no. 2, p. 329, 2018.

[313] M. Al-khafajiy, T. Baker, H. Al-Libawy, A. Waraich, C. Chalmers, and O. Alfandi, "Fog Computing Framework for Internet of Things Applications," in 2018 11th International Conference on Developments in eSystems Engineering (DeSE). IEEE, 2018, pp. 71–77.

[314] D. Zhao, G. Sun, D. Liao, S. Xu, and V. Chang, "Mobile-aware service function chain migration in cloud–fog computing," Future Generation Computer Systems, vol. 96, pp. 591–604, 2019.

[315] C. Wang, Y. Zhu, W. Shi, V. Chang, P. Vijayakumar, B. Liu, Y. Mao, J. Wang, and Y. Fan, "A dependable time series analytic framework for cyber-physical systems of iot-based smart grid," ACM Transactions on Cyber-Physical Systems, vol. 3, no. 1, p. 7, 2018.

[316] A. T. Coughlan and K. Joseph, "26 sales force compensation: research insights and research potential," Handbook on business to business marketing, vol. 473, 2012.

[317] K. Barbosa, A. Bucione, and A. P. Souza, "Performance-based compensation vs. guaranteed compensation: contractual incentives and performance in the Brazilian banking industry," Economia Aplicada, vol. 18, no. 1, pp. 5–33, 2014.

[318] C. Pahl, S. Helmer, L. Miori, J. Sanin, and B. Lee, "A container-based edge cloud paas architecture based on raspberry pi clusters," in Future Internet of Things and Cloud Workshops (FiCloudW), IEEE International Conference on. IEEE, 2016, pp. 117–124.

[319] Z. Hoque, Handbook of cost and management accounting. Spiramus Press Ltd, 2005.

[320] N. Hogue, "Service level agreements with penalty clause," South Carolina State Documents Depository, 2009.