

**ORCHESTRATION MECHANISMS FOR ENABLING
DISTRIBUTED PROCESSING IN THE FOG COMPUTING
ENVIRONMENT**

Thesis

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

JOHN PAUL MARTIN



DEPARTMENT OF MATHEMATICAL AND COMPUTATIONAL SCIENCES
NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA,
SURATHKAL, MANGALORE - 575 025

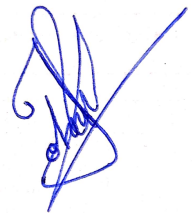
OCTOBER 2020

*”Experience is what you get when you didn’t get
what you wanted.”*

DECLARATION

by the Ph.D. Research Scholar

I hereby declare that the Research Thesis entitled **Orchestration Mechanisms for Enabling Distributed Processing in the Fog Computing Environment** which is being submitted to the **National Institute of Technology Karnataka, Surathkal** in partial fulfilment of the requirements for the award of the Degree of **Doctor of Philosophy** in Department of Mathematical and Computational Sciences is a bonafide report of the research work carried out by me. The material contained in this Research Thesis has not been submitted to any University or Institution for the award of any degree.



John Paul Martin, 165093 MA16FV01

Department of Mathematical and Computational Sciences

Place: NITK, Surathkal.

Date:05-10-2020


CERTIFICATE

This is to *certify* that the Research Thesis entitled **Orchestration Mechanisms for Enabling Distributed Processing in the Fog Computing Environment** submitted by **John Paul Martin**, (Reg. No.: 165093 MA16FV01) as the record of the research work carried out by him, is *accepted as the Research Thesis submission* in partial fulfillment of the requirements for the award of degree of **Doctor of Philosophy**.



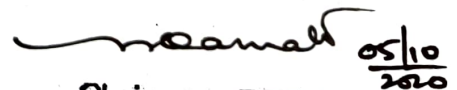
Prof. A Kandasamy

Research Guide



Prof. K. Chandrasekaran

Research Co-Guide



Chairman - DRPC

(Signature with ~~Chairman~~ Seal)

DUGC / DPCC / DRPO

Dept. of Mathematical and Computational Sciences
National Institute of Technology Karnataka, Surathkal
MANGALORE - 575 025

ACKNOWLEDGEMENTS

“It always seems impossible until its done.” - Nelson Mandela

At the foremost, I would like to place on record my sincere gratitude towards my supervisors, Prof . A. Kandasamy, Department of Mathematical and Computational Sciences, NITK and Prof. K. Chandrasekaran, Department of Computer Science and Engineering, NITK, for stimulating me to identify the impossible and make it possible.

I take this opportunity to remember with gratefulness, my Research Progress Assessment Committee members, Dr. N. Shekar V. Shet, Department of Electronics and Communication Engineering, NITK and Dr. Pushparaj Shetty D., Department of Mathematical and Computational Sciences, NITK for their valuable perusal throughout the progress of my research. I would like to express, in particular, my gratitude towards Prof. Shyam S. Kamath, Head of the Department of Mathematical and Computational Sciences, and to the entire faculty members, support staff and fellow research scholars. I am thankful to NITK Surathkal for providing me with all the necessary infrastructure to carry out my research works. I acknowledge the Visvesvaraya PhD Scheme of Ministry of Electronics & Information Technology, Government of India, implemented by Digital India Corporation, for providing scholarships and amenities to pursue my doctoral research. A special thanks to my overseas research coeval Joseph John, A.N.U, Australia.

I am extremely thankful to my parents and my better half Christina Joseph, who stood by me as the pivotal impetus that helped me to overcome even the greatest setbacks and to consistently strive to achieve this glorious goal.

Above all, I thank God Almighty for the innumerable blessings showered upon me throughout my life.

John Paul Martin

ABSTRACT

The recent years witnessed the rapid adoption of the Internet of Things (IoT) paradigm across business and non-business realms alike. Usually, IoT-based systems are located at a multi-hop distance apart from the Cloud datacenters. Consequently, relying on Cloud-centric execution results in a performance penalty for the real-time IoT applications. To circumvent this, the Fog computing paradigm emanated as a widespread computing technology to support the execution of the IoT applications. Fog computing extends Cloud services to the vicinity of end devices, thereby enabling the applications to be executed closer to the data sources. Thus, the Fog paradigm aids in reducing the service delivery time and network congestion. Even though this new paradigm opens a set of potential possibilities, it also introduces several additional challenges and complexities arising from its heterogeneity and resource-constrained characteristics. In order to harness the potential of Fog computing environments, it is imperative to adopt efficient orchestration mechanisms that can manage the resources in the system.

Application management is an intrinsic component of resource orchestration systems. This involves identifying suitable options for the initial placement of the applications. The placement decisions have significant impacts on the overall performance of the application. Placement schemes for Fog environments must take into consideration the requirements and characteristics of the different entities in the Fog ecosystem, including the Fog nodes, IoT applications and IoT devices. The category of mission critical IoT applications makes reliable service delivery an essential requirement in Fog environments. There is a resolute need for placement schemes that ensure reliable service delivery. Accordingly, in this research, a placement policy that addresses the conflicting criteria of service reliability and monetary cost is proposed. The proposed Cost and Reliability aware Eagle Whale optimizer (CREW) derives placement decisions that maximize the reliability and minimize the cost. Realtime experiments substantiate that the proposed approach succeeds in improving the performance of applications executed in Fog computing environments.

User mobility is another particularity in Fog computing environments. Mobility of the end users may result in an increase in the hop distance between the data source and the Fog node. In order to ensure that this does not adversely impact service delivery, application modules must be migrated across Fog nodes with the objective of maintaining low hop distance values. Therefore, this research also presents a Mobility-aware Autonomic Framework (MAMF) to perform migrations in the Fog environment. The developed framework relies on predetermined user locations to take migration decisions. Performance of the approach is evaluated using real-time mobility traces.

Keywords: Fog Computing, Service placement, Application module migration, Reliability, Mobility support, Meta-heuristic optimization, Autonomic Computing

CONTENTS

List of Figures	viii
List of Tables	ix
List of Abbreviations	xi
1 Introduction	1
1.1 Related Computing Paradigms	2
1.1.1 Cloud Computing	3
1.1.2 Edge Computing	3
1.1.3 Cloudlet	4
1.1.4 Mobile Edge Computing	4
1.1.5 Mobile Cloud Computing	5
1.1.6 Mist Computing	6
1.2 Background	6
1.2.1 Purpose and scope of Fog Computing	7
1.2.2 Fog Computing Paradigm	8
1.2.2.1 Fog Nodes	8
1.2.3 Fog Computing Characteristics	10
1.2.4 Fog Computing Reference Architecture	13
1.2.5 Virtualization in Fog computing environments	14
1.2.5.1 Container Virtualization	15
1.2.6 Fog Computing Environment	16
1.3 Motivation	18
1.4 Organization of the Thesis	19

2	Literature Review	23
2.1	Fog computing aspects	23
2.1.1	Application Domains	24
2.1.2	Platform Management	28
2.1.2.1	Communication Models and Technologies	29
2.1.2.2	Orchestration and Coordination	32
2.1.2.3	Programming Models / Frameworks	38
2.1.2.4	Security and Privacy Issues in Fog environments	39
2.2	Outcome of Literature Review	41
2.3	Summary	43
3	Problem Description	45
3.1	Scope and Focus of the thesis	45
3.2	Research Problem and Objectives	47
3.2.1	Research Objectives	48
3.3	Research Methodology	48
3.4	Research Contributions	49
4	Cost and Reliability aware Eagle-Whale optimizer for Service Placement in Fog	53
4.1	System Architecture	55
4.1.1	Service Execution Time Prediction	59
4.1.2	Waiting Time Model	61
4.1.3	Communication Time Estimation	62
4.1.4	Response Time Estimation	62
4.1.5	Reliability Model	63
4.1.6	Cost Model	64
4.2	Problem Formulation	65
4.3	CREW-Cost and Reliability-aware Eagle-Whale Optimizer	66
4.3.1	Whale optimisation Algorithm	66
4.3.2	Eagle Strategy	68

4.4	Experimental Design and Setup	71
4.4.1	Workload applications considered	72
4.4.1.1	E-commerce Industrial Applications	72
4.4.1.2	Healthcare Application	74
4.4.2	Performance Metrics	75
4.5	Experimental Results and Analysis	76
4.5.1	Performance comparison of various Placement Schemes	77
4.5.1.1	Execution Cost	78
4.5.1.2	Reliability	80
4.5.1.3	Response Time	82
4.5.2	Performance Comparison of various Machine Learning Tech- niques for execution time prediction	84
4.5.3	Performance Analysis in Large Search Spaces	85
4.6	Summary	86
5	Mobility aware autonomic approach for the migration of application mod- ules in Fog Computing Environment	87
5.1	Motivation	88
5.2	System Model	89
5.2.1	Optimization Model	92
5.2.2	Model Example	94
5.3	Proposed Conceptual Framework	96
5.3.1	MAPE Control Loop	98
5.3.1.1	Monitoring Phase	99
5.3.1.2	Analyse Phase	100
5.3.1.3	Planning Phase	100
5.3.1.4	Execution Phase	102
5.3.2	Genetic Algorithm	102
5.3.2.1	Chromosome Representation	103
5.3.2.2	Crossover and mutation operator	104
5.3.2.3	Fitness function, selection operator	104

5.3.2.4	Stopping Criteria	105
5.4	Experimental Evaluation	105
5.4.1	Evaluation Environment	105
5.4.2	Simulation Parameter Settings	106
5.4.3	Mobility Trace Description	106
5.4.4	Baseline Approach	106
5.4.5	Evaluation metrics	108
5.5	Results	110
5.5.1	Network Usage	110
5.5.2	Average Loop Delay	111
5.5.3	Average Tuple Execution Delay	112
5.5.4	Execution cost in Cloud	112
5.6	Discussion	114
5.6.1	Location value at time $(t + 1)$	114
5.6.1.1	Evaluation criteria	116
5.6.2	Evaluation of GA	116
5.6.2.1	Evolution of Objective Function	117
5.6.3	Integer Linear Programming & Heuristic Approach	121
5.7	Summary	121
6	Conclusions and Future Scope	123
6.1	Future Scope	124
	Bibliography	126
	Research Outcomes	146

LIST OF FIGURES

1.1	Fog Reference Architecture (Cisco 2016).	13
1.2	Hypervisor-based Virtualization	16
1.3	Container Virtualization	16
1.4	Fog Computing Environment	17
1.5	Organization of the thesis	20
2.1	Classification of works considering different aspects in Fog Computing	25
3.1	An orchestration framework for Fog computing Environment	46
3.2	Overview of our Research Methodology	49
3.3	Contributions of this Thesis	50
4.1	Three tier Fog computing Environment	56
4.2	System architecture	57
4.3	Service Execution Time Prediction process	60
4.4	Model of the inter-connected queuing system in hierarchical Fog environments	62
4.5	Encoding Scheme for the Service Placement Problem	70
4.6	Interactions between service components in Hipster Shop Application .	73
4.7	Interactions between service components in Tea Store Application . . .	74
4.8	Interactions between service components in Smart-Health Application .	75
4.9	Comparison of Monetary Costs of Execution with varying placement schemes	79
4.10	Comparison of Reliability values with varying placement schemes . . .	81
4.11	Comparison of Response Time values with varying placement schemes	83
4.12	Performance analysis in large search space with varying placement schemes	85

5.1	Problem Scenario	90
5.2	MAMF framework based on control MAPE loop	97
5.3	Chromosome representation example for a system with four fog nodes and six application components	103
5.4	Network Usage	111
5.5	Average Loop Delay	112
5.6	Average Tuple Delay	113
5.7	Execution Cost in Cloud	113
5.8	MAE, MAPE and RMSE value comparison for different prediction tech- niques	115
5.9	Convergence of Time to Completion (TTC) across generations for dif- ferent configurations	117
5.10	Evolution of Time to Completion with Distance across Generations . . .	118
5.11	Variation of Time to Completion and deadline(delay) with number of migrating modules	120

LIST OF TABLES

1.1	Comparison between different processing environments	12
2.1	Different usecases employing Fog Computing	26
2.2	Research works on Placement	43
2.3	Research works on Migration	43
3.1	Scope and Focus of this Thesis	46
3.2	Details of Contributions of this Thesis	52
4.1	Notations used in the System model	58
4.2	Characteristics of Service Components of Hipster Shop Application . . .	73
4.3	Characteristics of Service Components of Tea Store Application	74
4.4	Characteristics of Service Components of Smart-Health Application . . .	75
4.5	Parameter values for different characteristics of Cloud and Fog devices .	76
4.6	Mean Hypervolume Values over 30 independent runs	78
4.7	Parameter settings for ML-based prediction techniques	84
4.8	Comparison of prediction accuracy values for varying ML-based ser- vice execution time prediction techniques	85
5.1	Notations used in the system model	91
5.2	Mathematical Model Solution	95
5.3	Experimental details	106
5.4	Comparison of Solutions	121

LIST OF ABBREVIATIONS

<u>Abbreviations</u>	<u>Expansion</u>
API	Application Programming Interface
AR	Autoregression
ARIMA	AutoRegressive Integrated Moving Average
CDN	Content Distribution Network
CRAN	Cloud Radio Access Network
DES	Double Exponential Smoothing
ECG	Electrocardiogram
ES	Exponential Smoothing
FCFS	First Come First Serve
GA	Genetic Algorithm
GCP	Google Cloud Platform
HV	Hyper Volume
ILP	Integer Linear Programming
IoE	Internet of Everything
IoT	Internet of Things
ITS	Intelligent Transportation System
MA	Moving Average
MAE	Mean Absolute Error

<u>Abbreviations</u>	Expansion
MAPE	Monitor-Analyze-Plan-Execute
MCC	Mobile Cloud Computing
MEC	Mobile Edge Computing
MOWOA	Multi-objective Whale Optimization Algorithm
NAT	Network Address Translation
NFV	Network Function Virtualization
NSGA	Nondominated Sorting Genetic Algorithm
OS	Operating System
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
REST	Representational State Transfer
RMSE	Root Mean Squared Error
SDN	Software-Defined Networking
SLA	Service Level Agreement
SVR	Support Vector Regression
VM	Virtual Machine
WOA	Whale optimisation Algorithm
XGBOOST	eXtreme Gradient Boost

CHAPTER 1

INTRODUCTION

Prevailing pervasive and ubiquitous computing technologies are directed towards making the world totally connected. The constantly available, networked computing devices empower end users by providing them access to a range of cost-effective and high-performance services. These smart devices have the potential to interconnect distinct physical business worlds, and their usage in real time complex applications helps to generate efficient, comfortable and simple technological solutions. The widespread emergence of the Internet of Things (IoT) paradigm, has led to a proliferation of IoT devices connected to the Internet in all the sectors encompassing the public and private business spaces.

According to recent estimates, the number of devices connected to the Internet is expected to be around 3.6 per capita by 2022 and around 71% of the global traffic will be generated by wireless devices (Cisco 2018). The enormous number of connected IoT devices generate huge volumes of heterogeneous data. However, the limited computing and processing capabilities of the devices generating this data, deem them insufficient for processing the large amount of data generated. To overcome such issues, these devices at the network edge rely on the Cloud data centres. Data from the edge devices is uploaded to the Cloud, where all the processing activities are carried out and the results of the processing activities are sent back to the edge devices. Usage of the Cloud provides efficient processing functionalities and unlimited storage for the large

volume of heterogeneous data. However, the latencies involved in uploading the entire data to the Cloud, and the bandwidth requirements are the downsides. This leads to a degradation in the user experience.

Many efforts have been directed towards the integration of the Cloud and the IoT. Nevertheless, there is a need to devise approaches that fulfil all the requirements of IoT applications which include low latency, mobility support and location-aware processing (Barcelo et al. 2016; Qiu et al. 2018). In an effort to provide better suitability for such applications, researchers propounded to carry out processing activities nearer to the end users, in the edge devices. In the Edge Computing paradigm, the edge devices are augmented with data processing capabilities rather than holding the processing power solely in a quasi-central Cloud. This facilitates preliminary processing to be carried out in the edge resources, thereby reducing processing latencies. Nonetheless, the Edge Computing paradigm also had several drawbacks such as resource contention issues, security issues and limited support for scalability.

The limitations in Edge Computing and Cloud Computing for performing real-time operations led researchers to propose a new computing model that can overcome these issues, which is termed as Fog Computing. The Fog Computing distributed paradigm leverages the computational capabilities of an additional infrastructure layer along with the resources of the edge and cloud layers.

1.1 RELATED COMPUTING PARADIGMS

Fog computing and edge computing are the extensions of Cloud computing, and both the paradigms appear to be similar because of their involvement in distribution of intelligence at the edges of the network. Few widely used paradigms were developed by combining the concepts of Cloud Computing and Edge Computing. Two such paradigms that are most closely inter-related to Cloud and Edge Computing are Mobile Edge Computing (MEC) and Mobile Cloud Computing (MCC). Mist computing is another more recent form of lightweight computing. This section presents few related paradigms and highlights how each paradigm lacked certain functionalities, which led to the instigation of the Fog paradigm.

1.1.1 Cloud Computing

Cloud is a dynamically scalable pool of resources that can be accessed over a network. The increasing demands of connected edge devices led to their integration with the Cloud. Several research efforts exploited the integration of the Cloud and IoT. The offerings of the Cloud can be used to store and process Internet of Things (IoT) data. Carrillo et al. (2015) proposed a Cloud-based smart city application ecosystem. Smart systems are complex, dynamic and heterogeneous. Thus, Cloud computing is used to transform and analyze the large sets of diverse data generated by these systems. Abawajy and Hassan (2017) applied Cloud computing in the health-care sector for developing a health status monitoring system such that the integration of the capabilities of the Cloud with IoT makes the system more flexible and scalable.

Even though many efforts have been directed towards the efficient integration of the Cloud and IoT, the rapid explosion in the number of services and edge devices ensued in the Cloud Computing being insufficient. The traditional Cloud, where data and processing is done through few centralized servers, proved to be inadequate in satisfying all the requirements of IoT applications. There are many IoT applications which require low latency (Qiu et al. 2018), mobility support and location dependent information such as traffic light system in smart transportation, smart grids (Qiu et al. 2017), smart healthcare and emergency response. Using the existing network configurations and low bandwidth, relying on the Cloud environment alone to carry out the analytics operations is impractical.

1.1.2 Edge Computing

Edge computing involves processing the data at the periphery of the network (Shi et al. 2016). The Edge paradigm provides quicker response time, reduced traffic and reduced network latencies. Several researchers attempted to integrate edge computing with IoT. Jutila (2016) proposed an adaptive computing method deployed at the edges of an IoT network for the optimization and control of traffic. Ahmed et al. (2017) discussed various scenarios where edge computing can be used for the efficient processing of data. Real-time image processing, gaming, smart grids, smart transportation and big data

real-time analytics are the few areas where edge computing can be applied for data processing and for converting data into useful information. Sharma and Wang (2017) proposed a novel framework which avails the advantages of both Cloud and edge computing. Several shortcomings such as limited computing capabilities, constrained processing resources, restrained analytical and networking capabilities of IoT devices make them unsuitable for executing complex, processor or memory intensive data processing applications.

1.1.3 Cloudlet

The concept of cloudlet was introduced for enhancing the resource availability for mobile users. Though cloudlets originated as dedicated servers, it evolved to small datacenters composed of several multicore computers. Cloudlets are also located at the end of the network and provides services to the end users near the edge of the network. Cloudlets are generally deployed in public spaces such as hospitals and shopping malls at a one-hop distance from the end devices. Cloudlets may also be deployed in an ad-hoc manner (Verbelen et al. 2012). It is placed in the middle layer of the three-level hierarchy where the first layer is constituted of the mobile devices and the third layer is the Cloud (Liu et al. 2016). Cloudlets provided the solution for latency issues involved in relying on distant wide area networks and also reduced the cellular energy consumption problems (Satyanarayanan et al. 2009). The usage of cloudlets also reduced bandwidth contention. Face recognition, augmented reality, crowd sourcing and video analytics are few application areas where cloudlets can be employed for faster processing of computational intensive tasks (Pang et al. 2015). The major limitations of the Cloudlet paradigm resulted from limited coverage and issues related to performance and security.

1.1.4 Mobile Edge Computing

Mobile Edge Computing (MEC) comprises of geo distant servers or virtual servers that are capable of providing IT services and are located close to the mobile users. MEC brings computation and storage resources to the edge of the mobile network to run the

applications with high resource and strict delay requirements at the end devices. Edge may refer to either the base stations themselves or data centres located close to the radio networks. MEC can either be connected or disconnected with the Cloud data center (Sabella et al. 2016). MEC accelerates service delivery and application responsiveness from the edge. MEC is based on a virtualized platform, with an approach complementary to Network Function Virtualization (NFV). This can reduce the round trip time and provides a layer of abstraction between the core network and applications running in the Cloud (Hu et al. 2015). MEC is used in many application areas for enhancing the performance. Augmented Reality (AR), Intelligent video acceleration, connected cars, health-care, smart grid, smart buildings, ocean monitoring, wireless sensor and actuator networks are few application areas where MEC can be adopted. Holo, Wal-laMe, Junaio, and Google goggles are few examples of AR-based applications. These applications demand quick response coupled with high computation abilities and large bandwidth requirements. MEC can be used to enhance the throughput of such applications by providing computation services at the edge of the network, rather than relying on the core network (Abbas et al. 2018). MEC can also be applied in big data analytics for quick extraction of meaningful information from the heterogeneous, unstructured raw data (Alsheikh et al. 2016; Laurila et al. 2012). Nevertheless, MEC faces challenges like secure deployment of MEC, lack of proper interface for user interaction, resource optimization and lack of proper billing mechanisms.

1.1.5 Mobile Cloud Computing

Mobile Cloud Computing (MCC) combines the concepts of Cloud computing, mobile computing and wireless communication to enhance the Quality of Experience (QoE) of mobile users and creates new business opportunities for both network operators and Cloud service providers. Mobile computing allows application developers to develop and host the applications in the Cloud, thus providing additional privileges to the developers. Developers are provided with the flexibility to develop applications that are not bound to a specific operating system and are not restricted by the capacities of the end devices. Computation offloading is carried out to transfer the complex processing

activities from the resource constrained devices to the Cloud (Abolfazli et al. 2014). Thus, this solution resolves battery draining issues of the resource-limited end devices. Capabilities of the Cloud increase the overall flexibility of the application (Xia et al. 2014). Considering the implementation aspects, MCC raises few issues. Mobile clouds make use of radio waves which have very limited bandwidth when compared to wired networks. MCC demands high bandwidth for uploading all the computationally intensive tasks into the cloud. The usage of wireless network for the transfer of data also imposes additional security issues. Connectivity is another challenge faced by MCC. Ensuring uninterrupted connectivity with external networks consumes more energy.

1.1.6 Mist Computing

Mist computing is a simple lightweight computing paradigm consisting of devices located at the edge of the network. It consists of specialised and dedicated nodes such as microcomputers and micro-controllers with limited computational resources. These nodes are called mist nodes.

Mist pushes the capabilities further down to the sensors and actuators, thus increasing the system autonomy (Preden et al. 2015). Edge computing offers fixed application configurations, whereas mist computing supports dynamic and adjustable configurations. Mist computing layer may be a part of Fog computing but not essentially a mandatory layer of Fog.

Hence, the existing paradigms are not equipped with the desirable characteristics to support the deployment and execution of IoT applications, thereby substantiating the need for a paradigm that better fulfills the requirements of IoT applications.

1.2 BACKGROUND

In this section, the rudiments of Fog Computing, such as characteristics of Fog environments, architecture of the Fog computing environment and the key entities of the Fog computing layer, are presented.

1.2.1 Purpose and scope of Fog Computing

IoT devices continuously generate streams of data, and often analysis must be very rapid. Performing rapid analysis of the large volume of heterogeneous data requires a computing model with the following properties:

- **Minimum latency:** Real-time applications that are time-critical are latency sensitive. Requests to such applications have to be processed and analysed with minimum time delay. Existing Cloud and Edge platforms incur significant delays in transforming data into information (Pan and McElhannon 2018).
- **Mobility aware processing:** Applications that are location dependent requires location information to be taken into consideration during decision-making. Thus, location awareness must be incorporated into the system (Cisco 2015). End user movement information must also be considered.
- **Preserve bandwidth:** IoT applications generally generate huge amounts of data at regular time intervals. Uploading all the data through the interconnecting networks to the distant Cloud data centers for processing, incurs high overheads. Thus, it is preferable to perform some processing activities closer to the data origin (Shang et al. 2016).
- **Secure Data:** The storage and transfer of the huge amount of collected data should be performed in a secure manner (Touati and Challal 2016), in order to preserve the privacy of the users.
- **Reliable Operation:** The IoT devices are generally integrated with mission-critical systems, which may affect the society directly or indirectly. Thus, the processing of data from IoT devices must be done in a reliable manner.
- **Flexibility in choice of optimal processing node location:** The location of the processing node is determined considering various requirements such as the nature of the operation to be performed and the allowable delay in response (Bonomi et al. 2014).

Traditional Cloud and Edge computing technologies do not meet all of these requirements. This led to the emergence of a new computing model, the Fog computing model, that can tackle these issues and efficiently handle all the scenarios.

1.2.2 Fog Computing Paradigm

In the recent years, multiple overlapping definitions have been formalized for Fog computing. An overview of the common definitions and our view on the Fog paradigm is described in this section.

Cisco views Fog computing as “*an extension of the Cloud towards the edge of the network*”. Fog offers storage, network and computation facilities to the end users. The geo-distributed nature of the Fog environment enables the fulfilment of the dynamic demands of the mobile client applications without incurring additional delays (Bonomi et al. 2014). Vaquero and Rodero-Merino (2014) defined Fog computing as “*a paradigm where a huge number of heterogeneous virtual or physical devices communicate among themselves and collaboratively work together to process compute and storage tasks without the intervention of a third party*”.

Fog computing is a horizontal layer between the end devices and the Cloud computing environment, which consists of virtual or physical resources. This layer supports delay sensitive applications by providing computing, storage and network capabilities in an ubiquitous, scalable, and distributed manner. The data is processed by geographically distributed devices at the edge of the network and these devices are called Fog nodes.

1.2.2.1 Fog Nodes

Fog nodes or Fog servers are the dispensers of Fog Computing, that are placed at different levels between the Cloud and the edge of the network to enable efficient data storage, processing and analysis of data while significantly reducing the latency by limiting the amount of data transported to the Cloud. The OpenFog consortium defines a Fog node as: “*the physical and logical network element that implements Fog computing services. It is analogous to a server in Cloud computing*” (Cisco 2016).

According to other definitions, the facilities or infrastructures that can provide resources for services at the edge of the network are called Fog nodes (Yi et al. 2015a). Fog paradigm need not follow a single node implementation, rather, it can include the deployment of a wide variety of devices which possess computing, storage and networking capabilities. Routers, switches, gateways and Cisco unified computing system servers are few possible candidates for Fog nodes (Cisco 2014). Fog nodes are placed in different locations to provide horizontal expansion of functionalities over geographically distributed locations.

Based on the functionalities, Fog nodes can be categorized into three:

- Fog nodes which directly upload the raw data without processing, to the Cloud.
- Fog nodes that perform pre-processing activities on the raw data, such as filtering.
- Fog nodes that possess capabilities to run distributed applications.

Fog nodes can also be classified based on their deployments according to the possessing agency, size and access modes:

- Private Fog node: Fog nodes which are exclusively provisioned by a single organization comprising of one or many users of the organization.
- Community Fog node: Community Fog nodes are dedicated to a specific community of users who share similar concerns.
- Public Fog node: Public Fog nodes are open to all and it can be accessed by anyone on a pay-per-use basis.

The location of the Fog nodes, the number of Fog nodes to be placed, the hierarchy level, is all decided on a case-by-case basis. The research in this field is still at its infancy stage, which is evident from the article by Marín-Tordera et al. (2017) where an attempt is being made to reach a consensus on what can be a Fog node and how and where it can be deployed.

1.2.3 Fog Computing Characteristics

Even though Fog computing can be considered as an extension of Cloud computing capabilities, it possesses various unique characteristics. Some of the characteristics are listed in this subsection.

- **Proximity to end user and low values for latency**

Fog nodes are located close to the sources that generate data and thus it acquires the data from the end devices, processes the requests and stores the results at the edge of the network. Therefore, Fog nodes aid in reducing the amount of data that has to be transferred through the network, thereby ensuring that the demands of delay sensitive critical systems are met (Hu et al. 2017).

- **Dense geographical distribution**

Fog computing environments consist of a large number of geographically distributed heterogeneous nodes spanning multiple domains. The improved capabilities of the smart devices provide opportunities for processing the data in closer proximity to the end users, rather than having to send all the collected data to the distant Cloud datacenters. This characteristic of Fog supports location-based services and enables requests to be processed with reduced response times.

- **Support for user mobility**

The Fog computing environment provides services through geographically distributed Fog nodes. Fog nodes can be either stationary or non-stationary. They may be deployed stationarily in coffee shops, bus stands or may be placed in moving objects. Fog nodes can communicate directly with the mobile devices and process the substantial amount of data generated by the devices to enable mobile data analytics (Bittencourt et al. 2017). Streaming of the data from moving vehicles for creating a smart vehicular network, processing crowdsourcing based applications and real-time gaming applications are few applications where the mobility feature of Fog computing proves advantageous.

- **Heterogeneity and interoperability**

Fog nodes can be placed across different layers between the Cloud and end devices and can be deployed in a wide variety of environments. Fog node can be any device which can provide services, ranging from base stations, access points, edge gateways or routers to dedicated servers and virtual devices. Therefore, Fog nodes exhibit heterogeneity (Aazam and Huh 2016). The network infrastructure used in the Fog environments is also diverse in its properties. It may comprise of high-speed wired connections or low-speed wireless communication links. Interoperability allows the different heterogeneous devices and networks to communicate and collaborate with each other for providing the services. The Fog computing elements must be able to interoperate and cooperate with the different providers to ensure seamless service delivery. Complex, distributed Fog environments adopt policy-based schemes for the secure collaboration among the resources for interoperability support (Dsouza et al. 2014).

- **Low energy consumption**

Pushing the data processing capabilities to the edge of the network rather than relying on the distant Cloud datacenters alone, helps in reducing the energy consumed by the network for the transmission of data, thereby resulting in significant reductions in the overall energy consumption. Sarkar et al. (2015) carried out experiments to compare the energy consumption in Fog and Cloud environments, in scenarios where large number of connected devices demand real-time services. Researchers observed that the Fog environment outperforms the Cloud in terms of both energy-savings and Quality of Service (QoS).

These characteristics make the Fog paradigm distinct from the Cloud and Edge computing paradigms. Table 1.1 presents a comparison of the Cloud, Edge and Fog paradigms.

Key Feature	Edge Computing	Fog Computing	Cloud Computing
Application Hosting	Limited application hosting capability and some end devices do not have ability to host.	Application hosting capabilities inherent in Fog nodes.	Cloud can accommodate wide variety of applications.
Resource pooling	Edge is limited to the particular end device.	Few number of resources are pooled together near the edge of network.	Cloud consists of wide variety of pool of resources.
Processing capability	Edge devices may provide no or limited support for processing.	Provides processing capability for all real time applications.	Unlimited processing capability for all applications.
Scalability	Edge devices are not scalable.	Fog paradigm supports scalability .	Cloud provides unlimited scaling capabilities.
Latency	Processing takes place at the source itself, so latency is negligible.	Latency values are low since processing is performed near to the end devices.	Latency values are very high, since the processing is done in remote data centers.
Modular Hardware	Depends only on the device on the edge.	Fog is a collection of modular hardware.	Cloud is constructed according to the principles of modular hardware.
Permanent storage	Edge devices do not provide any permanent storage facility.	Limited capability for permanent storage; ample support for transient storage.	Provides unlimited permanent storage.
Security Scope	Single device	End to End	End to End
Virtualization	Virtualization is not supported.	Virtualization is supported; lightweight virtualization offers better support.	Different kinds of virtualization are supported.
Geographic Coverage	Scope is limited to a single end-device.	Fog devices have wider scope of geographic coverage, though limited within its coverage area.	Cloud servers have global coverage. User can access services through the network from anywhere .

Table 1.1: Comparison between different processing environments

1.2.4 Fog Computing Reference Architecture

Fog computing offers tremendous support for variegated application realms. Fog computing performs better in terms of bandwidth efficiency, response time, security and reliability when compared to existing Cloud-based systems. The widespread interest in Fog computing gives rise to the need for an interoperable Fog computing architecture. The OpenFog consortium heeded to the need and devised a reference architecture (Cisco 2016). The OpenFog reference architecture consists of multiple stakeholder views, as illustrated in Figure 1.1.

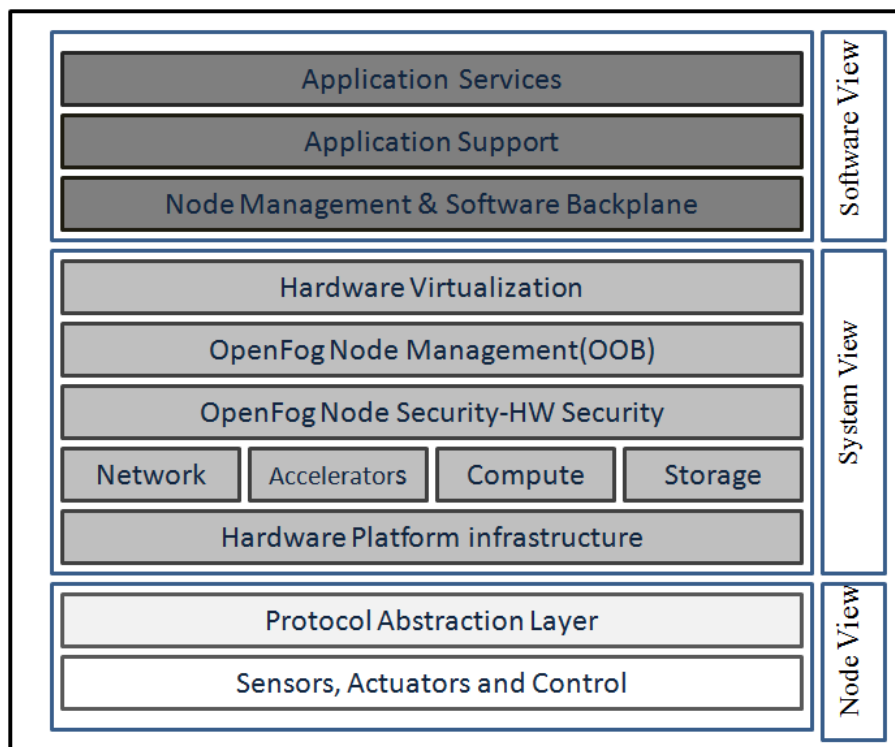


Figure 1.1: Fog Reference Architecture (Cisco 2016).

The lowest level view in the architectural description is termed as the ‘node level’ view. It consists of two component levels. The first layer consists of sensors, actuators and controllers that constitute devices connected to the Fog nodes. The end devices may or may not be equipped with processing capabilities. The next layer is the protocol abstraction layer. The protocol abstraction layer realizes the connections between the end devices and Fog nodes. The abstraction layer supports interoperability by providing an interface layer for effective communication between the multi-vendor end devices

and Fog nodes.

The System architecture view encompasses one or more node views. This view provides information for the system architects and platform manufacturers. The layer provides specifications about the requirements for the hardware platform infrastructure of Fog environments. The System architecture view also contains a hardware virtualization layer. Hardware virtualization techniques allow multiple entities to be hosted on the same hardware component. Fog platforms support hardware virtualization allowing different users to share the infrastructure provided by the Fog nodes. Virtualization also ensures an additional level of security by providing an extra layer of isolation for each entity.

The Software architecture view consists of three layers. Node management and software backplane manages the Fog nodes and communication between the other nodes. This layer contains all the necessities to run a software. The application support layer encapsulates a broad category of software which are commonly used by various IoT applications.

1.2.5 Virtualization in Fog computing environments

‘Fog computing is a highly virtualized platform, typically, but not exclusively, located at the edge of the network, that provides compute, storage and networking services between the end devices and the conventional Cloud Data Centers (Bonomi et al. 2012)’. Similar to Cloud computing, virtualization is the backbone technology of Fog Computing. Virtualization is the technology which enables users to share a single entity among a group of users. Virtualization materializes the task by creating separate customized virtual environments of the system based on the requirements of each user. Based on the position of the virtualization layer, virtualization can be of different types like Full Virtualization, Paravirtualization and Operating System (OS) level virtualization.

There exists different kinds of virtualization techniques (Kniep 2014). One of the popular techniques involves a hypervisor or Virtual Machine Manager (VMM). In this technique, virtualization services are provided through Virtual Machines (VM). How-

ever, this creates an additional overhead due to the running of a fully installed OS. As a solution to this overhead, the virtualization technique based on the OS level offers a model called container virtualization, which gives near native performance (Kozhirbayev and Sinnott 2017).

1.2.5.1 Container Virtualization

Container virtualization or containerization was initiated by the UNIX operating system in 1979 with their system called *chroot*. Then, in 2000, the Free BSD jail container technology evolved, which was similar to the *chroot*, but incorporated features for isolating file systems, users and networking. Linux VServer was another jail mechanism that was an initial implementation of virtual private servers. OpenVZ containers which used a patched variant of the Linux kernel emerged in 2005. Each of the OpenVZ container possessed isolated file systems. In 2007, the first complete implementation of Linux container manager, the LXC was released (Felter et al. 2015). Later, containers were considered as processes with extra isolation thereby helping in reducing the overheads associated with VMs. Container virtualization allows systems to deploy and run applications without creating separate VMs for each user. Multiple isolated containers are run on a single host by sharing a single kernel. The Linux features such as namespaces, *chroot* and *cgroups* provides secure execution of containers in the same kernel. Since containers do not use separate OS instances, it requires less CPU, memory and storage, when compared to traditional virtualization. Thus, the same host can incorporate more number of virtualized containers. The time required to create and deploy containers is very less compared to the virtual machine manager based systems. Containers possess packaged, ready to deploy applications or parts of applications, and if necessary, middleware and business logic to run those applications (Ciuffoletti 2015). Figure 1.2 and Figure 1.3 shows the two different virtualization architectures (Babu et al. 2014). Being a lightweight virtualization technology, the container virtualization technique is adopted in Fog computing environments to enable the deployment of multiple IoT applications.

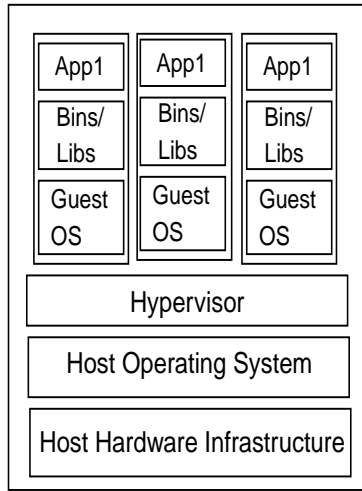


Figure 1.2: Hypervisor-based Virtualization

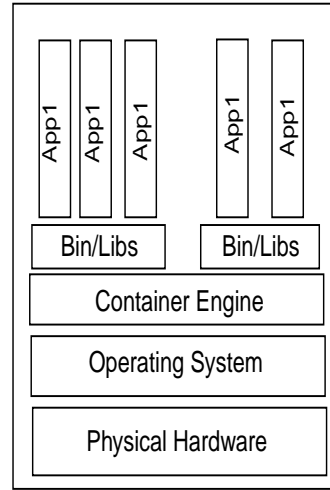


Figure 1.3: Container Virtualization

1.2.6 Fog Computing Environment

Fog computing is realised by annexing a resource-rich layer between the end devices and Cloud computing layer. Figure 1.4 gives an overview of the Fog environment. It consists of three layers. The bottom layer consists of various types of edge devices located closest to the users. It may consist of various sensors, smart vehicles, mobile phones, smart cameras and home appliances. These devices are distributed across geographically distant areas.

Their primary purpose is to acquire data from the environment and act based on the instructions from the higher layers (Garcia Lopez et al. 2015). Some of the edge devices possess the ability to process data and store data.

The Fog nodes are generally placed at a one-hop distance from the edge devices. The streaming data collected by the edge devices are transferred to the Fog nodes, rather than transporting the bulk volumes of data to the Cloud. This can lead to significant reduction in delays and latencies. Time-critical real-time application requirements can be handled in the Fog layer itself. The Fog nodes receiving this data, process the data and take decisions which are then communicated to the edge devices.

Data that requires to be persistently stored for use in future analytical operations, is

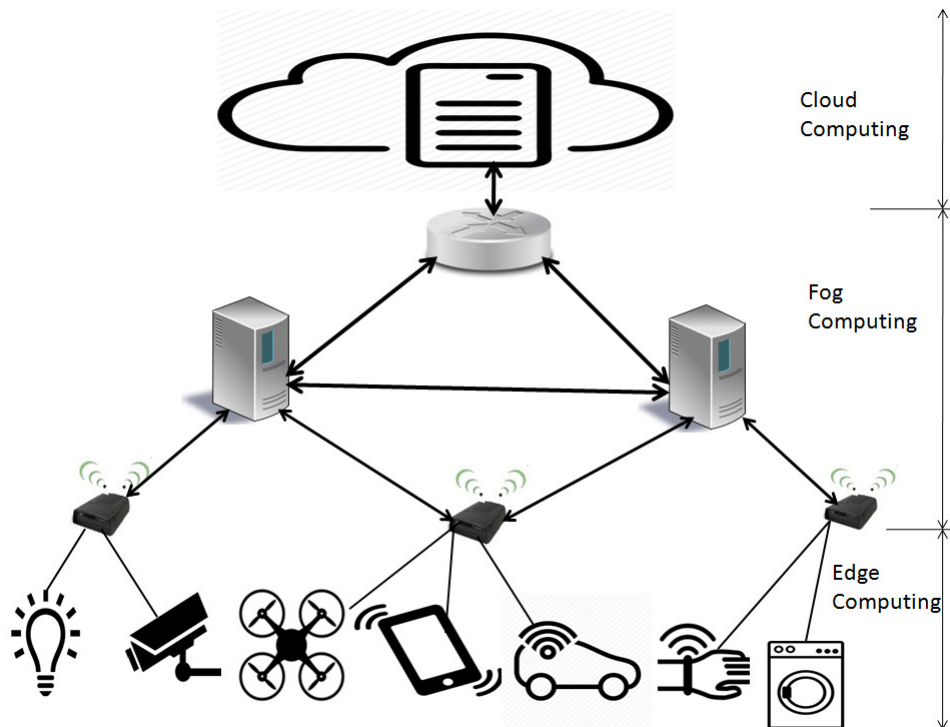


Figure 1.4: Fog Computing Environment

transported from the Fog node to the Cloud, which provides persistent storage mechanisms. The remaining operational and business-related information processing that is not performed at the Fog level, is carried out in the Cloud. The Cloud layer consists of high capacity servers for data processing and storage.

In order to harness the distributed nature of the Fog paradigm, IoT applications are modelled as a collection of application modules or services. In this work, the terms application component, application module and application service have been used interchangeably and refer to a component that focusses on a single business capability. These components interact among themselves (usually through a stateless server) in the course of their request processing (Skarlat et al. 2017). Each module may be deployed and executed independent of each other. The processing of data generated by the edge devices is handled by the corresponding modules of the application deployed on the Fog nodes in the Fog Layer. To ensure negligible performance overhead, the Fog Computing environments may opt for lightweight virtualization, where the application modules are hosted in containers, rather than on virtual machines. The containers encapsulate

the user data and application modules (Bellavista and Zanni 2017).

Dynamic, co-ordinated management systems are necessary to manage the whole ecosystem of Fog and Cloud. Decisions regarding which requests are to be handled at the Fog level and which requests are to be forwarded to the Cloud, are also taken by the management system.

1.3 MOTIVATION

The limited capacities and heterogeneity of Fog devices demand efficient orchestration mechanisms for the resource management in Fog computing environments. In addition to the orchestrational challenges that are present in the Cloud and Edge computing paradigms, Fog environment introduces few additional challenges on the orchestrator, due to its unique features such as distributed large-scale structure, dynamic network, mobility support, quick response and other specific QoS requirements. A Fog Orchestrator should be capable of developing a centralised pool of resources from the scattered dynamic heterogeneous nodes and the control layer in the orchestrator should be able to scale, in order to incorporate the increasing number of Fog devices. The orchestrator maps user requests into specific hardware configurations. The Orchestrator should manage the overall workflow and make sure that the user gets all the services with the desired QoS values.

Orchestration tasks involve different activities to efficiently manage the resources of the Fog ecosystem. Orchestration in Fog environments thus include resource management tasks such as allocation, load balancing and workload execution management. This research focusses on two major orchestrational challenges posed in the deployment and runtime of Fog applications. In particular, this work delves into application module placement/allocation and reallocation of the application modules through migration in the Fog computing environment.

The placement activity involves the selection of the most suitable nodes for the deployment of different services. Existing Fog service placement schemes consider various QoS metrics such as response time (Mahmud et al. 2019a), energy, efficient utilisation of Fog resources and cost (Mahmud et al. 2019b). The placement solu-

tions are either centralised or decentralised (Guerrero et al. 2019). However, there is a dearth of research in Fog service placement schemes which considers both reliability and monetary cost of execution. Reliable service delivery in Fog environments is essential, but, nevertheless, very exigent. The failure of individual services renders the entire application useless. The efforts for increasing the reliability of the system leads to an increase the overall cost. The deployment should also ensure the delivery of the services within a tolerable delay. *Therefore, this research presents a placement policy that takes into consideration the parameters of reliability, cost and deadline.*

User devices connected to Fog nodes are often non-stationary. The location-awareness attribute of Fog computing deems it necessary to provide uninterrupted services to the users, irrespective of their locations. Migration of user application modules among the Fog nodes is an efficient solution to tackle this issue. However, approaches that keep migrating modules along the trail of the mobile user may result in several unwanted migrations. Each migration action incurs an overhead on the system. This overhead is due to resource constraints and network bandwidth constraints. Thus, migration is opted only in situations where it is not possible to prolong the execution further. In the event of no viable options to transfer the processing, the application may be offloaded to the Cloud to ensure uninterrupted service delivery. Even though many of the Fog based applications are based on container virtualization, the existing research on migration in Fog supports only VM based migrations and does not support container migration (Lopes et al. 2017). They have not considered any user attributes (such as user location, user mobility pattern, etc.) while taking migration decisions. This raises the need for a migration framework which perceives the environment and relays information about the environmental context to determine the migration decision. Hence, in this research, *an autonomic framework is developed to trigger migrations in the Fog environment, based on the user mobility pattern.*

1.4 ORGANIZATION OF THE THESIS

The remainder of the thesis is organized as illustrated in Figure 1.5. Chapter 2 provides a taxonomy that covers the different aspects of existing research in the Fog computing

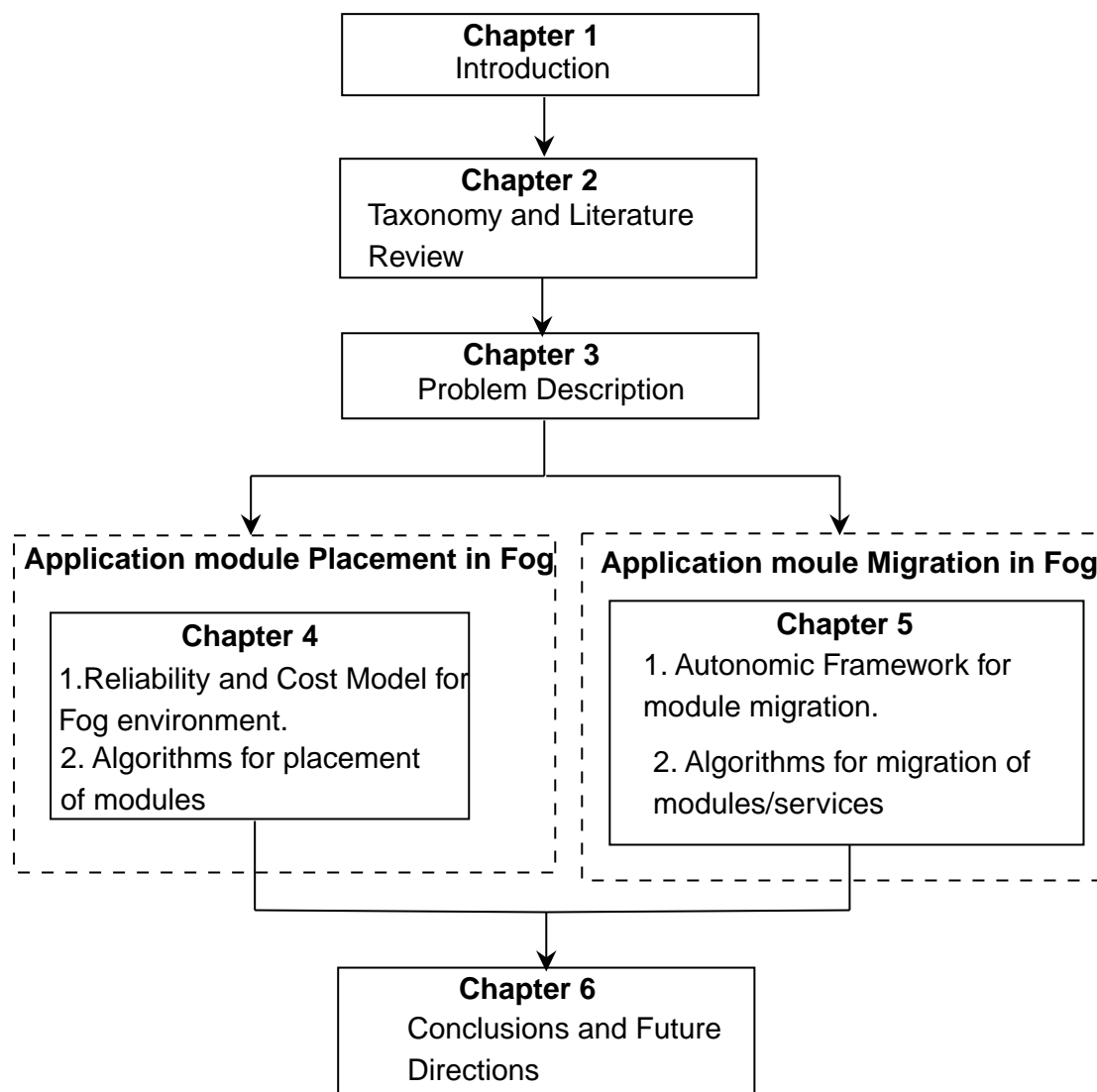


Figure 1.5: Organization of the thesis

environment along with a compendious review of the research approaches in the Fog computing domain, with special emphasis on the resource orchestration techniques. The chapter also provides a list of research challenges in this domain which requires attention from researchers. Chapter 3 describes the research problem, the focus of this thesis, the methodologies adopted to solve the problem and the major contributions of the thesis. Chapter 4 describes how application service modules can be placed in the Fog computing environment, and describes our proposed application service placement scheme for the selection of the most suitable nodes for the deployment of different

services. The experimental analysis of the proposed scheme is performed to analyse various performance metrics. Chapter 5 addresses the problem of migrating the containers corresponding to the user application modules, across the Fog nodes. The chapter also provides a mathematical model representing the optimization problem in the re-allocation phase. An experimental evaluation of our approach is also performed to analyze the performance. Finally, Chapter 6 presents a summary of the research work presented in this thesis and also includes some possible future research directions.

CHAPTER 2

LITERATURE REVIEW

The Fog paradigm, as any distributed paradigm, has its set of inherent challenges. The Fog environment necessitates the development of management platforms that effectuates the orchestration of Fog entities. Owing to the plenitude of research efforts directed towards these issues in a relatively young field, there is a need to organize the different research works. Accordingly, this chapter provides a compendious review of the research approaches in the Fog computing domain, with special emphasis on the resource orchestration techniques. Based on exhaustive analysis of the literature, a taxonomy that covers the different aspects of existing research in the Fog computing environment is proposed. Further, the research gaps for investigation in the domain of resource orchestration in Fog computing environments are also identified.

2.1 FOG COMPUTING ASPECTS

Bonomi et al. (2012) provided an abstract view of the Fog Computing paradigm in 2012, stressing on the need for the introduction of a new paradigm. However, the concepts of Fog Computing was expatiated later by Cisco (2015) and since then, Fog Computing has been the center of attention of the industry and academia alike. The research efforts are mainly directed towards tackling various issues that hinder the realization of the Fog Computing paradigm, which include orchestration or management issues, security-related concerns and communication techniques. Researchers have also attempted to

apply Fog Computing in different application realms.

A multi-level taxonomy that categorizes the research works according to the aspect considered is presented in this section. The proposed taxonomy for categorizing the existing research in the domain is provided in Figure 2.1.

According to the taxonomy, the research in the Fog domain can be broadly classified into two. The first category includes the application domains where Fog computing can be applied. The usage of Fog computing in these domains enhances the overall performance and reduces the latency of the system. Research efforts that adopt Fog computing in different application domains are categorized in Section 2.1.1. The second category of works addresses the different issues in application runtime and management of the Fog platform. Research works that fall under this category are discussed in Section 2.1.2.

2.1.1 Application Domains

Fog computing extends Cloud computing facilities to the edge of the network and provides support to IoT applications that demand quick responses and mobility support. In this section, we discuss various real time application domains where Fog computing is applied. Table 2.1 presents details of different research works that adopt Fog Computing into different application domains.

- **Internet of Things**

Many IoT applications demand real-time analytics which implies that, the devices must be able to react to events as they occur. The Fog computing paradigm can be used as a solution to meet the processing needs of IoT applications. Along with low latency values, the Fog also provides some advanced features like mobility support and location awareness (Dastjerdi and Buyya 2016). Fog leverages one or more devices at the edge to create a pool of resources by effective collaboration and communication among them. Tang et al. (2017a) proposed a Fog computing based big data analysis framework for smart cities. The proposed hierarchical framework provides responses in real time by applying analytics operations on

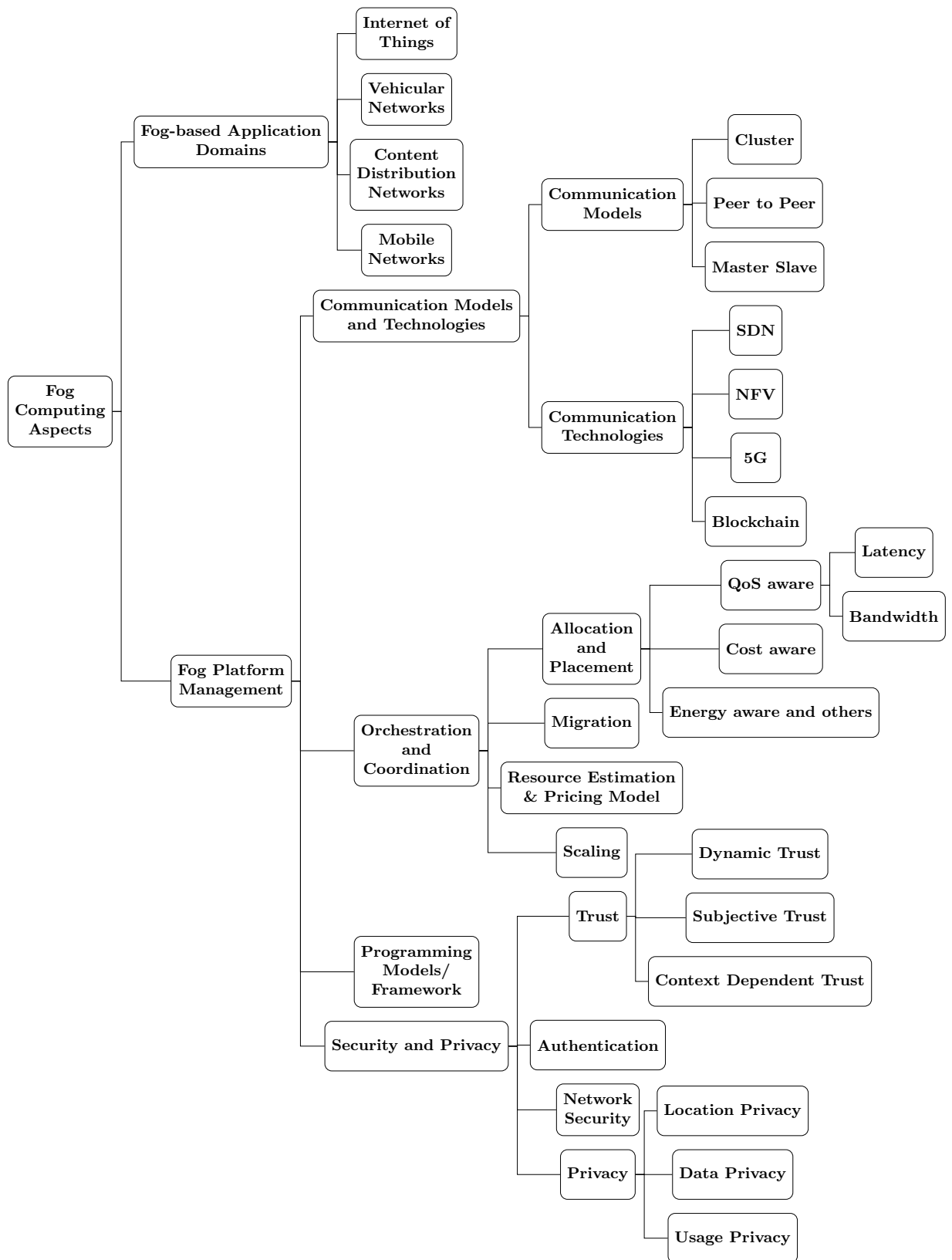


Figure 2.1: Classification of works considering different aspects in Fog Computing

Author	Scope	Major Focus
Tang et al. (2017a)	Smart City	Hierarchical Fog computing architecture for big data analysis.
Yan and Su (2016)	Smart grid	Fog enabled data storage and processing infrastructure for smart meter.
Mayer et al. (2017)	Social sensing service	Architecture for social sensing services with Fog computing.
Masip-Bruin et al. (2016)	Healthcare	Proposed an architecture to support group of patients, taking the advantage of mobility support characteristics of Fog computing.
Xu et al. (2016)	Data analytics	SDN based Fog computing for data analytics
Tang et al. (2017b)	Video processing	Cooperation model for video processing among edge devices
Huang et al. (2017)	Vehicular network	Architecture for vehicular Fog computing.
Zhu et al. (2013)	Content distribution network	Fog based web optimization method.
Zhang et al. (2017a)	Radio access network	Fog radio access network architecture
Gu et al. Gu et al. (2017)	Cyber physical system	Cost efficient Fog supported medical cyber physical system.
Do et al. (2015)	Content delivery network	Resource allocation and approaches for reducing carbon foot print for streaming services.
Shih et al. (2017)	Mobile network	Fog radio access network architecture for low latency applications.
Dastjerdi and Buyya (2016)	Internet of Things	Fog computing architecture for Internet of Things.

Table 2.1: Different usecases employing Fog Computing

the data generated from millions of devices deployed across the city. Gia et al. (2015) adopted Fog computing in the healthcare sector for real time monitoring of human health using IoT devices. They applied Fog computing for enhancing the throughput of Electrocardiogram (ECG) feature extraction.

- **Vehicular Networks**

Intelligent Transportation System (ITS) is one of the emerging technologies, which applies advanced technologies to improve the safety, reliability and efficiency of the transportation system. Major challenges involved in the implementation are mobility of the vehicle node, different speeds of the nodes and real-time processing requirements of the applications (Karagiannis et al. 2011). Vehicular Cloud computing was one of the solutions to overcome the resource limitations of ITS vehicular networks. However, bandwidth scarcity and connectivity are the major challenges in realizing vehicular cloud (Mekki et al. 2017). Vehicular Fog computing is the paradigm which combines Fog computing with vehicular networks. Vehicular Fog pushes the computing resources near to the edge of the network thus reducing the problems due to bandwidth and connectivity (Xiao and Zhu 2017). Vehicles serve as the infrastructure for vehicular communication (Hou et al. 2016). Huang et al. (2017) proposed an architecture consisting of three layers, for vehicular Fog networks. The data generation layer performs data gathering and preprocessing of the data. Then, the Fog layer performs data fusion, preprocessing and takes area level decisions. The detailed analysis and exploration of the data is done in the Cloud layer.

- **Content Distribution Networks**

Content Distribution/ Delivery Networks (CDN) consist of geographically distributed servers which are capable of delivering the web pages as information. CDN uses edge caching approaches for speeding up the delivery of the web contents. Geographically distributed content delivery servers cache the contents of the web pages and when a request for the page arrives, CDN redirects the request to those servers which are geographically nearer than the distant webserver

(Pathan and Buyya 2007). Fan et al. (2016) takes the advantage of Fog computing in content delivery networks. Fog nodes are used as content servers at the edge of the network and all the client requests are first directed to the Fog nodes and subsequently the information is retrieved from the distant web servers and this information is cached for future usage (Zhu et al. 2013).

- **Mobile Networks / Radio Access Networks**

Radio Access Network (RAN) is the backbone of modern telecommunication systems. It establishes radio connections between the different devices and other network regions. The limited capacities of mobile networks and user demands can be better satisfied by employing Cloud in mobile networks. This led to the emergence of a new paradigm called Cloud Radio Access Network (CRAN) (Wu et al. 2015). The major limitation of CRAN was the requirement of huge bandwidth for connecting a mobile device with the Cloud resource pool. CRAN was also not capable of handling unpredictable mobility of users and increasing number of base stations. Fog computing based Radio Access Networks (FRAN) came to existence as a solution for all these. FRAN allows distributed processing and provides computing capabilities at the edge devices rather than confining the processing capabilities to the centralised servers (Peng et al. 2016). Shih et al. (2017) applied FRAN for solving ultra-low latency applications. Tandon and Simeone (2016) developed a framework to compute trade-offs between the fronthaul capacity and edge system resources.

2.1.2 Platform Management

Proper mechanisms for managing the run-time platform is a pre-requisite for the successful execution of applications in the Fog computing environment. Platform management intends to provide a secure ecosystem capable of mapping the application specific requests to a pool of available hardware configurations, while ensuring the defined QoS levels. Consequently, platform management in a Fog computing environment includes tasks such as handling communication among Fog entities, orchestration and coordination, developing programming models and ensuring security and privacy. Several

research approaches have been proposed to develop efficient solutions for each of these tasks. The research approaches addressing each task are discussed in Sections 2.1.2.1, 2.1.2.2, 2.1.2.3 and 2.1.2.4, respectively.

2.1.2.1 Communication Models and Technologies

The Fog layer acts as an intermediary layer between the end devices and the Cloud. Fog paradigm is founded on the inter-connections between the end devices, the Cloud datacenters and the Fog nodes. Connectivity may be provided using wired or wireless connection links. Various communication models and technologies are used to enable communication between the Fog-edge and Fog -Cloud interfaces and for the effective collaboration among the Fog nodes.

Communication Models Communication models specify the different entities involved in communication and specifies the interaction between these entities. There exists three types of communication and collaboration among the Fog nodes.

- **Cluster Model** Fog nodes can communicate among themselves by forming a cluster. Cluster-based collaborations can harness the capabilities of the several Fog nodes. Clusters can be formed with Fog nodes of same types or Fog nodes which are located in adjacent geographic regions (Oueis et al. 2015). Since Fog computing environments are dynamic in nature, dynamic clusters are more efficient than static clusters.
- **Peer-to-Peer Model** The peer-to-peer network model is a decentralized model consisting of nodes with equal capabilities. In the peer-to-peer model, each node serves as both the client and the server. The Fog computing paradigm embraces this model to enable collaboration among the Fog nodes. The collaboration among the nodes may follow either a hierarchical order or a flat order.
- **Master-slave Model**

The master-slave network model can be used in Fog environments. In the master-slave model, one node or process, designated as the master, controls the other

nodes, called slaves. In the Fog environment, this model is realised by one master Fog node controlling all other slave Fog nodes (Lee et al. 2016).

Communication Technologies Communication technologies refer to the technical aspects of the connections. Apart from the common communication technologies such as 3G, 4G, WLAN, Bluetooth and Zigbee, Fog computing also employs the recent communication technologies. These technologies are discussed in this section.

- **Software Defined Networks (SDN)**

The distributed nature of the Fog environment manifests SDN as an apt technology for the efficient traffic control and connectivity management of end devices (Truong et al. 2015). The geo-distributed nature of the Fog environment requires more than a single controller and OpenFlow switches are used for data exchange between these controllers. In IoT applications that include time critical activities which must be processed without any delays, an OpenFlow controller can assist in distinguish emergency traffic from the delay-tolerable operations. Thus, it empowers the Fog system to handle critical activities with high priority (Tomovic et al. 2017).

SDN can also be used as a solution for Fog orchestration issues. Dynamic management of services for the end devices can be done through these controllers. The SDN controller can log the location information of the end devices based on the mobility of the user. It can also be used for efficient hand-off mechanisms. For example, in vehicular networks, Fog based on SDN can provide better services with high throughput and minimal delay (Truong et al. 2015).

- **Network Function Virtualization (NFV)**

Network Function Virtualization applies virtualization based methods for designing, deploying and managing network services. The network functions such as Network Address Translation (NAT), intrusion detection, domain name services and firewall mechanisms are provided by virtualized infrastructure (Mijumbi et al. 2016).

Integration of NFV with Fog computing helps in better delivery of Fog services (Yi et al. 2015a). The gateways, switches, firewalls and other networking components can be virtualized and placed in the Fog nodes. Majority of the Fog nodes work in energy constrained environments. Since NFV reduces the use of dedicated hardware components, it leads to reduced power consumption, which is a major requirement in Fog environments (van Lingen et al. 2017). NFV also supports orchestration functionalities of geo-distributed heterogeneous Fog environments (Hu et al. 2017).

- **Fifth Generation (5G)**

The fifth generation wireless communication systems offer many advantages over the fourth generation systems. Their attractive features include low battery consumption, multiple data transfer paths, better coverage and more secure means of communication (Akpakwu et al. 2018).

5G can be used for satisfying several requirements of the Fog computing environments. 5G systems offer more bandwidth resulting in high speed communication, as compared to 4G (Amendola et al. 2016). It enhances the overall performance throughput in the Fog environment and can provide low latency services (Singh et al. 2016). 5G can also reduce resource limitation issues and provides high-quality wireless communication in Fog computing environments (Vilalta et al. 2016).

- **Blockchain**

Blockchain technology can be used for the control of the distributed Fog computing environment. The autonomous nature of the operating system and decentralised structure of the IoT, demands direct communication and interaction between the entities. Blockchain can be used for secure interactions between the entities and thus avoids the need for a central controller that effectively coordinates these entities. Stanciu (2017) developed a blockchain based control system for edge computing using the IEC 61499 standard. Hyperledger fabric was used as the solution and smart contracts were used for the implementation of func-

tional blocks. Sharma et al. (2018) developed a secure Fog architecture which uses SDN and blockchain technologies. They developed a new protocol called Proof-of-Service to mine blocks. Though blockchain technologies may be effectively used for constructing reliable Fog networks, the possibilities have not been fully explored in the existing literature.

Communication standards are adopted to provide interoperability, uniform design and testing strategies and thus improve the overall quality of experience. In 2018, the OpenFog consortium developed the IEEE1934 standard to handle the huge data generated by IoT and 5G. This provided a standard framework for developing Fog based applications and business models and also ensured that the services and applications are processed nearer to end devices. IEEE802.11p is another standard that is widely used in Fog-based intelligent transportation systems.

Reliability is one of the key requirements of a real-time system. Wireless communication between the devices may impair the reliability of Fog networks leading to issues such as radiated electro magnetic interference and deterioration in end to end packet reliability. Therefore, more efforts may be directed to reduce the overhead resulting from the communication techniques.

2.1.2.2 Orchestration and Coordination

Efficient execution in the Fog environment demands orchestration mechanisms capable of coping with the dynamics of the workload. An orchestrator in the Fog environment should be able to handle dynamic requirements of the applications. Wen et al. (2017) explored the challenges involved in the design of an orchestrator in IoT based Fog environments. The uncertainties in the working environment of IoT applications may create some internal transformations and corresponding dynamic changes in workflow components. Hence, a dynamic orchestrator which has the capability to cater services is required.

Orchestration of the Fog components is a quintessential feature to ensure service delivery. This section discusses the major issues in orchestration of Fog environments

and highlights relevant research works directed towards each issue.

Allocation and Placement Fog provides decentralized processing power and distributed intelligence. The requirements of each application or components of an application will be different. Some demand low latency response, other application focus on low cost execution and others may possess high storage requirements. The existing allocation and placement policies consider any one of these metrics or a combination of few of the metrics. Based on the metric considered, the existing approaches can be further classified as:

- **QoS aware approaches:**

Many allocation methods are based on the QoS parameters. QoS is a measure to quantify the overall performance level of the system or service. Ni et al. (2017) proposed a Priced Timed Petri Net (PTPN) based dynamic resource allocation and scheduling strategy for Fog computing environments. He et al. (2018) developed QoS aware admission control, offloading and resource allocation schemes to support data analytics services aimed at maximizing the analytics service utilities. The admission control function takes decisions based on the work models developed from offline execution of benchmarks. Among the QoS metrics, two metrics that have been commonly addressed by researchers in Fog computing, are the bandwidth and latency requirements.

- Latency and Response time:

Critical real-time applications demand quick responses to incoming transactions while satisfying the stringent bandwidth requirements. The time taken to process a request largely depends on the location where it is processed. Processing in Fog environments, can take place either at the end devices, smart routers, gateways, dedicated Fog nodes or in the Cloud. Latency aware allocation policies consider latency as a metric to determine where the application is placed. Gupta et al. (2017) developed a simulator to evaluate resource management and scheduling policies applicable to

the Fog environment with respect to their impacts on latency, energy consumption and operational cost. Application modules are placed nearer to the end devices as far as possible. However, in this approach, modules that are dependent are placed at the same level without considering the timeliness and complexity of modules separately. Bitam et al. (2017) proposed a bio-inspired optimization approach called bees life algorithm for job scheduling in Fog environments. Their approach distributes tasks among the Fog nodes and finds an optimal trade-off between Central Processing Unit (CPU) execution time and the memory allocated.

– **Bandwidth:**

One of the major reasons behind the evolution of Fog computing paradigm was the increased bandwidth requirement of end devices. QoS-based allocation approaches are mainly focussed on reducing the bandwidth usage of overall application. Yousefpour et al. (2018) proposed a dynamic application module deployment scheme for the Fog environment, which satisfies QoS requirements. Application module placement guarantees that the overall bandwidth consumption and latency will be minimal. Skarlat et al. (2016) considered bandwidth as one of the major metrics to be reduced while allocating an application or a set of applications for execution in the Fog computing environment. Brogi and Forti (2017) proposed a model to support QoS aware deployment of multicomponent IoT applications on the Fog infrastructure. They considered latency and bandwidth as the only QoS parameters. The component deployment problem was mathematically modelled and a heuristic solution with two stages which included preprocessing and backtracking was proposed.

• **Cost of execution:**

Cost aware allocation approaches emphasis on the objective of reducing the overall execution cost of the application without compromising on the performance requirements. Cost aware approaches takes two forms based on the requirements of the application: ‘minimize total cost such that the overall performance meets

the target’ or ‘maximize the overall performance such that resource consumption does not exceed a limit’. Jingtao et al. (2015) proposed an efficient method to share resources in a Fog cluster for providing local access to services for mobile users, in a cost-efficient manner. The Fog cluster is composed of many function-specific servers. Their connected nature creates a topological abstraction called the connecting layer. The authors developed a method for transferring data between the connected servers in a cost-efficient manner. Sarkar and Misra (2016) aimed at evaluating service latencies and energy consumption of the Fog paradigm applied to the IoT as compared to traditional Cloud scenarios. The proposed model deals mainly with the behaviour of software already deployed over the Fog infrastructure. There are frameworks which facilitate the distribution of the execution of tasks among different edge devices in order to reduce the overall cost of the execution. Dinh et al. (2017) proposed semi-definite relaxation based approaches for distributing tasks from a single mobile device onto different access points. In addition to the cost of execution, the VM capacity and service size can also be considered as metrics for the efficient allocation of services. Alsaffar et al. (2016) proposed a distributed allocation approach based on the collaboration between the cloud and Fog and they have considered all the aforementioned three parameters for obtaining an optimal allocation.

- **Other Allocation approaches:**

Apart from these approaches, few other approaches that consider other parameters have been proposed for the allocation process in Fog Computing. Few researchers applied game theory concepts for solving the allocation and resource management issues in Fog computing (Zhang et al. 2017b,c). Deng et al. (2016) proposed a framework for optimal workload allocation in the Fog computing environment. Minimal power consumption and service delay are the two major constraints considered while formulating the primal problem. Different allocation policies have been proposed to efficiently deploy the application modules

in the hierarchical environment to ensure placement of the operations closer to the user (Bittencourt et al. 2017). Kochar and Sarkar (2016) proposed an approach for two-level dynamic allocation of application modules with an aim of maximizing resource utilization at the edge of the network.

Along with the resource requirements and availability of resources among the heterogeneous devices, orchestration mechanisms must also consider any dependencies between the application modules. Power consumption plays an important role in the edge devices. Thus, the module deployment policies should try to reduce the energy consumption. Existing orchestration mechanisms for allocation may be enhanced to consider these characteristics.

Migration Dynamic changes in the workflow components are generated in response to internal transformations or abnormal system behaviour. This may result in the initial allocation being no longer optimal or even totally invalid. Therefore, dynamically orchestrating task execution and resource reallocation is essential. For this, following the initial allocation and placement, re-allocation decisions may be made. Migration of application modules from one Fog node to another is used as a solution to realize workflow redeployments. Migration process may be triggered either to meet the QoS expectations, for maintenance purposes or for balancing the load (Saurez et al. 2016). The process of migration imposes some costs on the Fog computing environment. One of the important tasks of migration is to decide when a migration action is required. This decision making problem can be formulated using Markov decision process and a mathematical framework can be used to solve the problem (Wang et al. 2015). The design of such decision-making frameworks in Fog must also consider the impact of the mobility feature in the Fog computing environment on the decision making process (Kattepur et al. 2016).

Since the Fog computing environment is resource-constrained, lightweight container virtualization mechanisms such as Docker, Rkt and LXC are more suitable than traditional hypervisor-based virtual machine approaches. Planning migration ahead of time will ensure that QoS violations do not occur and also reduces the overall network

utilization (Ottenwalder et al. 2013). Decisions to migrate must be made considering the profits to both the provider and the users (Bittencourt et al. 2015).

Resource Estimation and Pricing Model The devices in the Fog environment vary in their resource capabilities, architectures and platforms. Due to this heterogeneity, it is not possible to accurately determine the quantity of resources required to process each application request. The dynamic nature of the end devices in the Fog makes it difficult to verify whether the end devices are efficiently utilizing the allocated resources. Thus, resource estimation approaches, that can estimate the amount of resources, are required. Similar to the pay-as-you-go model in Cloud, Fog environments can make use of billing and pricing models which can accurately calculate the price for the Fog based services.

Probability-based estimation methods are efficient methods for calculating estimates of the amount of resources required and can thus help in reducing the wastage of resources and enhances the profit of the providers (Aazam et al. 2016). Aazam and Huh (2015b) proposed a pricing model based on the prediction of required resources for the completion of the application request and the actual amount of resources utilized. Aazam and Huh (2015a) incorporated the concept of ‘relinquish probability’ for the effective prediction of the resource usage and also developed price calculation measures for use in Fog computing.

There is much future scope for researchers to realize Fog computing as a business model. Service providers should be able to offer wide variety of pricing schemes to attract the customers and to fulfil the varying demands. A financial economic cost model developed considering different Fog parameters would help the provider to generate higher profit while ensuring that customers pay a fair price for the resources they use. Deep learning techniques may be applied for developing efficient resource estimation models.

Scaling The orchestrator determines whether the existing system can handle all the specific requirements of an application. It should be able to automatically detect scalability bottlenecks and solve such issues. Kapsalis et al. (2017) proposed a platform for the efficient management of Fog resources and allocation of tasks across the dif-

ferent layers of resources. They also introduced a workload balancer for selecting the most appropriate host for allocating each task. The proposed approach is able to handle large-scale applications with thousands of tasks and multiple hosts. Yangui et al. (2016) proposed an architecture for the Platform-as-a-Service model, to allocate application modules into an integrated Cloud-Fog environment. The orchestrator module manages the flow of execution between the different application modules and the deployment module decides where to deploy these application modules. Moreno-Vozmediano et al. (2017) proposed a Hybrid Fog-Cloud (HFC) interconnection framework for carrying out effective interactions between the geographically separated Cloud and Fog layers. HFC agents provide scalability support by allowing enhancements on the number of Fog/Cloud nodes corresponding to an increase in the resource demands.

Other Orchestration Approaches The dynamic nature of Fog environments raise the need for load balancing of tasks. To ensure balanced loads, reallocation of application components already allocated may be performed. Graph repartitioning can be used to devise solutions in such scenarios (Alsaffar et al. 2016). Dinh et al. (2017) developed a method for transferring data between connected servers in a cost-efficient manner. They applied the Steiner tree concepts from graph theory for developing such a caching method in Fog environment. A Graph G is composed, where the vertices represent the servers and the edges represent the connections between them. Steiner tree is used to find the minimum cost of the subgraph which can be generated from graph G . Dsouza et al. (2014) proposed a policy management module for supporting the resource orchestration layer in the Fog environment. The module takes decisions based on the information collected from all the components in the environment and also considers the service request.

2.1.2.3 Programming Models / Frameworks

Developing applications for the Fog is a challenging task as it involves management of dynamic heterogeneous resources from different levels of hierarchy. Programming frameworks should provide the required libraries, interfaces and running environments for the development of applications that can cope with these conditions. The program-

ming framework should also be able to deal with on-demand computing instances. Hong et al. (2013) proposed a programming model called MobileFog which can be used for developing large scale applications for the IoT. MobileFog provides a simplified programming abstraction and also accommodates dynamic scaling of applications. Santoro et al. (2017) proposed a framework called Foggy, for orchestrating workloads in the IoT environment. Foggy runs applications while ensuring that all other functional and non-functional requirements are satisfied. Saurez et al. (2016) proposed Foglet-a programming infrastructure for the geo-distributed Fog computing environment. Foglet manages the execution of application components on the Fog nodes by providing a high-level programming management model for distributed heterogeneous nodes spread across a wide area. Distributed data flow programming models can also be used for developing applications for the Fog environment. Giang et al. (2015) proposed a distributed dataflow programming model which allows easy application development and deployment considering both the available hardware resources and application requirements.

2.1.2.4 Security and Privacy Issues in Fog environments

The characteristics of Fog computing unfolded new challenges in security and privacy. The existing security and privacy methods in Cloud computing are not capable of handling the challenges caused by the mobility, heterogeneity and large-scale geo distributed nature of the Fog environment (Li et al. 2017). This subsection discusses the major concerns related to security and privacy in Fog computing environment.

Trust Fog is a distributed computing network in which each object may have to interact with other unfamiliar objects. Thus, trust must be ensured in a Fog network for the collaboration of nodes in the network. The heterogeneous nature of Fog nodes complicates the trust context in Fog computing (Mukherjee et al. 2017). Trust allows the Fog nodes and/or clients, to predict the expected behaviour of its peers and can thus aid the decision making process. The following characteristics are the requirements of trust in Fog computing environments:

- **Dynamic trust:** The mobile nature of elements in the Fog computing environment results in changes to the topology of the Fog. The behaviour of the objects may also vary with time. Thus, trust values must be calculated dynamically, at defined intervals (Mukherjee et al. 2017).
- **Subjective Trust:** Depending on the characteristics of each object in the network, the security requirements also vary. Thus, there is a need for different trust policies and subjective trust in Fog computing environments. The trust will also exhibit asymmetric properties.
- **Context-dependent Trust:** Context information plays an important role in the trust values associated with Fog computing environments .

Authentication Fog nodes provide varied type of services to the end users and users must be authenticated before accessing services (Stojmenovic and Wen 2014). Authentication prevents unauthorized access to Fog nodes and/or For services. Since majority of the Fog nodes are resource-constrained, common authentication mechanisms such as Public Key Infrastructure (PKI) and certificate based mechanisms are not suitable for the Fog (Alrawais et al. 2017). Recent advances in authentication such as biometric-based authentication schemes may be applicable in Fog computing.

Network Security Wireless security issues form a major network security concern in Fog environments (Yi et al. 2015b). Dictionary attacks, stolen verifier attacks, bit flipping attacks and impersonation attacks are the few popular security issues in wireless networks. There exists a lot of challenges for securing all communications in the Fog environment (Khan et al. 2017). Minimizing the message overhead in the resource-constrained environments pose additional challenges in securing the network.

Privacy The unauthorized access of information such as privacy-sensitive data, location information and usage patterns are few privacy-related issues in the Fog environment. The distributed nature of Fog makes privacy-preservation even more challenging (Khan et al. 2017). Few privacy issues are discussed in this subsection.

- **Location Privacy:** The location of end devices will be always linked to the owners to facilitate accessing services from the nearest Fog nodes . For example, consider a scenario of vehicular networks where Fog paradigm is employed for improving the throughput by temporarily storing the data and providing services locally. The Fog nodes exchange location information to reflect the mobility of the vehicle. Compromised Fog nodes can thus lead to the attacker obtaining information about the location of the node. The attacker can also observe the mobility pattern of the vehicle. Identity obfuscation can be used to deal with location privacy issues in Fog environment (Stojmenovic and Wen 2014).
- **Data Privacy:** Data transmitted by the sensor devices to the Fog nodes includes sensitive user-data. Compromised Fog nodes can lead to loss of data and security impairments of the entire ecosystem (Guan et al. 2018). Homomorphic encryptions can be used for preserving data privacy without much additional overheads.
- **Usage Privacy:** Since users are closely associated with the end devices and the Fog nodes, attackers can easily obtain user behaviour patterns by gaining access to the Fog devices. Hong et al. (2017) investigated the various cases of possible information leakages by analysing the smart meter readings in power grid. They found that by closely observing the smart meter, one can easily identify whether homes are inhabited or not. Privacy preserving streaming algorithms can be used for tackling such issues.

2.2 OUTCOME OF LITERATURE REVIEW

On conducting an exhaustive review of the existing literature, we have identified that even though there exists various methods for the management of the Fog environment, the research in this direction is considered to be premature and there exists significant scope for future research. The research gaps that were identified as the outcome of this literature survey and directed the research in this thesis, are as follows:

- Fog platform management, which includes the effective control and co-ordination of the different entities in the Fog ecosystem, is a key concern in the domain of

Fog computing. The scope of Fog platform management includes the orchestration and coordination of Fog resources. Though researchers have addressed few orchestrational issues, there exists much scope for further research and optimization.

- Multiple research efforts have been directed to solve the Fog Service Placement problem. However, there are several conflicting issues that must be considered while taking placement-related decisions, which have not been addressed in the existing research.
- The distributed and heterogeneous features of Fog environments deem it imperative to consider the reliability parameter in placement decisions to provide services without interruptions. The existing research has not considered reliability while taking service placement decisions (as inferred from Table 2.2). The efforts for increasing the reliability of the system leads to an increase in the overall monetary cost. Hence, there is a need for service placement policies that address the two conflicting criteria.
- Fog environments are highly dynamic. In order to ensure low latencies, there may arise the need to re-consider the location of the existing applications and migrate them. The existing systems take such migration decisions after the request for an application arrives. To better handle the dynamic nature, the decisions on when migration should occur must be based on prediction mechanisms.
- Users of the Fog environment tend to have different mobility profiles. Migration decisions in the Fog environment should thus consider the mobility of the user to ensure close proximity of the Fog nodes to the users. There is a lack of research on systems that can autonomically decide whether to perform the migration and complete the process if required.
- A multitude of research works that consider migration in Fog environments considered application modules running in virtual machines. These techniques cannot be directly adopted to container-based Fog environments. There is a dearth of

research that targets the migration in containerized datacenters(as inferred from Table 2.3).

Work	Problem Ad-dressed	Metrics Considered	Solution	Evaluation
Venticinque and Amato (2019)	Fog service Placement Problem	Resource utilisation, Delay	Benchmarking, evaluation and Testing	BET
Mahmud et al. (2019a)	Fog service Placement Problem	Delay	Latency aware policy	Simulation
Skarlat et al. (2017)	Fog service Placement Problem	Resource Utilisation, Cloud cost	Genetic Algo-rithm	Simulation
Brogi and Forti (2017)	Application module deployment	Delay, Bandwidth	Heuristic Solu-tion	Simulation

Table 2.2: Research works on Placement

Work	Problem Ad-dressed	Metrics Considered	Solution	Virtualization level
Bi et al. (2018)	Mobility Support	Performance of handover	Route optimiza-tion algorithm	Virtual Machine
Lopes et al. (2017)	Migration of Vms	Latency	Distance based	Virtual Machine
Bittencourt et al. (2015)	VM migration	Cost	Proposed Archi-tecture	Virtual Machine
Islam et al. (2016)	Migration in MCC	User mo-bility, increased/ decreased load at cloudlet	Genetic algo-rithm based	Virtual Machine

Table 2.3: Research works on Migration

2.3 SUMMARY

Fog computing is an emerging trend which has been the recipient of increasing interest from industrialists and academia alike. In this chapter, a structured review of the state-of-art in Fog computing is presented. The review specifically emphasised on the platform management concerns and application domain aspects of Fog computing. The

2. Literature Review

study presented in this chapter captures the applicability domains of Fog computing and provides a comprehensive review of recent advancements in this area. Additionally, a taxonomy to organise the state-of-the-art under different categories based on the different aspects of Fog computing, is also presented. The research gaps identified in the context of Fog platform management are also highlighted.

CHAPTER 3

PROBLEM DESCRIPTION

The Fog Computing paradigm was primarily introduced to accelerate the processing of IoT applications. Data collected by the sensors and other devices at the edge will be continuously streamed to the Fog Nodes, where a preliminary processing or a decision regarding the processing will be made. The major decision to be made is regarding the processing location. To ensure the characteristics of Fog such as low latency, location awareness and mobility support, the processing should be delegated to the most appropriate Fog node. The mobility support attribute of Fog computing implies that irrespective of the user locations, they are entitled to avail the Fog services. Considering the heterogenous and dynamic nature of the Fog environment, efficient orchestration systems are required to enable processing of the distributed IoT applications and ensure seamless service delivery.

3.1 SCOPE AND FOCUS OF THE THESIS

The Fog ecosystem provides a platform for distributed processing of IoT applications. Efficient mechanisms to manage the run time platform is essential to ensure that applications execute successfully in the Fog environment. This thesis investigates the orchestration challenges involved in enabling distributed processing in the Fog computing environment. The objective of this thesis is to enhance the performance of Fog environments by employing efficient orchestration mechanisms. The focus is on han-

3. Problem Description

Facet	Thesis Scope
Target systems	Fog computing environments
Virtualization	Container Virtualization
System resources	Processing and memory resources
Application workload	Real-time applications
Orchestration techniques	Application module placement and Application module migration

Table 3.1: Scope and Focus of this Thesis

Addressing the deployment and runtime challenges of the modular IoT applications in the Fog ecosystem (as illustrated in Table 3.1).

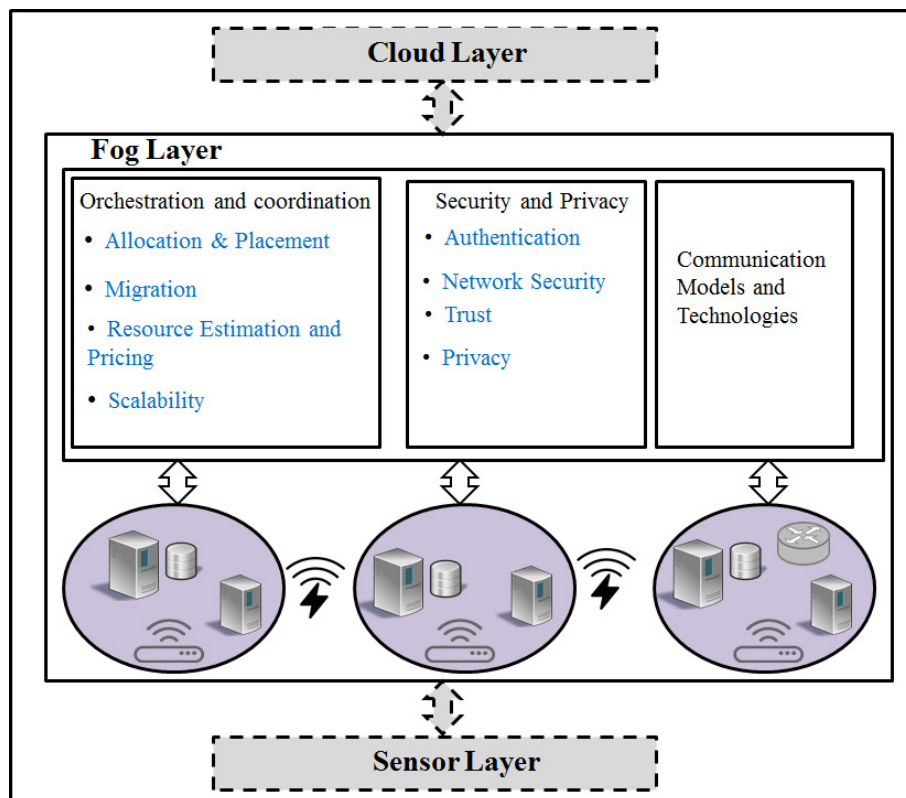


Figure 3.1: An orchestration framework for Fog computing Environment

The Fog environment includes a Fog Layer, which consists of Fog nodes. The Fog layer interplays with the Cloud Computing Layer and the Edge Computing Layer. The data generated from the sensor devices in the Edge Computing Layer, must be processed in close proximity with the data source. The Fog layer has several responsibilities which include coordinating and communicating with the edge devices, communicating with the Cloud data centers and coordinating the several Fog nodes to process the data received from the edge devices, in a secure, reliable and efficient manner.

Figure 3.1 highlights the Fog platform ecosystem, where applications are deployed in the Fog environment on Fog nodes. The Fog Layer includes units responsible for orchestration and coordination, units for ensuring security and privacy, and units to enable communication, which include communication models and technologies. The orchestration and coordination unit performs the dynamic allocation and management of applications. Our focus is on how different applications can be managed to ensure efficient processing. In other words, this work focusses on the deployment and runtime challenges of the Fog applications.

3.2 RESEARCH PROBLEM AND OBJECTIVES

The aim of this thesis is to tackle challenges related to resource orchestration mechanisms applicable for Fog environments in which IoT applications are processed. Specifically, this thesis explores the research problems of application module placement and application module migration. Accordingly, the research problem is stated as follows:

”To design and develop dynamic decision making schemes for the optimal placement and migration of application modules in the Fog computing environment”.

The first goal of this work is to develop an application module/service placement strategy for selecting the most suitable nodes to deploy different services. The application placement problem is an NP-hard problem (Urgaonkar et al. 2007). Application placement strategies must be devised by carefully considering the characteristics of the application services and the resources available, to ensure optimized performance of the application. The placement decision must aim to improve the QoS. The highly distributed and heterogeneous features of the Fog environment introduce more complexities in the selection of the nodes to host the application services, when compared to the centralised systems.

The second goal of this work is to design and develop a framework for the autonomic mobility-driven migration of application modules. The heterogeneous, dynamic nature of the Fog environment and mobility of user devices demands efficient reallocation of application modules. In order to ensure timely service delivery, it is essential to ensure that the control of processing is relinquished to a Fog node which is closer to the access

point to which the user device is currently connected. The migration of an application module is a multistage process which involves identifying when to initiate a migration, identifying the modules to be migrated and determining the destination nodes for the migrating modules.

This aim can be decomposed into 4 research objectives.

3.2.1 Research Objectives

1. To formulate a mathematical model that characterizes the performance metrics of Fog environments.
2. To design and develop a QoS-aware decision-making strategy using the developed model for the placement of application modules in Fog environments.
3. To propose a machine intelligence based model to pre-determine the migration of application modules in Fog.
4. To design and develop a mobility-driven autonomic approach for the migration of application modules.

3.3 RESEARCH METHODOLOGY

Research paradigms in information system discipline can be categorized into two - behavioural science and design science (Hevner et al. 2004).

Design science involves the creation of solutions for various problems identified. The research in this thesis is in alignment with the design science paradigm. The methodology adopted in this thesis is as shown in Figure 3.2. The artifacts developed from this thesis involves a set of models, algorithms and frameworks. Considering the general guidelines on design science methodology, research activities were carried out according to the following steps:

1. Conducting Research Literature Review and Defining Research Problem:

An exhaustive literature review on Fog computing was conducted and based on the review, the research gaps were identified. The primary research problem and

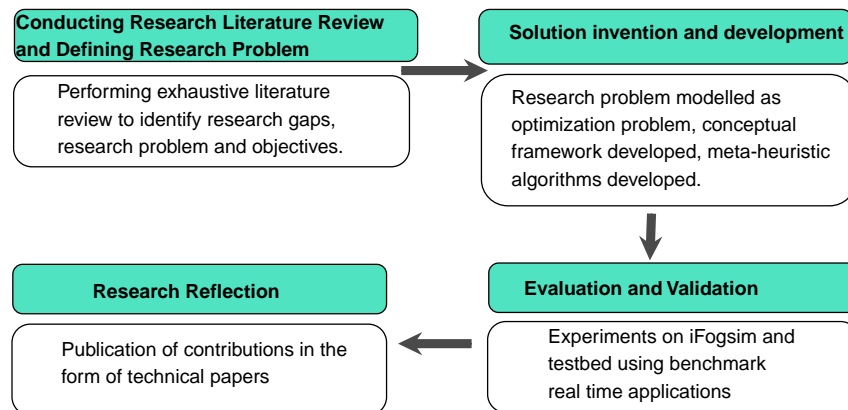


Figure 3.2: Overview of our Research Methodology

ensuing research objectives were formulated.

2. **Solution invention and development:** This phase involved designing solutions for the research problem and objectives. The solution includes mathematical models, conceptual frameworks and algorithms.
3. **Evaluation and Validation:** The evaluation and validation of the developed solutions were carried out in this phase. The initial placement and migration approaches were evaluated using real time applications on testbed and iFogsim simulator.
4. **Research Reflection:** This phase involved the activities carried out to illustrate the impact of this research on the scientific community. In this regard, the outcomes of research were communicated through publications in conference proceedings and International acclaimed journals.

3.4 RESEARCH CONTRIBUTIONS

The major contributions of this thesis can be mainly divided in to three (as shown in Figure 3.3). This includes a systematic analysis of related works in the Fog domain, a strategy for application module placement that considers different QoS parameters and an autonomic framework which supports the migration of application modules to provide mobility support in the Fog computing environment. The detailed contributions

3. Problem Description

are as follows:

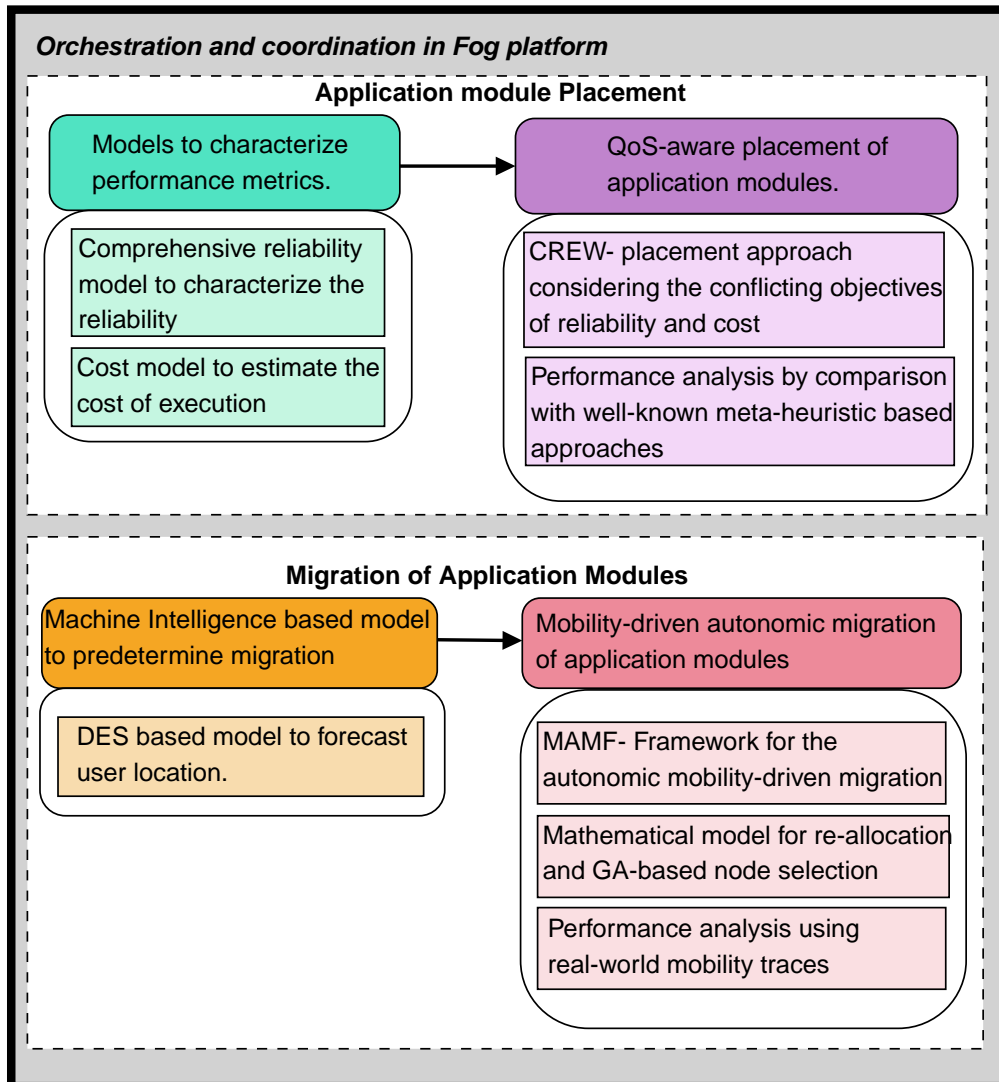


Figure 3.3: Contributions of this Thesis

1. An extensive study on Fog Computing with special focus on Fog platform management concerns.
 - A multi-level taxonomy that categorises the existing research on Fog Computing, based on the aspect considered in each research work.
 - A comprehensive review of the different research works that deal with different aspects of Fog Computing.

2. A strategy for cost and reliability aware application module placement in Fog computing environment.
 - A comprehensive reliability model to characterize the reliability in Fog environments. The considers the different types of failures in the hierarchical Fog environments.
 - A cost model to estimate the costs for execution in different environments. The cost model depends on application-specific characteristics and execution times evaluated by time estimation models.
 - A novel placement approach, **CREW**, which considers the conflicting objectives of reliability and monetary cost of execution of the application.
 - A performance analysis of the proposed approach and comparison with existing well known meta-heuristic algorithms such as NSGA-II and MOWOA.

3. A mobility aware autonomic approach for the migration of application modules in fog computing environment.
 - A Double Exponential Smoothing (DES) based model to forecast user location.
 - A conceptual framework for the autonomic mobility-driven migration of application modules.
 - A mathematical model representing the optimization problem in the re-allocation phase and a Genetic Algorithm (GA) based approach to determine the target node for the migrating application modules.
 - Performance analysis of the proposed approach and comparison using real world mobility traces.

The work presented in this thesis addresses the objectives listed in Section 3.2. The contributions drawn from the thesis corresponding to the different objectives are provided in Table 3.2. Due to the logical dependencies between the objectives, Research Objective (RO) 1 and RO 2 have been jointly presented in Chapter 4. Similarly RO 3 and RO 4 have been presented in Chapter 5.

Research Contribution	Specifics of Research Contribution	Research Objective (RO) Addressed	Thesis Chapter Presenting Contribution	Chapter derived from Publication
Extensive study on Fog Platform Management	Multilevel Taxonomy		Chapter 2	John et al. Elucidating the challenges for the praxis of fog computing: An aspect-based study. International Journal of Communication Systems, Wiley 2019
	Comprehensive review of research works			
Strategy for Cost and reliability aware application module placement in Fog environment.	Reliability Model	RO 1	Chapter 4	John et al. CREW: Cost and Reliability aware Eagle-Whale optimizer for Service Placement in Fog. Software: Practice and Experience, Wiley 2020.
	Cost Model	RO 1		
	CREW Placement Approach	RO 2		
	Performance Analysis	RO 2		
Mobility Aware Approach for Autonomous Migration	DES based Model	RO 3	Chapter 5	John et al. Mobility aware autonomic approach for the migration of application modules in fog computing environment. Journal of Ambient Intelligence and Humanized Computing, Springer, 2020.
	Conceptual Framework	RO 4		
	Mathematical Model	RO 4		
	Performance Analysis	RO 4		

Table 3.2: Details of Contributions of this Thesis

CHAPTER 4

COST AND RELIABILITY AWARE EAGLE-WHALE OPTIMIZER FOR SERVICE PLACEMENT IN FOG

Service placement strategies assign incoming application services to the appropriate Fog nodes. The strategies must efficiently place the application services in a performance optimized manner. The highly distributed and heterogeneous features of the Fog environment introduce more complexities in the selection of the nodes to host the application services, when compared to the centralised systems.

The service placement problem in Fog computing environment has been addressed by multiple researchers. Research has been directed to propose several solutions based on heuristic algorithms (Hong et al. 2016; Mahmud et al. 2019a), genetic algorithms (Wen et al. 2017), Petri nets (Ni et al. 2017), graph theory (Lera et al. 2018) and linear programming approaches (Velasquez et al. 2017; Zeng et al. 2016).

Brogi and Forti (2017) proposed a heuristic based QoS aware service placement algorithm for IoT applications in the Fog computing environment. They have mainly considered latency and bandwidth constraints for determining the placement. Developers are also required to provide component binding details. A detailed trade-off analysis between power consumption and delay in Fog computing environments is presented by Deng et al. (2016). They have not made any distinction in the type of resource requests, rather they have only considered the workload request rates. Guerrero et al. (2019) pro-

posed a decentralised service placement policy which aims to place the services near to the end users. Their heuristic algorithm is executed locally on each device and allows the devices to take decisions regarding offloading and processing. Skarlat et al. (2017) proposed a Fog colony based framework for Fog computing environments. They also modelled a service placement scheme which aims to maximise the number of services placed in Fog computing environment. They have not considered the monetary cost of execution in the determination of placement decisions. Venticinque and Amato (2019) proposed a service placement scheme which extends the research work by Skarlat et al. (2017). They evaluated the performance in smart energy domains based on the BET methodology (Benchmarking, Evaluation and Testing activities). Zhang et al. (2017b) modeled the service placement problem as a Stackelberg game. They considered application of Fog computing in a data service environment and a matching game is used to obtain the mapping between providers and Fog nodes. Liu et al. (2017) proposed an offloading decision model based on queuing theory. The proposed model considered energy consumption delay and cost while taking the offloading decisions. He et al. (2018) proposed a resource allocation and offloading scheme for Fog computing environment. Network bandwidth, latency, costs for computation and communication are considered for taking decisions. They have considered only synthetic workloads. The characteristics of real time applications were not considered.

Finding the most appropriate location for service placement is an NP-Hard problem. Thus, there still exists much scope for further improvement. Most of the existing solutions are focused on reducing the response time, energy, network usage and cost. Due to the heterogeneous and distributed characteristics of Fog environments, its susceptibility to failure is high. To improve the user QoS, service reliability must be maintained. Reliable service delivery ensures the delivery of essential services without interruption and failure for the time of period under consideration (Sharma et al. 2016). To the best of our knowledge, none of the previous studies have considered the reliability aspects while taking placement decisions. The efforts for increasing the reliability of the system generally leads to an increase the overall cost. Hence, there is a need for service placement schemes that consider the conflicting objectives of maximizing reliability

and minimizing cost.

In this chapter, we have proposed a service placement policy named Cost and Reliability aware Eagle-Whale optimizer (CREW). The proposed approach derives an optimal placement mapping of application modules to the appropriate fog nodes with the objectives of maximizing reliability and minimizing cost. CREW addresses these conflicting objectives and also ensures the delivery of services within the specified deadlines. CREW adopts multi-objective meta-heuristic based techniques to derive solutions for the efficient placement of services in Fog computing environment. Two well known meta-heuristic techniques, the Whale Optimization and Eagle strategy, are combined to form a hybrid strategy that is applied to solve the placement problem. The services of an application are distinguished based on relevance of the components in the overall working of the application into mandatory and non-mandatory components. The application fails when any of the mandatory component fails. CREW aims at enhancing the application reliability by ensuring high reliability values for the mandatory components.

The remainder of this chapter is organised as follows: the architecture of the proposed system is explained in Section 4.1. Section 4.2 presents the modelling and formulation of the optimisation problem. The proposed solution is explained in Section 4.3. The Section 4.4 describes the experimental design details and results are provided in Section 4.5. The chapter is concluded in Section 4.6.

4.1 SYSTEM ARCHITECTURE

The conceptual Fog computing model proposed by Industrial Internet Consortium (IIC) consists of 3 layers namely the Cloud computing layer, Fog computing layer and the end devices. Skarlat et al. (2017) proposed a Fog computing framework spanning these three layers. In this research, we have also used and extended the same model as shown in Figure 4.1 for our experimentation. The set of all devices in the system (D_i) is divided into three different classes: the Cloud Devices (C), Fog Control nodes (FCN) and Fog Cells (FC). The number of devices in D is represented by p . The set of all the Fog nodes is represented as F and the number of Fog nodes in the environment is denoted as N . The Fog environment is further divided into different Fog colonies, which

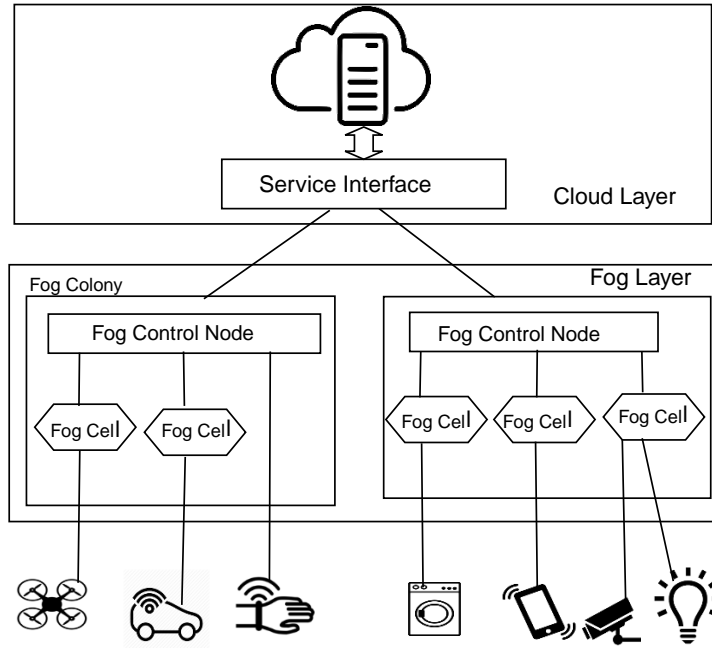


Figure 4.1: Three tier Fog computing Environment

consists of Fog cells headed by a Fog control node. The Fog control node is responsible for taking decisions regarding the deployment of services within the colonies and also takes offloading decisions. Fog cells constitute the fog devices lowest in the hierarchy. They have limited computational resources and are connected to the IoT devices i.e, sensors, actuators and others. The communication links between Fog cells and Fog Control nodes possess negligible communication delay whereas the link between the Fog Control node and Cloud possesses a link delay d_i .

Figure 4.2 shows the architecture of the proposed system. The application deployment requests are submitted to the Fog Control node. A Fog node f_j is characterised by a vector f_j^{avail} which includes the available number of CPU cores ($f(cpu)_j^{cap}$), amount of memory available ($f(mem)_j^{cap}$), and available storage ($f(storage)_j^{cap}$). For simplicity, but without loss of generality, we have considered only CPU and memory resources. Applications conforming to the microservice architectural style are increasingly being used in the IoT environment.

The microservice based applications consist of different self-contained stateless ser-

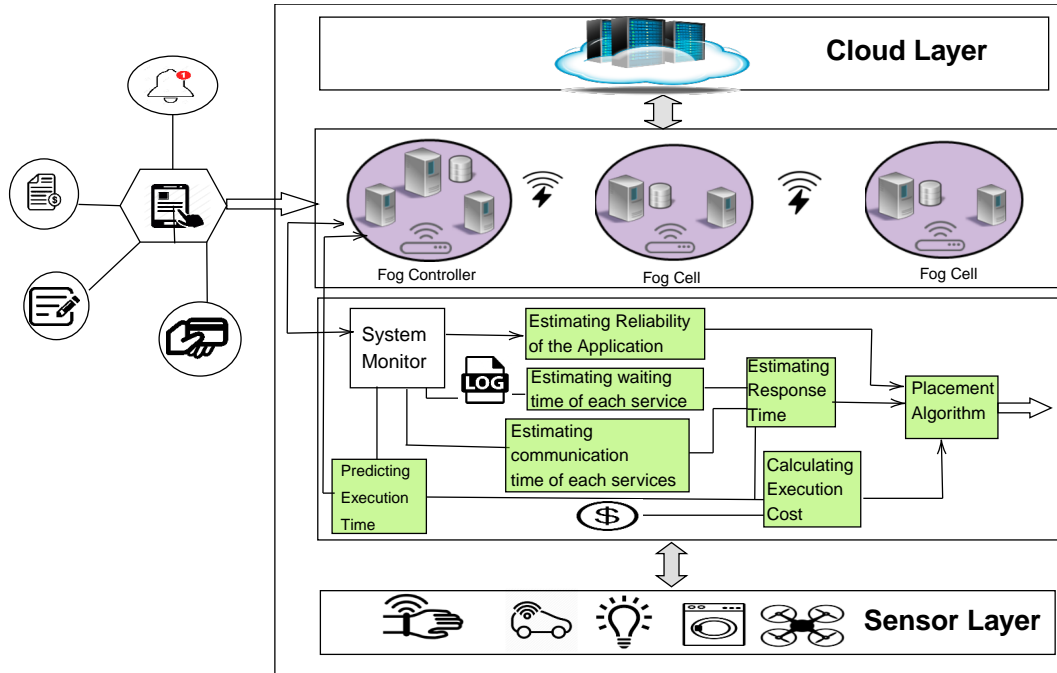


Figure 4.2: System architecture

vices which implement a single business capability. The different entities interact with each other through lightweight communication protocols. A service in an application X can be denoted as s_i . An application comprises of a set of M services running in the Fog environment and the set of all services is represented as S . In this work, the terms module and component have been used interchangeably to refer to a constituent service of the application. The number of instances of a service depends on the user demands and nature of service it provides. To simplify the definition of the problem, we consider that each service has l number of instances. Services of an application are marked as either mandatory or non-mandatory. $man(X^m)$ is a binary variable that takes the value of '1' if the m^{th} service of an application X is mandatory and zero otherwise. Mandatory components have higher priority and represent the minimal services that should always be functional to ensure that the application is up. For example, in an e-commerce industrial application, the recommendation module can be considered to be non-mandatory whereas billing, cart etc. are mandatory services. The failure of mandatory components leads to failure of the application. Each s_i is allocated to a suitable Fog node which satisfies its resource requirements. The requirements of the i^{th} service can be denoted by a tuple $\langle s(cpu)_i^{req}, s(mem)_i^{req} \rangle$, where $s(cpu)_i^{req}$ is the number of CPU cores

Notations	Description
D	Set of all the devices in the System.
C	Set of Cloud devices in the System.
FCN	Fog Control Node
FC	Fog Cell
p	# devices in D .
F	Set of all available Fog Nodes in the Fog environment
N	# of Fog Nodes available in the system
d_i	link delay between Fog Control node and Cloud
$f(cpu)_j^{cap}$	# of CPU cores available on the Fog Node, f_j
$f(mem)_j^{cap}$	Amount of memory available on the Fog Node, f_j
X	Application
s_i	Service of an Application X
M	# services of an application
S	Set of all services of an application
l	# instances of a service type
$s(cpu)_i^{req}$	# of CPU cores requested by i^{th} service
$s(mem)_i^{req}$	Amount of memory requested by the i^{th} module

Table 4.1: Notations used in the System model

requested and $s(mem)_i^{req}$, is the amount of memory requested by the i^{th} module. A summary of the notations used in this work is given in Table 4.1.

Service deployment activities are initiated on the arrival of each service request and are controlled by the FCN . The sequence of activities involved in deployment/offloading decisions are as given in Figure 4.2:

- **System Monitor:** The system monitor in the FCN monitors the resource available at each node, current resource usage statistics, request arrival rate and service rate. The System monitor also keeps track of service resource requests, deadlines, service resource usage and execution details.
- **Predicting Execution Time:** For each incoming service s_i , the execution time of the service ($ET(s_i)$) is predicted using machine learning techniques. The overall execution time of the application is calculated based on the sum of the execution time of all the services of the application.
- **Estimating waiting time of each service:** Queuing models are used to estimate the waiting time ($WT(s_i)$) of the service at different deployment levels/de-

services. The overall waiting time is then calculated based on the sum of the waiting time of services.

- **Estimating communication time of each service:** The time involved for the communication ($Com(X)$) which contributes to the overall response time is calculated based on the communication model.
- **Estimating response time:** The response time of the application is estimated as a function of the waiting time, execution Time and communication Time.
- **Estimating reliability of the application:** Applying the reliability model, the reliability of the application ($R(X)$) is calculated for the different deployment configurations.
- **Calculating execution cost:** The execution cost of the application ($E(X)$) in different deployments is calculated based on the cost matrices.
- **Placement algorithm:** The placement algorithm determines the optimal placement for the services based on application response time, cost and reliability factors.

In this work, the primary aim is to reduce the cost of service placement and maximize service reliability. The provider should also be able to provide service responses within the tolerable delay. The following sections describe in detail the different components involved in the proposed system.

4.1.1 Service Execution Time Prediction

Machine learning based models are used to predict the execution time of the services for different placement configurations. The execution time of a service is largely determined by the software features of the service (program) and the hardware profile of the device on which it executes. The primary hardware features include the *size of total RAM* and *size of used RAM*, which quantifies the amount of workload in the system and the *cache or buffer size* which influences the performance of I/O operations in the system. The values for these parameters are obtained using system commands like the

4. Cost and Reliability aware Eagle-Whale optimizer for Service Placement in Fog

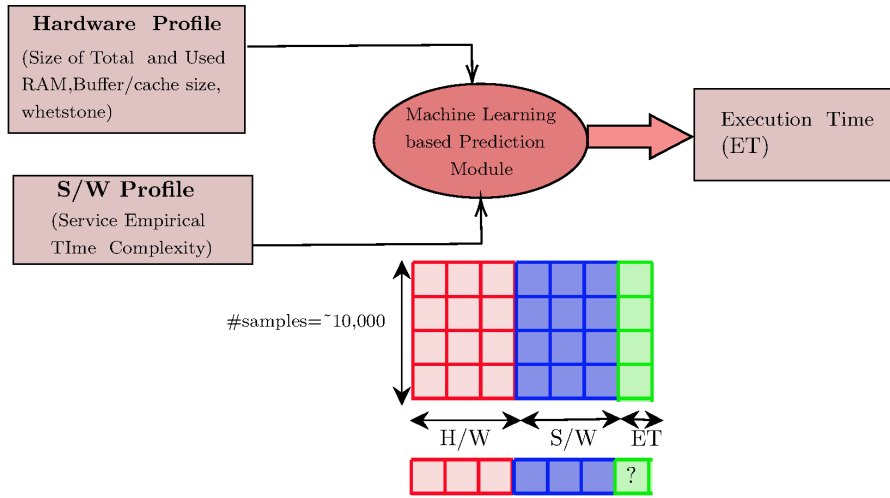


Figure 4.3: Service Execution Time Prediction process - The machine learning based prediction module takes the hardware profile features and the software profile features as inputs to estimate the execution time. The ML model is trained using a dataset of around 10k samples with input values and corresponding output values. The dataset was generated using experimental data.

free command. The processing capability is derived from the floating-point operations per second rating of the processor, rather than taking the CPU clock speed. This value is obtained using the *Whetstone* benchmark (Bao et al. 2019). The computational time complexity of the service module is calculated using the TPROF tool (Goldsmith et al. 2007). An overview of the prediction model is shown in Figure 4.3.

Based on the preliminary investigations conducted (details provided in Section 4.5.2), it was observed that the Extreme Gradient Boosting technique (XGBOOST) model (Chen and Guestrin 2016) is best suited for our system. In the proposed CREW system, XGBOOST was employed to predict the execution time of the services ($ET(s_i)$).

The overall application execution time is calculated as the sum of the individual service execution times (as provided in Equation 4.1)

$$Exe(X) = \sum_{i=0}^{M*l} ET(s_i) \quad (4.1)$$

4.1.2 Waiting Time Model

The execution environment consists of D devices which include the Cloud server and the set of Fog devices F . The application deployment requests are submitted to the Fog Control Node as shown in Figure 4.4. The inter-arrival time of requests is represented as a Poisson random variable with arrival rate λ . The service rate of the queue is exponentially distributed with service rate, μ . Based on the available capacity of devices, the traffic flow at the Fog cells can be represented by $M/M/1$ queues (single server), at the Fog Control nodes as $M/M/c$ queues (finite number of parallel servers) and at Cloud as a $M/M/\infty$ queue (large number of parallel servers) (Liu et al. 2017). All these three queuing models perform scheduling according to the First Come First Serve (FCFS) discipline. The waiting times at Fog Cells and Fog Control Nodes can be calculated using the Equations 4.2 and 4.3 respectively (Gross 2008).

$$WT(s_i^{FC}) = \frac{1}{(\mu_i - \lambda_i)} - \frac{1}{\mu_i} \quad (4.2)$$

$$WT(s_i^{FCN}) = P_q * \frac{1}{(c\mu_i - \lambda_i)} \quad (4.3)$$

$$\begin{aligned} \text{where, } P_q &= \frac{v}{u + v} \\ v &= \frac{(c\rho)^c}{c!(1 - \rho)} \\ u &= \sum_{i=0}^{c-1} \frac{(c\rho)^i}{i!} \end{aligned}$$

where c is the number of Fog Control Nodes, ρ is the utilisation of the node calculated as $\rho = \mu/\lambda$

Waiting time of the application $WT(X)$ is calculated by adding the waiting time of all the services which depends on where the services are placed. Since the Cloud possesses virtually unbounded resource capacity, the waiting time at the Cloud is taken as zero.

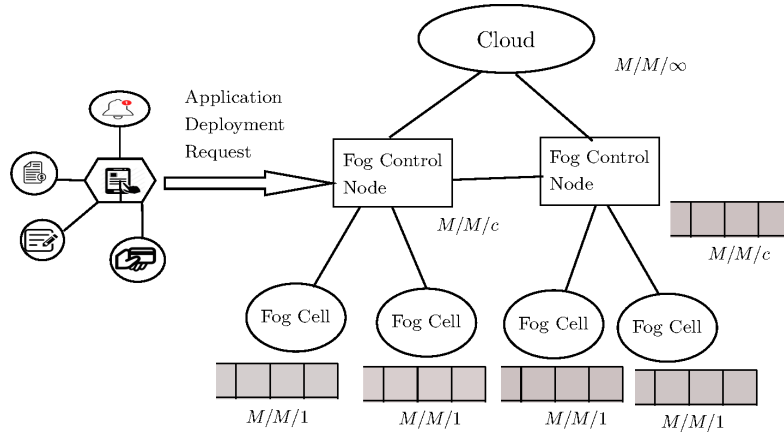


Figure 4.4: Model of the inter-connected queuing system in hierarchical Fog environments

4.1.3 Communication Time Estimation

Application response time includes the communication time required to transfer the data between the devices in different levels. The communication link between two nodes i and j is characterised by a constant propagation delay (d_{pd}) and bandwidth ($bw_{i,j}$). The communication time between the fog cells and Fog control nodes within the same network contributes negligible delay in comparison to the delay involved in communication with the Cloud, which can be estimated as given in Equations 4.4 and 4.5.

$$Com_{i,j}(s_i) = d_{pd} + \frac{s(data)_i^{req}}{bw_{i,j}} \quad (4.4)$$

$$Com(X) = \sum_{i=0}^{M*l} Com_{i,j}(s_i) \quad (4.5)$$

where $s(data)_i^{req}$ refers to the amount of data required by the service s_i that must be transferred through the considered link.

4.1.4 Response Time Estimation

The response time of an application is computed based on the summation of waiting time, execution time and communication time for all the services comprising the appli-

cation as given in Equation 4.6.

$$ReTime(X) = WT(X) + Exe(X) + Com(X) \quad (4.6)$$

4.1.5 Reliability Model

Application reliability is defined as the ability of the system to ensure that the application is up and running within a specified operating period, which implies ensuring that at least the mandatory service components of the application run as expected for a desired time duration i.e, atleast one instance of each mandatory component should be up and running at all times. Few considerations are taken as follows:

- Let $q^{k,m}$ be the failure probability of k^{th} instance of the service type $m \in [1, M]$.
- A Fog node f can fail with the probability of q^{f_j} independent of other Fog nodes and services running on it. When a Fog node fails, all the service instances on it fails.
- An instance of a service fails either due to the failure of the Fog node on which its deployed or due to the failure of the microservice itself.
- A service/component of an application fails when all the instance of that service fails.
- An application fails when any of the mandatory components fails.

The unavailability of a service may be either due to the failure of the service or due to the failure of the host node on which the service is deployed. The probability to failure for k^{th} instance of service type m is given in Equation 4.7.

$$Pr(failure\ of\ (k, m)) = q^{k,m} + Pr(failure\ of\ Fognode\ running\ (k, m))$$

$$Pr(failure\ of\ (k, m)) = q^{k,m} + \sum_{j=1}^N X_j^{k,m} * q^{f_j} \quad (4.7)$$

The service reliability is calculated using Equation 4.8.

$$Pr(\text{failure of service } m) = \prod_{k=1}^l (q^{k,m} + \sum_{j=1}^N X^{k,m} * q^{f_j}) \quad (4.8)$$

An application fails when any of the mandatory components fails. Thus, application reliability is calculated using the Equation 4.9.

$$Pr(\text{failure of application } X) = \sum_{m=1}^M \prod_{k=1}^l (q^{k,m} + \sum_{j=1}^N X^{k,m} * q^{f_j}) * man(X^m) \quad (4.9)$$

The overall reliability is calculated using Equation 4.10.

$$R(X) = 1 - \sum_{m=1}^M \prod_{k=1}^l (q^{k,m} + \sum_{j=1}^N X^{k,m} * q^{f_j}) * man(X^m) \quad (4.10)$$

4.1.6 Cost Model

The monetary cost of execution of an application depends on the execution time of the services which constitutes the application and also on the node in which those services are deployed. The service components can be deployed either on Cloud devices or on any Fog device F . A non zero value for the binary variable $X_{d_h}^{k,m}$ indicates that the k^{th} instance of service type m is deployed on the device d_h . The cost of execution of an application with M modules and each with l instances is given in Equation 4.11.

$$EC(X) = \sum_{m=1}^M \sum_{k=1}^l (X_{FC}^{k,m} E_{FC}^m C_{FC}^m + X_{FCN}^{k,m} E_{FCN}^m C_{FCN}^m + X_C^{k,m} E_C^m C_C^m) \quad (4.11)$$

where E_f^m = Execution time of service of type m on 'f' and C_f^m = Cost of execution on a device D per unit time.

4.2 PROBLEM FORMULATION

In the previous section, the important metrics which are essentially considered while taking decisions regarding the placement of services in the Fog computing environment were discussed. The cost of execution and reliability of the application also depends on where each of its service components are deployed. The binary decision variable of the problem represents the deployment of k^{th} instance of service type m on Fog node f . The decision variable can be defined as Equation 4.12.

$$X_f^{k,m} = \begin{cases} 1, & \text{If } k^{th} \text{ instance of service type } m \text{ is deployed on Fog node 'f'} \\ 0, & \text{Otherwise} \end{cases} \quad (4.12)$$

where f can be either a Fog Cell (FC), a Fog control node (FCN), or Cloud (C).

The two objectives of minimizing execution cost and maximizing reliability have been considered as given in Equations 4.13 and 4.14.

$$\text{Minimize}(EC(X)) \quad (4.13)$$

$$\text{Maximize}(R(X)) \quad (4.14)$$

subject to,

$$\forall f \in F, \sum_{i=1}^N X_f^{k,m} (cpu)_i^{req} \leq f(cpu)_j^{cap} \quad (4.15)$$

$$\forall f \in F, \sum_{i=1}^N X_f^{k,m} (mem)_i^{req} \leq f(mem)_j^{cap} \quad (4.16)$$

$$X_{FC}^{k,m} + X_{FCN}^{k,m} + X_C^{k,m} = 1 \quad (4.17)$$

$$WT(X) + Exe(X) + Com(X) \leq D \quad (4.18)$$

$$R(X) \geq R \quad (4.19)$$

$$C(X) \leq C \quad (4.20)$$

$$X_{FC}^{k,m}, X_{FCN}^{k,m}, X_C^{k,m} \in \{0, 1\} \quad (4.21)$$

Constraints 4.15 and 4.16 ensure that sufficient CPU and memory resources are available on the Fog devices to satisfy the resource requirements of the hosted applications. Constraint 4.17 ensures that each service instance is deployed on one and only one fog device. Constraints 4.18 - 4.20 guarantee that the response time, reliability and cost are within the tolerable limits. Constraint 4.21 specifies that the decision variables are binary variables and can take the values of either 0 or 1 only.

4.3 CREW-COST AND RELIABILITY-AWARE EAGLE-WHALE OPTIMIZER

The proposed CREW service placement policy finds an optimal mapping of microservice modules of an application to the most suitable fog devices while maximizing the service reliability and minimizing the cost of execution. Reliability and cost of execution are two conflicting objectives. The mapping of services to the Fog devices is a NP-hard problem. The novel Cost and Reliability aware Eagle-Whale optimizer (CREW) for the efficient placement of services in Fog computing environment uses multi-objective meta-heuristic based techniques to derive solutions for the placement problem. CREW effectively combines two bio-inspired techniques, the Eagle Search Strategy and Whale Optimization.

4.3.1 Whale optimisation Algorithm

Whale optimisation Algorithm (WOA) is a bio inspired meta-heuristic optimisation algorithm (Mirjalili and Lewis 2016). WOA is based on the bubble net attacking strategy of humpback whales. The positions of search agents are updated using one of the three methods: Random search method, Shrinking encircling approach or Spiral feeding method.

The humpback whales can locate the position of the prey and swim around the prey simultaneously tracing the surface of a shrinking circle and a spiral path. Thus exploitation is done in two ways. The equation for *Shrinking Encircling* is given as

4.3. CREW-Cost and Reliability-aware Eagle-Whale Optimizer

Algorithm 4.1: CREW Algorithm for FSSP

```

Input : Application  $A = \langle \text{List of modules, List of mandatory components} \rangle$ 
Input : Service Modules  $S = \langle s(cpu)_i^{req}, s(mem)_i^{req} \rangle$ 
Input : Set of Fog Nodes,  $F$ 
Output: Mapping of each service module to a feasible node, Allocation_vector
1 function CREW:
2   Generate initial whale (search agents) randomly
3   Evaluate fitness value for individual search agents
4   Identify non-dominated solution
5   Update the archive with regard to the obtained non-dominated solutions
6   while  $t < num\_iteration$  do
7     for each agent do
8       Determine probability,  $prob = 0.3 * (1 - \frac{t}{num\_iteration})$ 
9       Generate a second random number  $q$ 
10      if  $q < prob$  then
11        Update position of current search agent by  $X_j = X_{jmin} + rand(X_{jmax} - X_{jmin})$ 
12      end
13      Evaluate fitness value for individual search agents
14      Identify non-dominated solution
15      Update the archive with regard to the obtained non-dominated solutions
16      if the archive becomes full then
17        Call the archive maintenance procedure for removing inferior agents
18        Add the non-dominated agent to the archive
19      end
20      Determine a random number  $rand$ 
21      if  $P_e < rand$  then
22        Perform local search goto Line 27
23      else
24        Perform global search goto Line 40
25      end
26    end
27    for each search agent do
28      Assign updated values for  $a, A, C, l, p$ 
29      if  $p < 0.5$  then
30        if  $|A| < 1$  then
31          update position of current search agent by  $\vec{D} = |\vec{C} \cdot \vec{X}(t) - \vec{X}^*(t)|$ 
32           $\vec{X}(t+1) = \vec{X}(t) - \vec{A} \cdot \vec{D}$ 
33        end
34      end
35      else if  $p \geq 0.5$  then
36         $\vec{X}(t+1) = \vec{D}e^{bi} \cos(2\pi l) + \vec{X}^*(t)$ 
37         $\vec{D} = \vec{X}^*(t) - \vec{X}(t)$ 
38      end
39    end
40    Alter each search agent that goes beyond search space
41    Evaluate fitness value for individual search agents
42    Determine non-dominated solutions
43    Update the archive with regard to the obtained non-dominated solutions
44  end
45  if the archive becomes full then
46    Call the archive maintenance procedure for removing inferior agents
47    Add the non-dominated agent to the archive
48  end
49  Update  $t = t + 1$ 
50  Stop when termination condition reached
51 end
52 Obtain  $\vec{X}^*$  from the archive
53 Return  $\vec{X}^*$ 
54 end

```

Equation 4.22:

$$\vec{X}(t+1) = \vec{X}(t) - \vec{A}.\vec{D} \quad (4.22)$$

$$\vec{D} = \vec{C}\vec{X}^* - \vec{X}(t)$$

$$\vec{A} = 2\vec{a}\vec{r} - \vec{X}(t)$$

$$\vec{C} = 2.\vec{r}$$

\vec{r} is a random vector $\in [0, 1]$

\vec{a} is linearly decreased from 2 to 0

The *spiral updating position* follows a spiral path between the current position and prey as given in Equation 4.23.

$$\vec{X}(t+1) = \vec{D}e^{bl}.\cos(2\pi l) + \vec{x} * t \quad (4.23)$$

where l is a random number within range $[-1, 1]$ and b ensures logarithmic shape.

The *exploration* of the algorithm is ensured by the Random search technique of the whale which is decided by \vec{A} , as given in Equation 4.24.

$$\vec{X}(t+1) = X_{Rand}(t) - \vec{A}.\vec{D} \quad (4.24)$$

4.3.2 Eagle Strategy

Eagle strategy is an optimization approach which maintains the balance between the exploration and exploitation Yang and Deb (2010). The eagle strategy performs exploitation process inspired by the hunting strategy of eagles. Initially, it searches for the prey and after finding the prey it changes its chasing behaviour to mimic an attacking behaviour. Eagle strategy uses a parameter P_e for establishing a balance between exploration and exploitation and to prevent premature convergence. Eagle strategy considers an agent for exploration based on the probability calculated Gavvala et al. (2019) as given in Equation 4.25.

$$prob = 0.3 * \left(1 - \frac{iter}{num_{iter}}\right) \quad (4.25)$$

where $iter$ is the current iteration number and num_{iter} is the maximum number of iterations. For each agent, a random number $q \in [0, 1]$ is generated and compared with $prob$. If the q value is less than $prob$, then the position of the whale is updated using the Equation 4.26.

$$X_j = X_{jmin} + rand(X_{jmax} - X_{jmin}) \quad (4.26)$$

$$rand \in [0, 1]$$

where X_{jmin} is the minimum and X_{jmax} is the maximum values of the currently active agent.

The proposed approach, CREW, incorporates eagle strategy for the exploration purpose. Combining eagle strategy with whale strategy, aids in the prevention of premature convergence when the population is having low diversity and also improves the convergence rate when the population diversity is very high. The detailed pseudocode of proposed CREW approach is given in Algorithm 4.1.

The objective of CREW is to find a solution for the optimal mapping of service modules to feasible nodes. The input of the algorithm contains the modules to be mapped, their resource requirements, a list of mandatory components and the available Fog nodes. The detailed service placement based on CREW is described as follows:

1. Problem Encoding & Initialisation

The position of the whale represents possible solutions to the optimization problem. The main objective of our system is to find an optimal mapping between the services of application and most suitable nodes in an Fog computing environment. A whale position x_j represents a vector with $M * l$ dimensions, where $M * l$ is the total number of services to be placed. Figure 4.5 illustrates an agent/whale represented as an array, where the index of the array elements represents the service and value of the array element gives the identifier of the node to which the corresponding service is mapped. The values of the array falls within the range of $[1, p]$ where p is the number of available devices in the corresponding environment. The agents are initialised randomly. Line 2 in Algorithm 4.1 randomly

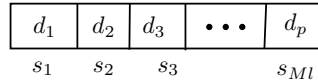


Figure 4.5: Encoding Scheme for the Service Placement Problem

initializes the whale agents to represent different points in the solution space.

2. Fitness function & Iteration Process

Our multi-objective optimization approach considers two objectives which are contradictory. The fitness functions used are *minimizing* $EC(X)$ where $EC(X)$ is given by Equation 4.11 and *maximizing* $R(X)$ where $R(X)$ is given by Equation 4.10. Depending on the number of iterations, the solutions will perform exploration and exploitation processes. After each iteration, non-dominated sorting method is used to update the best agent available in the solution space. The fitness value of each whale in the current population is evaluated in lines 3 - 5 of Algorithm 4.1.

3. Exploration based on Eagle Strategy

In our algorithm, each agent is changed according to a probability given in Equation 4.27.

$$prob = 0.3 \left(1 - \frac{iter}{MAX_{iter}} \right) \quad (4.27)$$

A random number q is generated for the currently active whale within the range of $[0, 1]$ and compared with the value of $prob$ (Gavvala et al. 2019). If the probability is greater than the generated random number q , then another random number is generated within the range $[1, m * l]$ to determine the position of the current active whale that has to be modified. The modification is performed according to Equation 4.26. After performing exploration operation, non-dominated sort is carried out to update any changes to the present Pareto fronts. The lines 8 - 24 in Algorithm 4.1 follow the eagle strategy to perform exploration of the solution space.

4. Exploitation based on Whale attacking Strategy

Exploitation is performed based on the equations used in the WOA algorithm. A

random number p is used to decide whether shrinking encircling mechanism or spiral updating is to be performed. The lines 27 - 37 in Algorithm 4.1 follow the whale strategy to perform exploitation of the current solution space.

$$\vec{X}(t+1) = \begin{cases} \vec{X}(t) - \vec{A} \cdot \vec{D} & \text{if } p < 0.5 \\ \vec{D}e^{bl} \cdot \cos(2\pi l) + \vec{x} * t & \text{if } p \geq 0.5 \end{cases}$$

After performing exploitation, non-dominated sort is again carried out and the best solution is updated.

The inferior agents are removed from the archive and it will be replaced with non-dominated agents. The exploration and exploitation process is repeated in the algorithm till the termination criteria is met. Then, the best solution is returned by the CREW from the archive (Lines 40-53 in Algorithm 4.1).

4.4 EXPERIMENTAL DESIGN AND SETUP

In order to examine and observe the impact of the proposed CREW algorithm based service placement policy, repeated experiments were conducted. The evaluation of the proposal was carried out on a testbed and also simulated over the iFogSim (Gupta et al. 2017) simulator. The experimental setup used for evaluating the fog landscape consists of Fog Control nodes which run on Intel i5 processor laptops with a speed of 3.07 GHz and 4 GB RAM. For the Fog cell devices, Raspberry Pi 3b+ is used with Hypriot Operating System. For cloud resources, Google Cloud Platform (GCP) is used. The fog landscape is implemented using Java 8 and Spring application framework. For communication between different Fog devices, we use REST API and JSON messages. The application deployment requests are received by the Fog Control node. Based on the cost model, reliability model and estimated response time, CREW algorithm decides where to place each services of the application.

4.4.1 Workload applications considered

We considered two different categories of real time microservice-based applications for evaluating the performance of the proposed CREW algorithm. The first category consists of two E-commerce applications and the second category includes a healthcare application. The two categories of applications are described in the subsequent sections.

4.4.1.1 E-commerce Industrial Applications

E-commerce industry is one of the fastest growing industries. According to recent estimates, by 2021 the global e-commerce industrial market sales is expected to reach up to \$5 trillion (Thiebaut 2019). IoT has brought tremendous changes in the e-commerce industry by personalising experiences, automating the entire business process, simplifying decision making and thus changing customer experience altogether. The integration of IoT with e-commerce industry provides automated purchase, RFID based improved supply chain and many more features. Nevertheless, it also demands real time processing of data for better results. Fog computing can be integrated with such systems for handling huge amount of data generated from these environments and providing quick responses. Nowadays, e-commerce applications are mainly developed based on microservice architecture, which can better harness the potential of distributed Fog computing environments.

We have considered two different real time microservice-based e-commerce applications for evaluating the proposed CREW algorithm, namely, Hipster shop and Teastore. The Hipstershop is a 10-tier microservice based e-commerce application which has almost all the functionalities demanded by an e-commerce industry. The application allows the customer to browse items, add the items in to the cart and allows online users to complete the purchase procedure. Figure 4.6 depicts the different interacting entities of the Hipster Shop application. The details regarding memory, resource requirements and the relevance field of each service (Mandatory/Not mandatory) is given in Table 4.2. The CPU resource is expressed in millicores and memory resource is expressed in mebibytes.

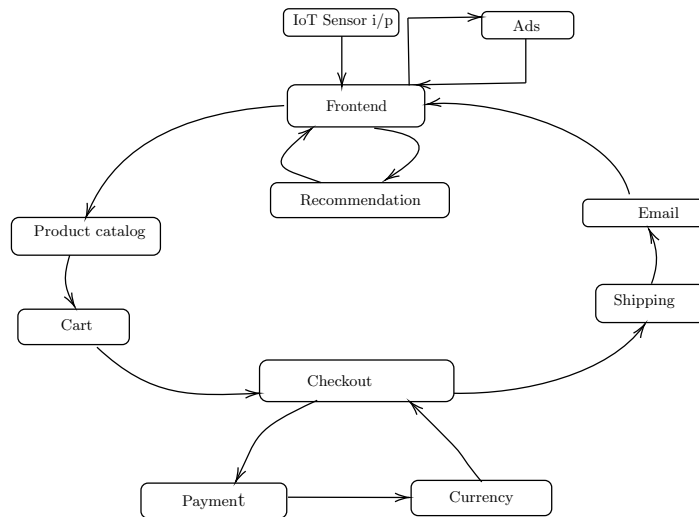


Figure 4.6: Interactions between service components in Hipster Shop Application

Service	CPU	Mem	Mandatory
Adservice	200	180	False
Cart	200	64	True
Checkout	100	64	True
Currency	100	64	False
Email	100	64	False
Frontend	100	64	True
Payment	100	64	True
Product	100	64	True
Recommend	100	220	False
Shipping	100	64	True

Table 4.2: Characteristics of Service Components of Hipster Shop Application

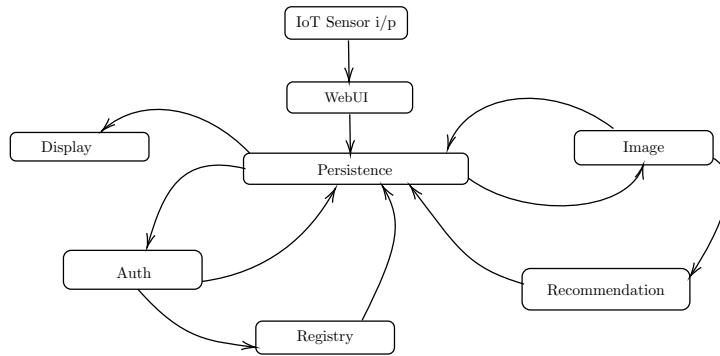


Figure 4.7: Interactions between service components in Tea Store Application

Service	CPU	Mem	Mandatory
webui	200	80	False
registry	100	150	True
recommender	100	80	True
persistence	200	100	False
image	100	150	False
auth	100	80	True

Table 4.3: Characteristics of Service Components of Tea Store Application

Tea Store is an example for reference application which belongs to a different category in the e-commerce industry. Tea store is a well known benchmarking application used for evaluating various metrics by different researchers in the e-commerce industry (Eismann et al. 2019). The different services in the Tea Store posses unique characteristics. Figure 4.7 represents the architecture of the Tea Store and Table 4.3 gives the characteristics of the Teastore application. The CPU resource is expressed in millicores and memory resource is expressed in mebibytes.

4.4.1.2 Healthcare Application

In the current era, the healthcare sector leverages various IoT devices and sensor devices for monitoring and collecting patient related information. Critical health related information must be processed without much delays and cater to the safety requirements while providing quality medical decisions regarding the patient health. Fog computing enables these devices to perform critical analytics locally rather than depending on the far away cloud servers.

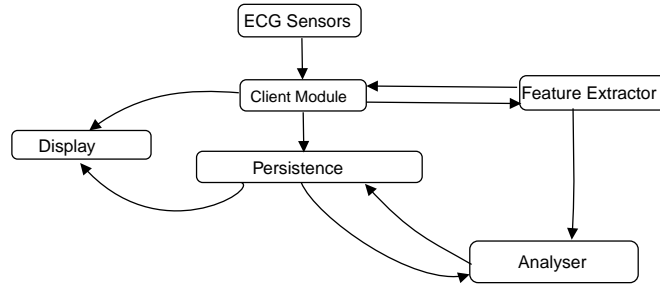


Figure 4.8: Interactions between service components in Smart-Health Application

Service	CPU	Mem	Mandatory
Client	200	100	True
persistance	150	200	False
feature extractor	200	80	True
analyser	200	100	False

Table 4.4: Characteristics of Service Components of Smart-Health Application

We considered a microservice based smart healthcare application (Pallewatta et al. 2019) for evaluating the performance of the proposed approach. The application components and their interactions are depicted in Figure 4.8. The Smart-Health application consists of a wearable Electrocardiogram (ECG) monitoring sensor which monitors the ECG of the patient continuously. The feature extractor module detects the anomalies and gives notifications immediately. The ECG analyser module performs the complex analysis operations and the persistence module takes the necessary actions for long term storage of data. The details regarding memory, cpu resource requirements and the relevance of each service are given in Table 4.4. The CPU resource is expressed in millicores and memory resource is expressed in mebibytes. The various parameters and their values used for characterisation of experiments are given in Table 4.5 .

4.4.2 Performance Metrics

The efficiency of the proposed CREW approach is measured and compared with NSGA-II based Fog Service Placement Problem (FSPP) and Multi-Objective Whale Optimisation approach (MOWOA).

- Cost : The cost gives the the total monetary cost of execution of applications. The

Element	Parameter	Units	Value
Cloud	Failure Prob		0.003
	CPU- Cost	\$(per unit time)	4/core
	Memory-Cost	\$(per unit time)	4/GB
FCN	Failure Prob		0.005
	CPU-Cost	\$ (per unit time)	4/core
	Memory-Cost	\$(per unit time)	5/GB
	Config1	# devices	2
	Config2	# devices	4
	Config3	# devices	6
FC	Failure Prob		0.00002
	CPU- Cost	\$(per unit time)	5/core
	Memory-Cost	\$(per unit time)	6/GB
	Config1	# devices	4
	Config2	# devices	8
	Config3	# devices	12

Table 4.5: Parameter values for different characteristics of Cloud and Fog devices

cost is calculated based on the Equation 4.11. The unit costs for the execution environment considered in our experiments is given in Table 4.5 (Brogi A 2019).

- **Reliability** : We consider a reliable execution of the application to ensure that atleast all the mandatory components of the application are up and running for a period of time. The reliability is estimated based on the failure probability of the services based on the Equation 4.10.
- **Response Time**: The response time of the service is estimated based on Equation 4.6 and compared with the provided delay constraints. Placement schemes should make sure that the application will be able to provide responses to service requests within the permissible delay.

4.5 EXPERIMENTAL RESULTS AND ANALYSIS

In this section, the proposed CREW approach is evaluated with the baseline placement approaches.

4.5.1 Performance comparison of various Placement Schemes

To measure the performance impact of the proposed CREW algorithm, the following baseline approaches are considered.

- NSGA-II based Service Placement:

Nondominated Sorting Genetic Algorithm is a multi-objective meta-heuristic algorithm which solves a multidimensional function and approximates the Pareto Front and Pareto set (Deb et al. 2002). The Service Placement scheme in CREW is replaced to use NSGA-II instead of eagle strategy based Multi-objective Whale Optimisation. The NSGA-II determines the mapping between the services and available devices in the Fog environment.

- MOWOA based Service Placement:

Whale optimisation algorithm is a recent meta-heuristic optimisation algorithm based on the characteristics of Hump back whales, which promises faster convergence in lesser number of iterations (Mirjalili and Lewis 2016). Multi-Objective Whale Optimisation Algorithm (MOWOA) reaps all the advantages of whale algorithm and also provides fast convergence of multi-objective functions to the true Pareto fronts. This service placement scheme varies from CREW in the meta-heuristic approach used.

- FFD-latency based Service Placement:

This service placement policy is based on the First Fit Decreasing approach (Natesha and Guddeti 2018). The approach mainly focusses on reducing application latency and efficient utilisation of the system resources.

The suitability of eagle strategy based Multi-Objective Whale algorithm in the CREW approach is analysed by comparing with the Nondominated Sorting based Genetic Algorithm (NSGA-II) and the Multi-Objective Whale Optimisation Algorithm (MOWOA). The *HyperVolume* (HV) metric is a performance measure that reflects the area of region in the search space dominated by the obtained set of solutions (Tan et al. 2018). It is highly desirable to have larger values for the hypervolume indicator. The mean HV

# of nodes	CREW	NSGA-II	MOWOA
18	0.8178	0.7366	0.5135
20	0.79525	0.7944	0.6539
30	0.8396	0.7754	0.5667
40	0.7726	0.7724	0.6304
50	0.7253	0.6487	0.5516
60	0.7362	0.6297	0.5895

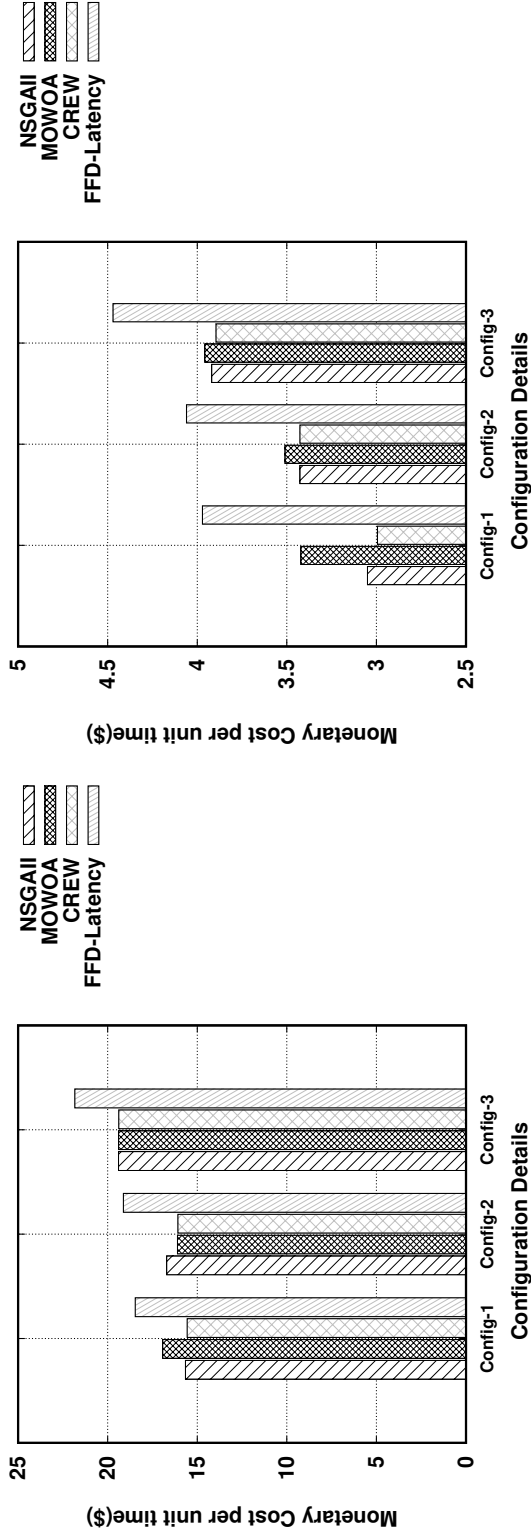
Table 4.6: Mean Hypervolume Values over 30 independent runs

values for all the three methods are given in Table 4.6 and it is observed that CREW performs better than the NSGA-II and MOWOA based approaches in all the different configurations.

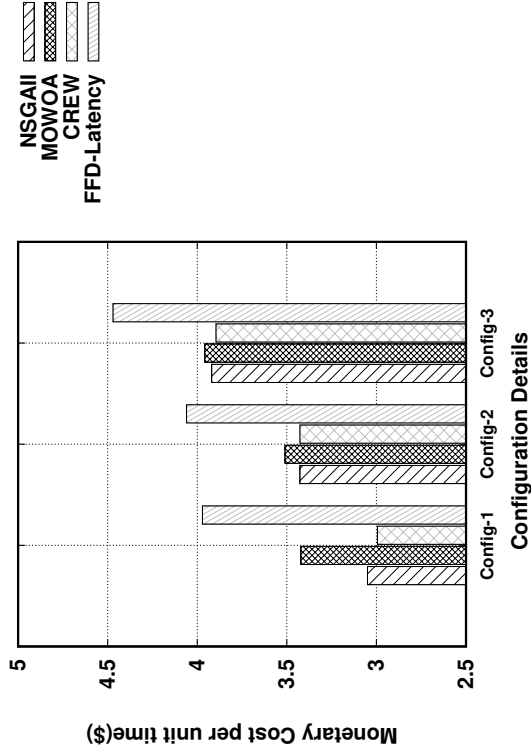
4.5.1.1 Execution Cost

The cost of execution gives the monetary cost required to run the application in the Fog-Cloud environment within the deadline. The service modules are distributed both in Cloud and Fog nodes based on the tolerable delay with the objectives of minimising the cost and maximizing the reliability. The execution cost of an application depends on the execution time required to run each service of an application and the device on which the service is running (Brogi A 2019).

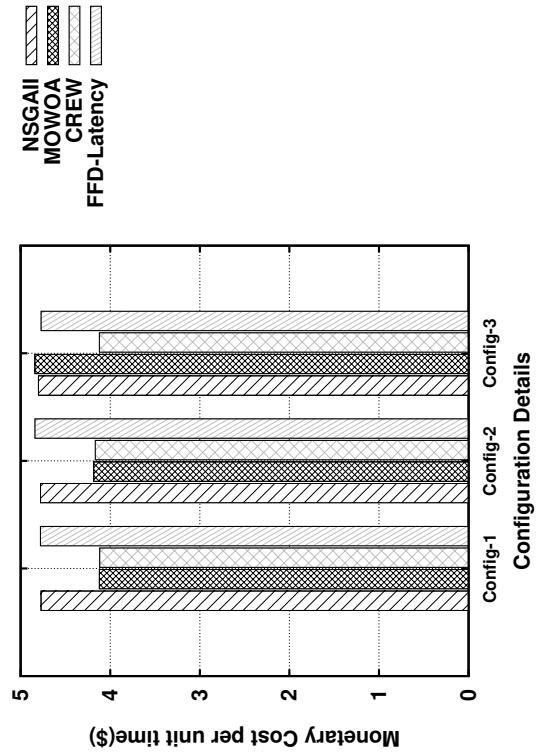
Even though the Fog devices are located near to the end devices, their sophisticated nature and capacity limitations add to the monetary cost to run the applications. We have evaluated the performance of the proposed CREW approach with two different applications having different characteristics (as presented in Section 4.4.1.1). Figures 4.9a, 4.9b and 4.9c show the results of the different meta-heuristic based placement policies.



(a) Hipster Shop Application



(b) Tea Store Application



(c) Smart-Health Application

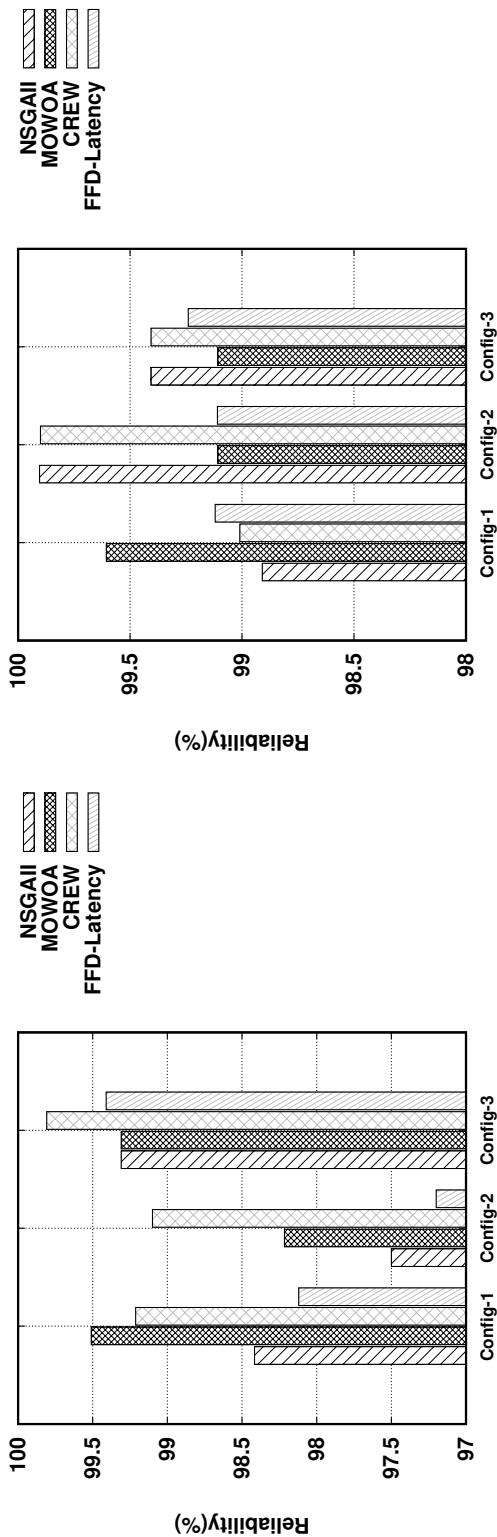
Figure 4.9: Comparison of Monetary Costs of Execution with varying placement schemes

The proposed CREW algorithm finds the mapping with lesser execution costs for all the three different applications. We have also evaluated our approach with different topologies varying in the number of *FCN* and *FC* nodes. The details regarding the different configurations considered are given in Table 4.5. From Figures 4.9a and 4.9b for the TeaStore and Hipster application, it is observed that placing services using the proposed CREW strategy results in an average reduction of 5% and 3% respectively, in the execution cost, when compared to the MOWOA. For highly delay-sensitive IoT applications, if more number of Fog nodes are available, our algorithm will place more services near the end devices in order to ensure timely service response delivery.

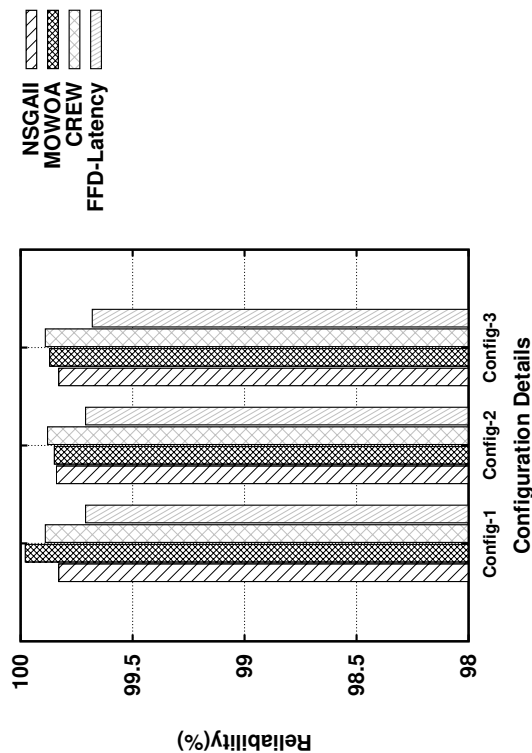
4.5.1.2 Reliability

The reliability of a system offered to the deployed application is its ability to ensure that atleast the mandatory microservices run without failures for the considered period of time.

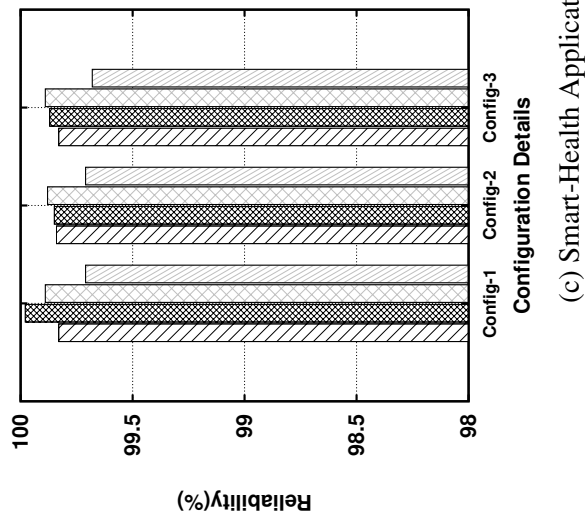
The minimisation of cost and maximisation of reliability are conflicting objectives. The reliability offered to the two different e-commerce applications (discussed in Section 4.4.1.1), were evaluated based on the reliability model presented in Section 4.1.5. The reliability of the application is calculated based on the failure probability of the services of the application as given in Equation 4.10. The values of the failure probabilities of various devices (Birke et al. 2014) are given in Table 4.5.



(a) Hipster Shop Application



(b) Tea Store Application



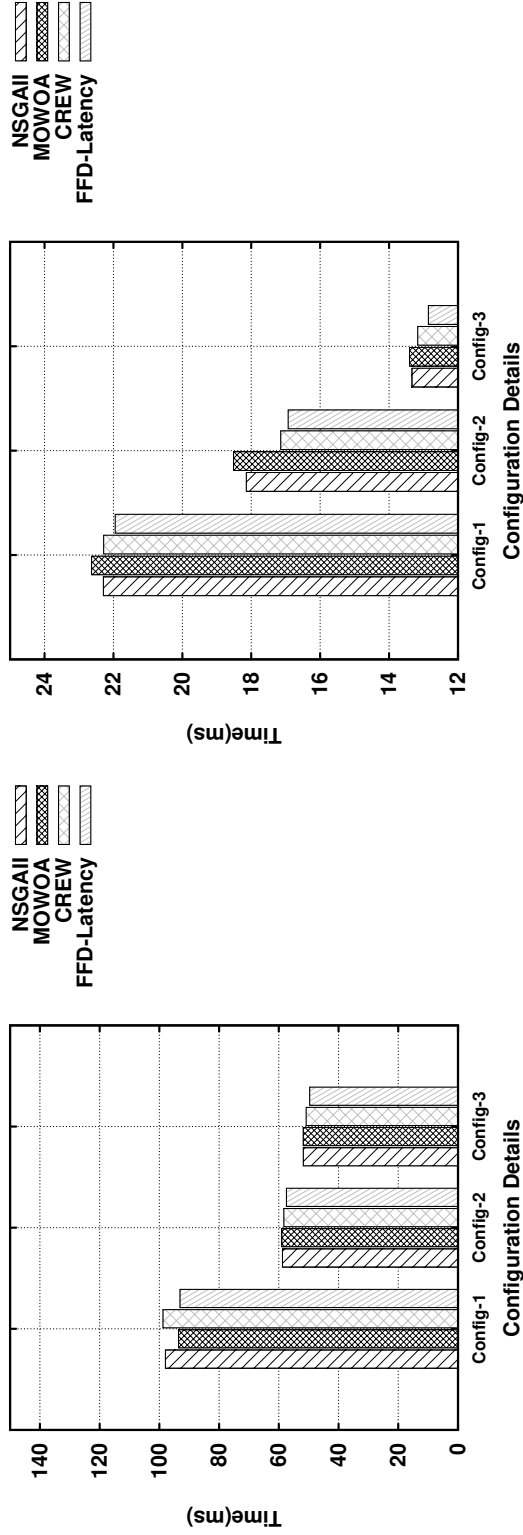
(c) Smart-Health Application

Figure 4.10: Comparison of Reliability values with varying placement schemes

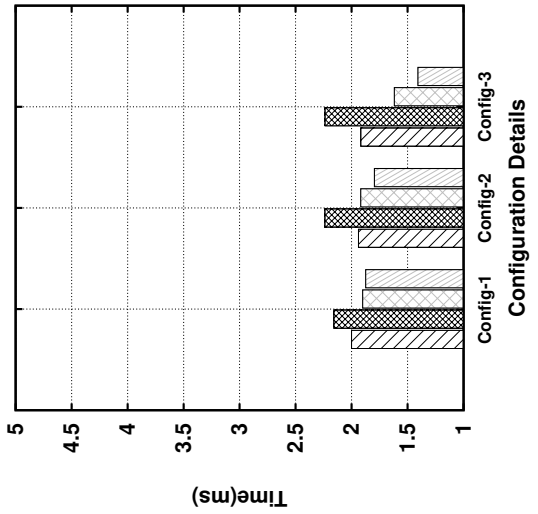
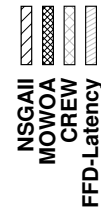
The application service instance failure rates ($q^{k,m}$) can be estimated from the Cloud application traces. For simplicity, but without loss of generality we have taken the values as 0.003, 0.0025 and 0.002 for the Hipster Shop, Teastore and Smart-Health applications respectively (Birke et al. 2014). From Figures 4.10a, 4.10b and 4.10c, it is observed that the MOWOA gives high reliability in scenarios with less number of Fog devices. It is however observed that the placement strategy generated by MOWOA for Config-1 with an increased reliability is inferior with respect to the execution costs incurred. The proposed approach CREW determines a superior placement strategy whose cost cannot be reduced further without affecting the reliability. As the number of Fog devices increases, the limitations in the exploration capabilities of the MOWOA restricts the placement strategy to generate local optimal solutions rather than the global optimum.

4.5.1.3 Response Time

Response time gives the total time taken from the instant at which the user submits the request till the time user receives the response. We measured response time in milliseconds. Each application is assigned with a tolerable delay within which the user expects a response. Our proposed approach ensures the delivery of the services within the tolerable delay. The response time of each application is calculated based on the sum of execution time, waiting time and communication time using the Equation 4.6. Figures 4.11a , 4.11b and 4.11c give the response times of Hipster Shop, Tea Store and Smart-Health. Even though the proposed approach CREW gives a higher response time for the Configuration 1 of Hipster Shop application compared to the other approaches, it ensures timely service response. When the number of Fog devices are increased, there is a drastic reduction in the response time of both applications due to the increased availability of the Fog devices for placement of services. The response time of Hipster Shop is higher in all the configurations compared to that of Tea Store and Smart-Health applications because of the higher number of service modules in the application.



(b) Tea Store Application



(c) Smart-Health Application

Figure 4.11: Comparison of Response Time values with varying placement schemes

Model	Parameters	Value
SVR	C	1000
	gamma	0.1
	kernel	rbf
Random forest	bootstarp	TRUE
	criterion	mse
	max_depth	None
	max_features	auto
	max_leaf_nodes	None
	min_impurity_decrease	0
	min_impurity_split	None
	min_samples_leaf	1
	min_samples_split	2
	min_weight_fraction_leaf	0
	n_estimators	10
	n_jobs	None
	oob_score	FALSE
random_state	None	
verbose	0	
warm_start	FALSE	
XGBOOST	colsample_bytree	0.9
	eta	0.05
	max_depth	9
	num_boost_round	100
	subsample	1

Table 4.7: Parameter settings for ML-based prediction techniques

4.5.2 Performance Comparison of various Machine Learning Techniques for execution time prediction

Three regression models using XGBoost, Random Forest and Support Vector Machine regressors (SVR) were built and evaluated for the prediction of execution time. The hardware characteristics of the Fog computing environment and software characteristics of the applications are inputs to the regressors. We have fine tuned the parameters by repetitively conducting training experiments. The parameter values used for the experimentation is given in Table 4.7.

The accuracy of the predicted values is calculated based on the difference between the actual values and predicted values. The accuracy of the different models were evaluated based on metrics such as the Root Mean Square Error (RMSE), Mean Absolute Error (MAE) and R-squared Error. The obtained values are given in Table 4.8. It is highly desirable to achieve lower values for RMSE, MAE and higher values for R-squared Error. Based on the conducted investigations, the XGBOOST model is best suited for our system. The Random Forest regressor takes more time for the learning process, thus making it unsuitable for real time predictions.

Metric	SVR	Random Forest	XGBOOST
RMSE	0.0005113	0.0007326	0.0007024
MAE	4.08127E-05	5.54E-05	3.73E-05
R2 Error	0.009334	0.030903	0.1092956

Table 4.8: Comparison of prediction accuracy values for varying ML-based service execution time prediction techniques

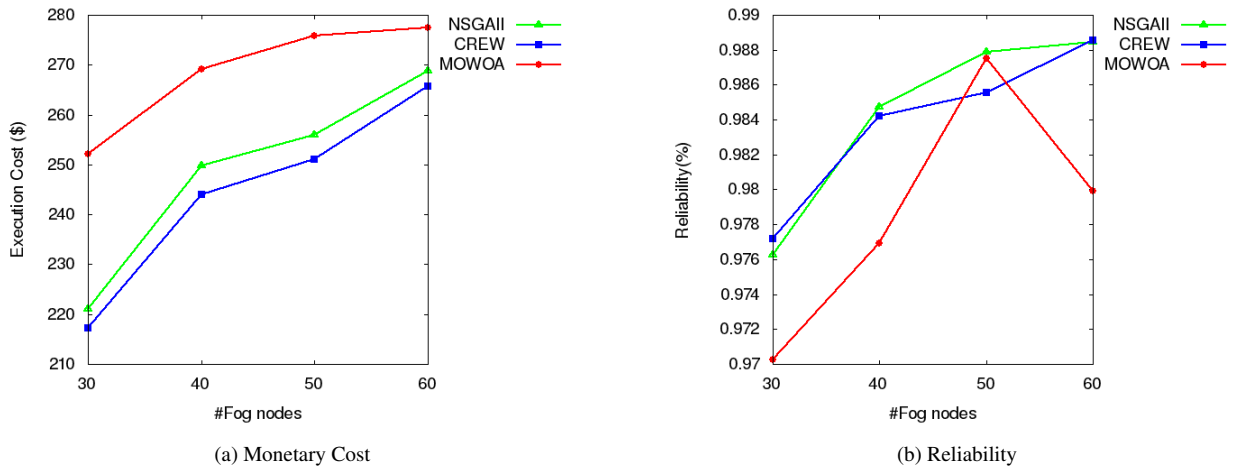


Figure 4.12: Performance analysis in large search space with varying placement schemes

4.5.3 Performance Analysis in Large Search Spaces

To evaluate the performance of the Fog service placement algorithms in scenarios with higher number of devices and services, we considered a simulation scenario in iFogSim with 100 services with varying number of Fog nodes. Figure 4.12a shows the Monetary cost of execution of services of the different placement policies. The analysis shows that the performance of NSGA-II and MOWOA deteriorates with the increase in problem size. The proposed CREW based placement maintains the performance even with increased problem sizes. The reliability values obtained using different placement strategies are given in Figure 4.12b. With expansion in the problem sizes, MOWOA gives an unstable performance which is primarily due to the inefficient exploration capabilities of the MOWOA. The proposed CREW based placement scheme takes better placement decisions compared to all the other placement schemes, even for large number of Fog nodes and Fog services.

4.6 SUMMARY

The proposed Fog Service Placement Scheme, CREW captures the two conflicting objectives of minimising the cost and maximising the reliability while taking decisions regarding the placement. In CREW, the FSPP was formulated as a multi-objective optimisation problem with these conflicting objectives and solved using an eagle strategy based Multi-objective Whale Optimisation Algorithm. The framework includes a monitor which monitors the resource requirements, usages and available capacities. The Monitor forwards the required information to the subsequent components. The Execution time predictor predicts the execution time of each incoming services based on their software and hardware profiles. The waiting time estimator estimates the waiting time of each services at different levels in the hierarchical Fog environment. The response time estimator calculates the overall response time of application based on the execution time, waiting time and communication time. Reliability and cost calculators calculate the execution costs and reliabilities. Eagle Strategy based MOWOA finds the placement mapping based on the execution cost, reliability and response time. Extensive experiments are conducted on the Fog testbed and simulation environments to validate the practical feasibility of the proposed approach. A detailed comparison with popular multi-objective algorithms such as NSGA-II and MOWOA establishes the superior performance of the proposed approach.

CHAPTER 5

MOBILITY AWARE AUTONOMIC APPROACH FOR THE MIGRATION OF APPLICATION MODULES IN FOG COMPUTING ENVIRONMENT

The user devices sense the environment data and generate streams of data to be processed by the application modules. Each request for processing is received by the Fog node to determine the most apt course of action. The user devices submitting these requirements may not be stationary. This implies that the access points that the devices use to communicate with the Fog nodes, may change. This results in an increase in the hop counts, which adversely impacts the deadline constraints and delay requirements.

To guarantee prompt service delivery, the processing must be transferred to a Fog node which is nearer to the user access point. This process involves transferring of data and the application context encased in the containers. In container-based systems, this can be realized by migrating the containers corresponding to the user, to a Fog node situated closer to the end device.

Researchers have propounded different solutions for providing mobility support in distributed environments. Bi et al. (2018) proposed a Fog computing architecture to support mobility. Their architecture decouples mobility control and data forwarding using Software Defined Networks (SDN). Islam et al. (2016) proposed a VM migration model for the Mobile Cloud Computing (MCC) environment. Their approach consid-

ered not only the user mobility but also the load in the cloudlet. Machen et al. (2018) developed a framework to support user mobility in Mobile Edge Computing (MEC) environments. In order to provide the service without interruption for mobile users, they have used migration of services across MECs. Bittencourt et al. (2017) emphasise the need for mobility aware scheduling in Fog computing environments. They accentuated the important metrics to be considered by a Fog scheduler while taking decisions in a mobility supported environment.

This chapter mainly addresses the problem of migrating the containers corresponding to the user application modules, across the Fog nodes. The migration of containers is done in an autonomic manner, by adopting the autonomic control loop. Nevertheless, naively migrating containers along the path of the mobile user may lead to unwanted migration actions. This causes an increase in the system overhead, resulting from the high resource and network bandwidth requirements of each migration process. Thus, migration may be opted only in situations where utmost necessary. In the event of no viable options to transfer the processing, the application may be offloaded to the Cloud to ensure uninterrupted service delivery.

The remainder of this chapter is organized as follows: Section 5.1 presents the motivation behind the research, Section 5.2 presents the model formulated for the elements of the Fog environment, Section 5.3 describes the developed framework, Section 5.4 describes the experiments conducted, Sections 5.5 and 5.6 present the details of the experimental results and Section 5.7 concludes the chapter.

5.1 MOTIVATION

There is a dearth of research that considers the mobility support feature in Fog environments. The research community has not considered the migration of containers running the applications to support the mobility of users. There is a need for research approaches for efficient migration of containers in Fog environments. Though a few approaches discuss the migration of VMs, the same techniques are not directly applicable in the context of container virtualization ¹. Designing approaches for migration

¹<https://www.virtuozzo.com/connect/details/blog/view/live-migration-in-virtuozzo-7.html>

include challenges in identifying the situations when migrations are required and also identifying the subset of containers and Fog nodes to be considered for the migration process.

5.2 SYSTEM MODEL

This section details the formulated model of the different elements of Fog environments. The decentralized deployment model of Fog varies according to the scenario in which it is posited. In this work, a hierarchical structure is considered. The data from the IoT devices are received by the Fog nodes, rather than transporting the bulk volume of data to the Cloud. The Fog nodes receiving this data, process the data and take decisions which are communicated to the edge devices. The data required for future analysis and those which cannot be processed by Fog layer is transported to the Cloud. The IoT devices are considered to exhibit mobility. When the users or mobile devices change their location with respect to time, data and processing related information must also be transferred in a timely manner, to avoid intermittent delays or interruptions in the service. A hybrid approach is proposed in this chapter, for the migration of application modules in the Fog environment based on autonomic computing and genetic algorithm.

We have considered lightweight virtualization technology, which can be leveraged in the form of containers for deploying the application modules and user data. Containers are placed on the Fog nodes and the mobile users are connected to these nodes for accessing the services. To provide lower latency values and better Quality of Experience (QoE), containers must be migrated to the Fog nodes which are closer to the current position of the user. Determining the instant at which migration actions must be initiated and identifying a suitable location for the module in action are the two major issues in the context of migration in Fog environments.

The system under consideration has been portrayed in Figure 5.1. User devices connected to the Fog form a layer represented as the ‘Sensor Layer’. The Fog layer basically acts as the arbitrator between the Sensor and Cloud layers. The Fog layer consists of different Fog nodes, that provide Fog services. Fog nodes may be gateways, routers, switches or dedicated physical servers. The data collected by the sensors in

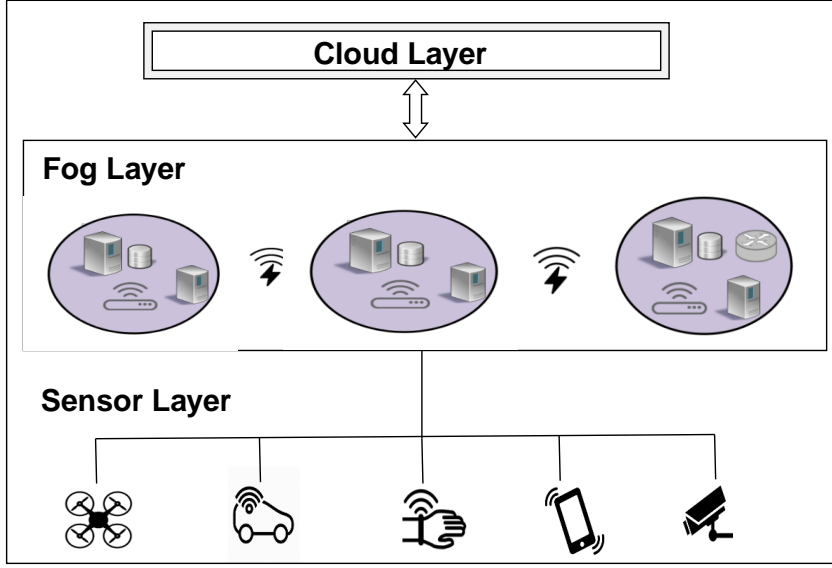


Figure 5.1: Problem Scenario

the Sensor layer is sent to the Fog nodes. We consider a scenario in which the Fog environment is used to provide low latency services for users whose locations vary with respect to time. The Fog environment considered in this work, consists of a set of Fog nodes $F = \{fn_1, fn_2, \dots, fn_N\}$, where the number of Fog nodes in the environment is denoted as N . The characteristics of a Fog node fn_j can be defined as a tuple $\langle fn(cpu)_j^{cap}, fn(mem)_j^{cap}, fn(bw)_j^{cap} \rangle$ where $fn(cpu)_j^{cap}$ represents the number of CPU cores available on the j^{th} Fog node and $fn(mem)_j^{cap}$ is the amount of memory available and $fn(bw)_j^{cap}$ is the bandwidth available, respectively. The mobile users access the services from the application modules which are deployed as containers on various Fog nodes or in the Cloud. An application module i can be denoted as app_i . There may exist a set of M modules running in the Fog environment and the set of all application modules is represented as AM . Each app_i is allocated to a suitable Fog node which satisfies the resource requirements of the corresponding module. The requirements of the i^{th} module can be denoted by a tuple $\langle app(cpu)_i^{req}, app(mem)_i^{req}, app(bw)_i^{req} \rangle$, where $app(cpu)_i^{req}$ is the number of cores of CPU requested by the module and $app(mem)_i^{req}$, $app(bw)_i^{req}$ are the amount of memory and bandwidth resources requested by the i^{th} module, respectively. The elements in the Fog system and their representations have been summarized in Table 5.1.

Notation	Element	Description
F	Fog Node	Set of all available Fog Nodes in the Fog environment
N		# of Fog Nodes available in the system
f^{n_j}		j^{th} Fog Node, $\forall_{j=1}^N f^{n_j} \in F$
$fn(cpu)_j^{cap}$		# of CPU cores available on the Fog Node, f^{n_j}
$fn(mem)_j^{cap}$		Amount of memory available on the Fog Node, f^{n_j}
$fn(bw)_j^{cap}$		Bandwidth available on the Fog Node, f^{n_j}
AM	Application Module	Set of all application modules to be migrated in the Fog environment
M		# of migrating Application Modules in the System
app_i		i^{th} application module $\forall_{i=1}^M app_i \in AM$
$app(cpu)_i^{req}$		# of CPU cores requested by i^{th} application module
$app(mem)_i^{req}$		Amount of memory requested by i^{th} application module
$app(bw)_i^{req}$		Amount of bandwidth requested by i^{th} application module
δ_i		Tolerable delay of i^{th} application module
TTC_i		Time to Completion of i^{th} application module
$\Delta_{app_{i,j}}$		Processing time of i^{th} application module on j^{th} Fog node
$MT_{app_{i,j}}$		Migration time for i^{th} application module to migrate to j^{th} Fog node
$distance_{i,j}$		Distance between current position of user submitting request for application module i and the j^{th} Fog node in which app_i is deployed
D_{kj}		Distance between the current Fog node k and target candidate Fog node j .

Table 5.1: Notations used in the system model

5.2.1 Optimization Model

The migration of application modules to destination Fog nodes from the currently allocated set of Fog nodes involves 3 steps:

1. Identifying when a migration action is to be initiated
2. Identifying which application modules are to be migrated
3. Determining destination nodes for each of the migrating modules

The first and second steps are considered in the subsequent sections. The third step of determining destination Fog nodes is considered as a sub-class of re-allocation problems. The re-allocation problem can be formulated as an optimization problem as discussed in this section. The system comprises of a number of application modules, M to be migrated and a set of Fog nodes F as possible destinations. The mapping of application modules to these Fog nodes can be formulated as a (0/1) Integer Linear Programming (ILP) Problem. The objective of this problem is to minimize the overall Time To Completion (TTC) of the migrating modules. Time to Completion is defined as the time taken by a migrating application module to complete the processing of an incoming request at the destination Fog node. The TTC_i of the i^{th} migrating module includes two components. The first component is the time required to migrate the application module ($MT_{app_{ij}}$) from the current Fog node k to the designated destination Fog node j (at a distance D_{kj} from the current user location). The migration time denotes the time taken to transfer the associated memory contents across the available network bandwidth. This time is calculated as given in Equation 5.1. The second component $\Delta_{app_{ij}}$ evaluates the time taken to process a request submitted to the application module at the destination node. In other words $\Delta_{app_{ij}}$ is the time required to process/service a request by the module ‘i’ on the Fog node ‘j’.

$$\begin{aligned}
TTC_i &= (MT_{app_{ij}} * D_{kj}) + \Delta_{app_{ij}} \\
MT_{app_{ij}} &= \text{Migration time for } app_i \text{ to migrate to } fn_j \\
&= \frac{\text{Memory size required by } app_i}{\text{bandwidth of } fn_j}
\end{aligned} \tag{5.1}$$

The decision variable of the problem denotes the Fog node to which the application module app_i can migrate. The binary decision variable can be defined as in Equation 5.2.

$$x_{ij} = \begin{cases} 1, & \text{if application module 'i' can be migrated to Fog node 'j'} \\ 0, & \text{otherwise} \end{cases} \tag{5.2}$$

$$\text{Minimize } \sum_{i=0}^M \sum_{j=0}^N x_{ij} TTC_i$$

subject to,

$$\forall fn_j \in F, \sum_{i=1}^N x_{ij} app(cpu)_i^{req} \leq fn(cpu)_j^{cap} \tag{5.3}$$

$$\forall fn_j \in F, \sum_{i=1}^N x_{ij} app(mem)_i^{req} \leq fn(mem)_j^{cap} \tag{5.4}$$

$$\forall fn_j \in F, \sum_{i=1}^N x_{ij} app(bw)_i^{req} \leq fn(bw)_j^{cap} \tag{5.5}$$

$$\forall app_i \in AM, \forall fn_j \in F, x_{ij} distance_{ij} < 600 \tag{5.6}$$

$$\forall app_i \in AM, \forall fn_j \in F, x_{ij} TTC_i \leq \delta_i \tag{5.7}$$

$$\forall app_i \in AM, \sum_{j=0}^N x_{ij} = 1 \tag{5.8}$$

The optimization problem tries to reduce the time to completion of migrating modules and thus reduces the latency experienced by users for processing the requests. Equa-

5. Mobility aware autonomic approach for the migration of application modules in Fog Computing Environment

tions 5.3, 5.4 and 5.5 represent the constraints on processing, memory and bandwidth resources. The sum of the resource requests by all the application modules on a Fog node should not exceed the total resource capacity of the Fog node. Equation 5.6 attempts to reduce unwanted migrations by permitting a migration only if it falls completely within the range of the target Fog node. Equation 5.7 ensures that the delay constraints are not affected due to the migration process. Equation 5.8 makes sure that every application module is placed on one and only one Fog node.

5.2.2 Model Example

Consider a Fog computing environment with two Fog nodes and three mobile devices which needs to be migrated. The location co-ordinates of the Fog nodes are as follows: f_{n_1} is at $\langle 100, 100 \rangle$ and f_{n_2} is at $\langle 1500, 100 \rangle$. The resource capacities of each of the Fog nodes are $\langle 3, 8, 10000 \rangle$ and $\langle 4, 8, 10000 \rangle$ respectively. The current locations of the mobile users are marked by the co-ordinates $\langle 100, 600 \rangle$, $\langle 900, 100 \rangle$ and $\langle 177, 400 \rangle$ and resource requirements of their corresponding containers are $\langle 1, 2, 800 \rangle$, $\langle 2, 5, 600 \rangle$ and $\langle 2, 3, 400 \rangle$. Identification of the suitable destination for migration can be formulated as a (0/1) Integer Programming Problem. The objective function formed is given in Equation 5.9.

$$\begin{aligned} \text{Minimize } & 1.33x_{11} + 3.23x_{12} + 4.67x_{21} + \\ & 3.5x_{22} + 5.55x_{31} + 1.58x_{32} \end{aligned} \tag{5.9}$$

subject to,

$$x_{11} + 2x_{21} + 2x_{31} \leq 3$$

$$x_{12} + 2x_{22} + 2x_{32} \leq 4$$

$$2x_{11} + 5x_{21} + 3x_{31} \leq 8$$

$$2x_{12} + 5x_{22} + 3x_{32} \leq 8$$

$$800x_{11} + 600x_{21} + 400x_{31} \leq 1000$$

$$800x_{12} + 600x_{22} + 400x_{32} \leq 8$$

$$x_{11} + x_{12} = 1$$

$$x_{21} + x_{22} = 1$$

$$x_{31} + x_{32} = 1$$

$$x_{11}distance_{11} < 600, x_{12}distance_{12} < 600,$$

$$x_{21}distance_{21} < 600, x_{22}distance_{22} < 600,$$

$$x_{31}distance_{31} < 600, x_{32}distance_{32} < 600$$

$$\delta_1 < 2.5, \delta_2 < 4.1, \delta_3 < 3.2$$

The optimization problem can be solved using any integer linear programming solvers. The solution using IBM CPLEX engine is given in Table 5.2. IBM CPLEX is considered as the top performing open source solver in terms of speed and capability (Gearhart et al. 2013). Classical optimization techniques provide accurate results for small problem spaces. However, with increase in the number of Fog nodes and applica-

Parameter	Optimization Value
Objective value	6.41
x_{11}	1
x_{12}	0
x_{21}	0
x_{22}	1
x_{31}	0
x_{32}	1

Table 5.2: Mathematical Model Solution

tion modules, this may not be feasible. The subsequent sections discuss meta-heuristic techniques that can be used to obtain near-optimal solutions in such cases.

5.3 PROPOSED CONCEPTUAL FRAMEWORK

The proposed approach adopts autonomic computing paradigm for the effective orchestration of the Fog computing environment. Fog computing environments face challenges to provide support for applications demanding mobility support. The mobile nature of the users and the heterogeneous nature of the resources available, raises the need for migration of application modules from one Fog device to the other. Migration of application modules minimizes the latency experienced by the users in motion thereby ensuring that the delay requirements of the applications are met. Autonomic systems can easily adopt themselves to fluctuations in the environment and this feature serves as a promising concept for the management of various distributed infrastructures. The proposed approach uses the MAPE (Jacob et al. 2004)(Ghobaei-Arani et al. 2016) autonomic control loop for the management of migrating application modules and their mapping to Fog nodes in the Fog environment. In the MAPE loop, M represents Monitor, A stands for Analyser, P stands for Planner and E is for Executor.

A framework called **MAMF**-Mobility aware Autonomic Migration Framework, is proposed, for the autonomic orchestration of mobility aware migration of application modules in the Fog computing environment. As depicted in Figure 5.2, the MAMF considers the three layers namely the Cloud layer, Fog layer and sensor layer. Sensor layer consists of end devices which may or may not be in motion. The Sensor layer generates the requests to be processed. It must be noted that the devices in this layer generally do not have any kind of processing abilities. These requests are forwarded to those nodes in the higher layers, which possess sufficient infrastructure and computing resources to process the data. The nodes may be either from the Fog computing layer or the Cloud computing layer. The Fog layer falls between the Sensor layer and the Cloud layer. The Fog layer receives the processing requests from the Sensor layer. The first phase of the processing of requests is carried out at the Fog node, rather than merely relaying the requests as received, to the Cloud. The Fog layer processes majority of

the requests and sends the results back to the Sensor layer. The requests which are delay tolerant and cannot be processed at the Fog layer due to lack of resources, are forwarded to the Cloud layer. In the proposed MAMF, the autonomic algorithm based on the MAPE autonomic loop runs in the Fog layer.

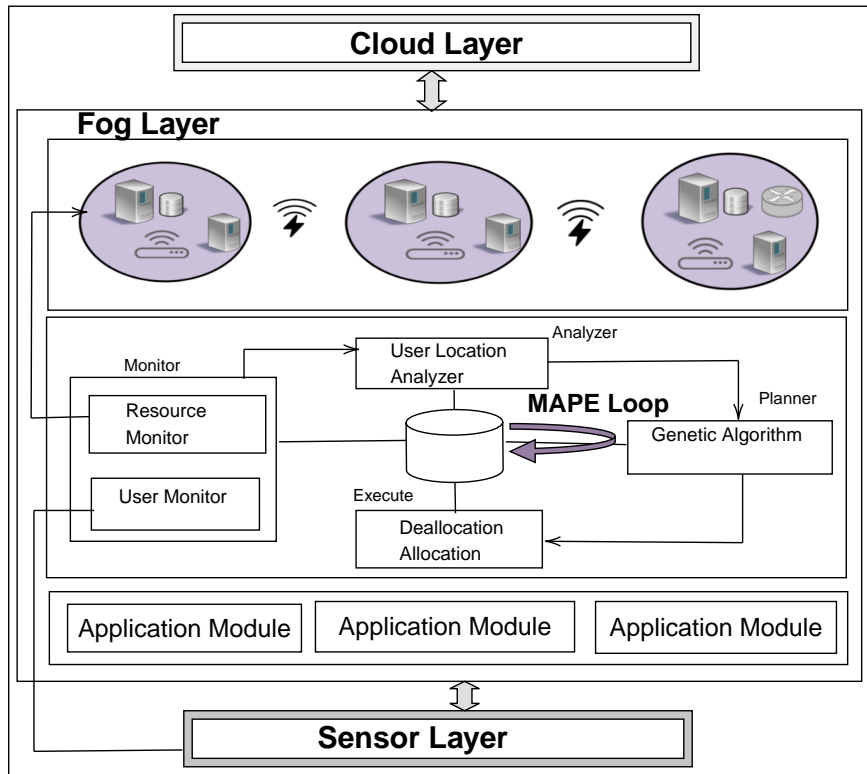


Figure 5.2: MAMF framework based on control MAPE loop

A model for smart vehicle system was considered, which uses Fog computing environment for hosting services. Each vehicle poses as a client availing services from the Fog nodes and Fog nodes providing the services are deployed across the roads. The communication protocol used for communication between the mobile clients and the Fog nodes is IEEE802.11p. The protocol is an enhanced version of the IEEE802.11, permitting wireless access in vehicular environments (Gozálvez et al. 2012). This protocol supports communication for transmission frequencies in the band range of 5.85GHZ to 5.9GHZ. Each Fog node/server controls and co-ordinates the mobile users located within its coverage area. The actual coverage area for each Fog server depends on several factors such as geographical conditions, propagation conditions and terrain

5. Mobility aware autonomic approach for the migration of application modules in Fog Computing Environment

types. The Friis transmission Equation (Friis 1946) may be applied to calculate the power received by the mobile user and for calculating the coverage area of each Fog server. The Friis equation is provided in Equation 5.10.

$$P_r = \frac{P_t G_t G_r \lambda^2}{(4\pi R)^2} \quad (5.10)$$

where

P_r = Power at receiver

P_t =Output power of transmitting antenna

G_t =Gain of transmitting antenna

G_r =Gain of receiving antenna

λ =Wavelength

R =Distance between antennas

The use of an omnidirectional antenna is assumed, implying that $G_t = 1$ and the antenna is assumed to have 100% aperture efficiency, $G_r = 1$. The value of P_t is taken as $1w$.

Considering the minimum power requirements and transmission range of IEEE802.11p, the lower and upper thresholds for migration was fixed at $-70dbm$ and $-75dbm$ which corresponds to $600m$ and $1000m$ respectively.

5.3.1 MAPE Control Loop

MAMF follows the MAPE control loop concept and provides mobility support through the efficient orchestration of application modules. The MAPE loop running in the Fog layer consists of *four* phases, namely Monitor, Analyze, Plan and Execute. The operations in each phase is discussed in this section. The overview of the MAPE loop is excerpted in Algorithm 5.1. Initially, the system boots the required number of Fog nodes. For each user request arriving, the MAPE control loop is executed at specified time intervals Δt . The loop includes a check to determine whether migration of application modules must be initiated or not. If migrations are to be initiated, the MAMF

proceeds to identify suitable destination Fog nodes for migrating application modules.

Algorithm 5.1: MAMF-Mobility aware Autonomic Migration Framework algorithm

```

Initialization: Boot required number of Fog nodes
1 Begin
2 while user requests arrive for application modules deployed on Fog nodes do
3   for every time interval  $\Delta t$  do
4     Monitor
5     Analyze
6     Plan
7     Execute
8   end
9 end
10 End

```

5.3.1.1 Monitoring Phase

In an autonomic environment, the monitoring phase is responsible for sensing the managed process and its operation context (refer Algorithm 5.2). In this phase, the location history of the users submitting requests for the application module app_i is monitored by the user monitor (line 1) and the status of resource is monitored by the resource monitor (line 2). The knowledge base collects the information from the monitoring base and stores it for use by the subsequent phases.

Algorithm 5.2: Pseudo code for Monitoring Phase

```

1 Begin
  /* Monitor the time-based location history of the user
  and store the positions of the mobile user in the
  Knowledge base */
2 Monitor  $\{X_1, X_2, \dots, X_t\}$ 
  /* Monitor CPU, Memory, Bandwidth Utilization */
3 Monitor (Resource utilization values)
4 End

```

5.3.1.2 Analyse Phase

The Analyse phase is devoted to process the data collected by the monitor. The Analyser module (refer Algorithm 5.3) reads the time-based location history of mobile devices from the knowledge database till time t , that is $\{X_1, X_2, \dots, X_t\}$ and forecasts the location at time X_{t+1} using the Double Exponential Smoothing method (DES).

5.3.1.3 Planning Phase

The planning phase decides the actions to be taken to achieve the goals of the MAMF (refer Algorithm 5.4). The Planning phase uses the forecasted location from the Analyse phase and calculates the distance between the user submitting request for application module, app_i and the Fog node, fn_j in which it is deployed. Equation 5.11 is used to analyse whether migration is required or not. Considering the minimum power requirements and transmission ranges of the IEEE802.11p protocol, the lower and upper thresholds for migration were fixed at $600m$ and $1000m$. The lower threshold indicates when the MAMF starts the check for a suitable Fog node for the migration of the application module. If a suitable Fog node is identified with lower value of TTC than the current Fog node on which the module is deployed, then app_i is migrated to the identified Fog node. Otherwise, the module continues to be deployed on the same node. The upper threshold indicates the maximum limit above which it is no longer feasible to continue the execution of the module on the same Fog node, since the location of the user submitting the request falls outside the coverage area. This implies that a migration of the application module is inevitable. Thus, our framework checks for a suitable Fog

Algorithm 5.3: Pseudo code for Analyse Phase
<pre>1 Begin Input : Time based Location history of a particular mobile device till time t Output: Forecasted location of the mobile device at $t + 1$ 2 Read the values of $(X_1, Y_1), (X_2, Y_2), \dots, (X_t, Y_t)$ 3 Forecasted value $(X_{t+1}, Y_{t+1}) =$ Location at $t + 1$ is forecasted using Double Exponential Smoothing method 4 Return Forecasted value (X_{t+1}, Y_{t+1}) 5 End</pre>

node to migrate to. If no such Fog nodes satisfying the requirements of the application module can be found, then the module is deployed to the Cloud, thereby ensuring that no service interruptions occur in future.

$$distance_{ij} = \begin{cases} < 600, & \text{No migration} \\ 600 - 1000, & \text{Check migration, if suitable Target Candidate then migrate} \\ > 1000, & \text{Check migration, if suitable Target Candidate then migrate,} \\ & \text{else, send to Cloud} \end{cases} \quad (5.11)$$

Check migration in lines 7, 15 of Algorithm 5.4 is a procedure that collects all the application modules requesting for migration at the current time instant. It then invokes

Algorithm 5.4: Pseudo code for Planning Phase	
1	Begin
	Input : Location at $t + 1$, (X_{t+1}, Y_{t+1})
	Output: Migration decision
2	Calculate the Cartesian distance, $distance_{ij}$ between the current position of user submitting request for application module i and the j^{th} Fog node in which the particular module is deployed
3	if $distance_{ij} < 600$ then
4	do nothing
5	end
6	else if $(distance_{ij} > 600)$ and $(distance_{ij} < 1000)$ then
7	<i>Target Candidate</i> = Check Migration(app_i)
8	if $distance_{i(Target\ Candidate)} < distance_{i(CurrentNode)}$ then
9	Migrate to <i>Target Candidate</i>
10	else
11	Do nothing
12	end
13	end
14	else
15	<i>Target Candidate</i> = Check Migration(app_i)
16	if <i>Target Candidate</i> \neq NULL then
17	Migrate to <i>Target Candidate</i>
18	else
19	Migrate to Cloud
20	end
21	end
22	End

5. Mobility aware autonomic approach for the migration of application modules in Fog Computing Environment

the GA procedure (refer Algorithm 5.6) to check for suitable destinations and returns the results to the invoking procedure. Genetic algorithm is used to identify the target candidate Fog nodes for the migration process.

5.3.1.4 Execution Phase

The Execution phase provides mechanisms to enact the plan provided by the Planner module. The action plan developed in the Plan phase is enacted in the Execution phase (refer Algorithm 5.5). If the migration destination is a Fog node, then the application module is deallocated from the current Fog node and re-allocated to the destination Fog node. If Cloud is obtained as the possible destination, then, the application module must be deallocated from the current Fog node and sent to the Cloud.

Algorithm 5.5: Pseudo code for Execute Phase

```
1 Begin  
   Input: Migration Destination  
2 if Migration destination is a Fog node then  
3   | Deallocate from current node  
4   | Allocate to destination Fog node  
5 end  
6 if Migration destination is Cloud then  
7   | Deallocate from current node  
8   | Send to Cloud  
9 else  
10  | Do Nothing  
11 end  
12 End
```

5.3.2 Genetic Algorithm

Genetic Algorithm (GA) is a classical meta-heuristic approach for solving optimization problems. GA can be effectively used to provide near-optimal solutions for NP-hard problems (Mitchell 1998)(Guerrero et al. 2018). To ensure better results using GA, choosing suitable genetic operators is paramount. Algorithm 5.6 discusses the adapted genetic algorithm used by the MAMF to identify a suitable Fog node to which the application module can be migrated. The parameter values chosen are provided in the

app_1	app_2	app_3	app_4	app_5	app_6
fn_1	fn_2	fn_3	fn_4	fn_3	fn_1

Figure 5.3: Chromosome representation example for a system with four fog nodes and six application components

algorithm. The parameter values were fixed after performing several trials with different values.

5.3.2.1 Chromosome Representation

In GA, the set of possible solutions are encapsulated in the population and each individual solution in the population is called a ‘chromosome’. The chromosome is encoded to represent the set of possible re-allocations as shown in Figure 5.3. The representation of possible re-allocations is an array where the indices of the array elements corresponds to the identifiers of the application modules to be migrated and the element values are the identifiers of Fog nodes to which the application modules can be migrated. The length of the chromosome corresponds to the number of modules which are to be migrated.

Algorithm 5.6: Genetic Algorithm

```

1 procedure GA
2  $population\ size \leftarrow 300$ 
3  $generation\ number \leftarrow 700$ 
4  $mutation\ probability \leftarrow 0.25$ 
5  $crossover\ probability \leftarrow 0.90$ 
6  $P_t \leftarrow$  Generate random population( $population\ size$ )
7  $fitness \leftarrow$  Calculate fitness( $P_t$ )
8 Assign  $iteration \leftarrow 1$ 
9 while ( $iteration < generation\ number$ ) and convergence not attained do
10   Apply Roulette wheel selection technique to select two individuals from  $P_t$ 
11   Perform crossover operation with  $crossover\ probability$ 
12   Perform mutation with  $mutation\ probability$ 
13   Calculate the fitness of each newly generated individual
14   Replace least fit in population  $P_t$  with new individuals
15   Update  $iteration \leftarrow iteration + 1$ 
16 end
17 Return the best solution (individual)

```

5.3.2.2 Crossover and mutation operator

In GA, the solutions from one population are transformed to generate the next population in each iteration. As the number of generations advance, the chromosomes in the generation draw closer to the near-optimal solution. GA progresses through the biological evolution concepts of crossover and mutation, where the better or stronger solutions are selected for reproduction and the weaker ones are eliminated.

Crossover provides structured yet randomized exchange of genetic materials among the solutions. Crossover combines two solutions to generate new solutions. The crossover operator is fuelled by the idea that the combination of two good solutions give rise to better offsprings. The combination can be interpreted based on the chromosome representation as taking pieces/portions from good solutions to generate new solutions. In the MAMF, the uniform crossover operator is used. Uniform crossover exchanges individual bits of the parents rather than dividing the array into separate segments. The bits in the new offspring are determined by swapping the bits from its parents, and the order in which they are swapped is indicated by a random real number $u \in \{0, 1\}$. The Uniform crossover operator selects two parents and generates two children from the parents such that the random number u , decides whether the i^{th} gene of the child is inherited from the first parent or the second parent.

The mutation operator is used to maintain diversity among the populations across the generations and is applied on an individual-by-individual basis. The purpose of the mutation is to avoid the local minima thereby improving the chances of obtaining better results. Random resetting was used as the mutation operator. Here, a random Fog node from a set of possible Fog nodes is selected to replace one of the Fog nodes in the solution.

5.3.2.3 Fitness function, selection operator

In GA, fitness functions are identified to quantify the quality of the solutions represented as chromosomes. Generally, fitness functions are directly derived from the objective function. Fitness functions play a vital role in ensuring the convergence of the GA. It

assigns a score to every solution and enables one to identify the best solutions in each of the generations. The score of fitness function indicates how close the obtained solution is to the desired solution.

The MAMF employs TTC_i as the GA fitness function. It represents the time to completion of the application module, app_i in the event of migration to a particular destination Fog node, fn_j . For the i^{th} application module migrating to fn_j , the TTC_i is computed as provided in Equation 5.1.

5.3.2.4 Stopping Criteria

The genetic algorithm is said to have converged if there is no much improvement in the solutions generated from one generation to the other. In the proposed MAMF, the genetic algorithm is stopped when successive iterations no longer produce better values for the fitness function.

5.4 EXPERIMENTAL EVALUATION

The experiments in our evaluation were designed to analyze the performance of the MAMF in various scenarios. The Fog simulation toolkit called iFogSim (Gupta et al. 2017) was used to simulate the Fog environment and Cloud resources. A real time mobility dataset was used in order to mimic the real time movement of users.

5.4.1 Evaluation Environment

The evaluation of our framework was done by considering a city with smart vehicle system which uses the Fog computing environment for accessing services. The simulation was done in the iFogSim simulator. iFogSim allows the modelling and simulation of Fog computing environments and creates an evaluation platform to demonstrate the capabilities of various resource management policies. Though the toolkit provides the necessary features to model the environment depicted in Section 5.2, few modifications were required. iFogSim extensions were developed to support the deployment of application modules in lightweight virtualization entities called containers. Migration of application modules hosted in containers, was enabled to support user mobility. Lo-

5. Mobility aware autonomic approach for the migration of application modules in Fog Computing Environment

Configuration number	Number of Modules	Number of Fog nodes
Config_1	4	2
Config_2	6	3
Config_3	8	4
Config_4	10	5
Config_5	20	10
Config_6	30	15
Config_7	50	25

Table 5.3: Experimental details

cation co-ordinates were added for the mobile devices to enable recording of the user location. The application modules are initially deployed according to the Edgeward Placement policy (Gupta et al. 2017) in all the considered scenarios.

5.4.2 Simulation Parameter Settings

The performance of the MAMF was evaluated in the Fog computing environment taking various combinations of application modules and Fog nodes. The details of the configurations are provided in Table 5.3.

5.4.3 Mobility Trace Description

In order to validate efficiency of our MAMF approach, a real-time mobility dataset was used. The vehicular dataset was populated by the General Departmental Council of Val de Marne situated in Creteil town, in France (Lèbre et al. 2015). The General Departmental Council is the regional agency that handles the control and co-ordination of the transportation system in France. The dataset contains the traces of the vehicle flows in the city for a period of two morning peak hours and two evening peak hours. The traces for different types of vehicles were considered in our experiments.

5.4.4 Baseline Approach

As there are no known policies for the migration of container based application modules in Fog computing environments, it is difficult to find a baseline approach to compare and evaluate the relative performance of the proposed framework. When the user moves

away from the coverage of a Fog node, in order to provide effective user mobility support, the application modules must be migrated from the present Fog node to another Fog node, closer to the current position of the user and thus provide better coverage for the user along with his/her movement. Selection of a suitable target node for the application module when a number of Fog nodes are available is challenging. An interesting baseline is introduced which selects the destination Fog node as the one with the minimum distance among the various Fog nodes that can provide coverage for the current

Algorithm 5.7: Pseudo code for SDP

```

1 Begin
   Input : Location at  $t$ ,  $(X_t, Y_t)$ 
   Output: Migration destination
2 Calculate the Cartesian distance,  $distance_{ij}$  between the current position of user
   submitting request for application module  $i$  and the  $j^{th}$  Fog node in which
   particular module deployed
3 Find all the Fog nodes which are nearer to current user position and assign them
   in to a set S.
4 Sort the set S in ascending order with respect to the current user position
5 if  $distance_{ij} < 600$  then
6 | do nothing
7 end
8 else if  $(distance_{ij} > 600) \text{ and } (distance_{ij} < 1000)$  then
9 | Consider each nodes from S, check whether the node can satisfy resource
   requirements of  $(app_i)$  if so assign it as Target Candidate
10 | if  $distance_{i(Target\ Candidate)} < distance_{i(CurrentNode)}$  then
11 | | Migrate to Target Candidate
12 | else
13 | | Do nothing
14 | end
15 end
16 else
17 | Consider each nodes from S, check whether the node can satisfy resource
   requirements of  $(app_i)$  if so assign it as Target Candidate
18 | if Target Candidate  $\neq NULL$  then
19 | | Migrate to Target Candidate
20 | else
21 | | Migrate to Cloud
22 | end
23 end
24 End

```

location of the user. The baseline method is called Shortest Distance Policy (denoted as SDP in the remainder of the chapter).

A migration strategy involves three activities: identification of when containers are to be migrated, selecting the containers to be migrated and re-allocating the migrating containers to appropriate Fog nodes. For the SDP migration strategy, containers are migrated when the users travel beyond the coverage area of the Fog nodes to which they are connected. The coverage area bounds are calculated using Equation 5.10 and also considering the minimum power requirements and transmission range of IEEE802.11p. Container migrations are initiated when the distance between the user position and the Fog node falls between $600m$ and $1000m$.

The SDP assumes that the user is within complete coverage of the Fog node when $distance_{ij}$ between the current position of user submitting request for application module i and the j^{th} Fog node in which particular module deployed is less than $600m$. When $distance_{ij}$ is in between $600m$ and $1000m$, the probability for the user to travel beyond the coverage region in the next time step is higher. Thus, SDP checks for a suitable migration destination by calculating the distance between the current user location and the different Fog nodes with sufficient resources available to host the application module. The Fog node closest to the user location is returned as the migration target node. When the $distance_{ij}$ is greater than $1000m$ since the user is already beyond the coverage area, SDP tries to find a suitable destination Fog node. If none of the suitable destinations are obtained then the module will be migrated to cloud. The SDP Algorithm is given in Algorithm 5.7.

5.4.5 Evaluation metrics

In order to assess the performance of the proposed MAMF, we have used several metrics. The first metric is the Time to Completion (TTC). Along with this metric, we have also considered the widely used evaluation metrics from the simulation environment (Gupta et al. 2017).

1. TTC_i : The TTC is the Time To Completion of a migrating application module.

When the distance between the user submitting the request for application and the Fog node on which the application is running, exceeds a bound then the proposed framework, MAMF starts to check for a suitable destination to which the module can be migrated. In order to choose one among the possible destination Fog nodes satisfying the resource requirements, we introduced the metric TTC_i . It is calculated as the sum of time required for the migration of application module, app_i to the destination Fog node, fn_j and the time required to process a request to the application module on the destination fn_j . The TTC_i is calculated as given in Equation 5.1.

2. **Network Usage:** This metric indicates the overall network usage of the application modules. As the number of devices connected to the application increases, the overall network usage increases. Uncontrolled use of the network may lead to congestions in the network which results in performance degradation of the applications.
3. **Average Loop Delay:** The processing of a user request may involve processing by a series of application modules or a loop of application modules. Users receive the response only after the complete execution of the loop of modules. A lag in any connection in this loop will impact the response time experienced by user. Thus, proper monitoring of average loop delay is essential to avoid the violations in user Service Level Agreements (SLAs).
4. **Average Tuple Execution Delay:** It is the average over the time taken to complete the execution of user requests in a particular Fog computing environment. Requests to a Fog application module contain values according to a defined data format. Each request can be thus be considered as a tuple of values, to be processed. Increased delays in processing the request may cause violations of the delay requirements. Thus, in a Fog computing environment average tuple execution delay can be considered as a measure to assess the conformance level of quality oriented services.
5. **Monetary Cost of Execution in Cloud:** In the hierarchical Fog computing en-

vironment, Fog nodes receive and pre-process data from the end devices rather than transporting the entire data in to the cloud. Thus, Fog reduces the need for increased bandwidth and also helps in reducing the congestion in the network. The data which cannot be processed by the Fog or demands long term storage are sent to the Cloud. An efficient orchestration framework for the Fog should balance between the effective utilization of Fog infrastructure and optimum usage of Cloud. The monetary cost of execution includes the cost required for the execution of application modules in the Cloud.

5.5 RESULTS

The performance of the proposed framework was evaluated using real world mobility traces based on the metrics described in Section 5.4.5. The values of these metrics were collected by averaging across several runs. We compare the proposed MAMF approach with the baseline SDP and also with the commonly used Void migration (VoMig) approach which does not offer migration support for container based application modules.

5.5.1 Network Usage

Figure 5.4 shows the overall network usage of all the application modules. The mobility of the user may cause an increase in the distance between the user and the Fog node in which the application is deployed. When the user travels beyond the coverage of Fog node, the service may be interrupted. The traditional approach VoMig, offloads the application module to the Cloud for avoiding service denial. This causes tremendous increase in the network usage, which increases with the number of modules migrating to the Cloud.

The baseline SDP approach migrates the application module to a Fog node among the set of available Fog nodes which is at the shortest distance from the current user location. The proposed framework, MAMF tries to find a suitable destination Fog node to which the application module can be migrated and the module will be offloaded to the Cloud only if a suitable destination which meets the requirements is not obtained. The migration of modules within the Fog layer does not create significant impacts on

the network usage. Thus, MAMF reduces the overall network usage in the Fog environment.

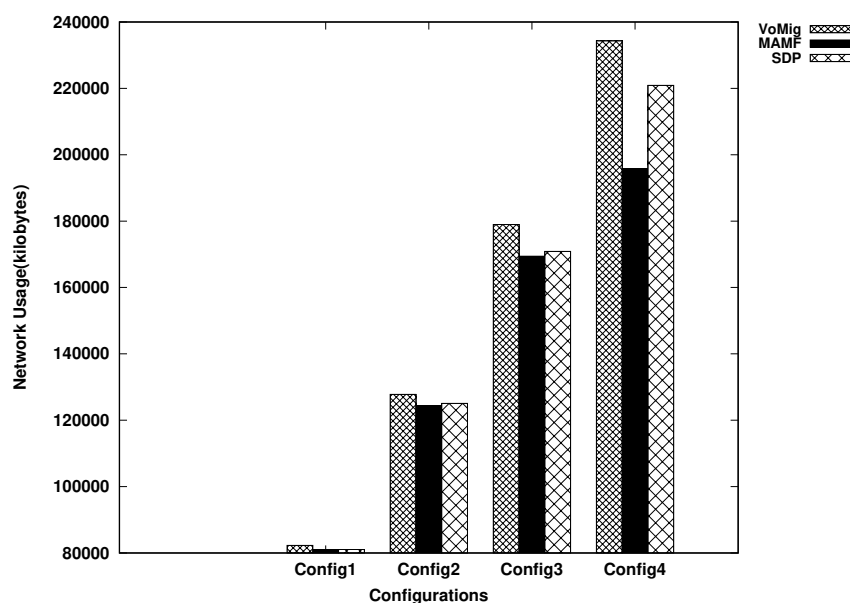


Figure 5.4: Network Usage

5.5.2 Average Loop Delay

In Fog environments, users are in motion. Once the user has travelled across the boundaries of the coverage area of the Fog node where the application is currently running, one can experience an increased delay in the execution completion time. This can be attributed to the increase in the number of hops that a request has to travel to arrive at the Fog node hosting the application. A single user request may consist of execution of a sequence of such application modules. Figure 5.5 shows the average loop delay in the various Fog computing configurations considered. The application loop delay includes the total time taken for a request to be processed by all the modules that form a loop in the application. It is observed that in the VoMig approach which does not offer migration support, more number of nodes are offloaded to the Cloud than necessary. This creates some additional communication latencies as communications with the Fog are significantly faster than communication with the distant Cloud servers. The SDP approach which supports migration based on only shortest distance tremendously reduces the amount of data uploaded to the Cloud. Our proposed approach MAMF outperforms

5. Mobility aware autonomic approach for the migration of application modules in Fog Computing Environment

both the other approaches by choosing the most appropriate migration destination based on a number of parameters.

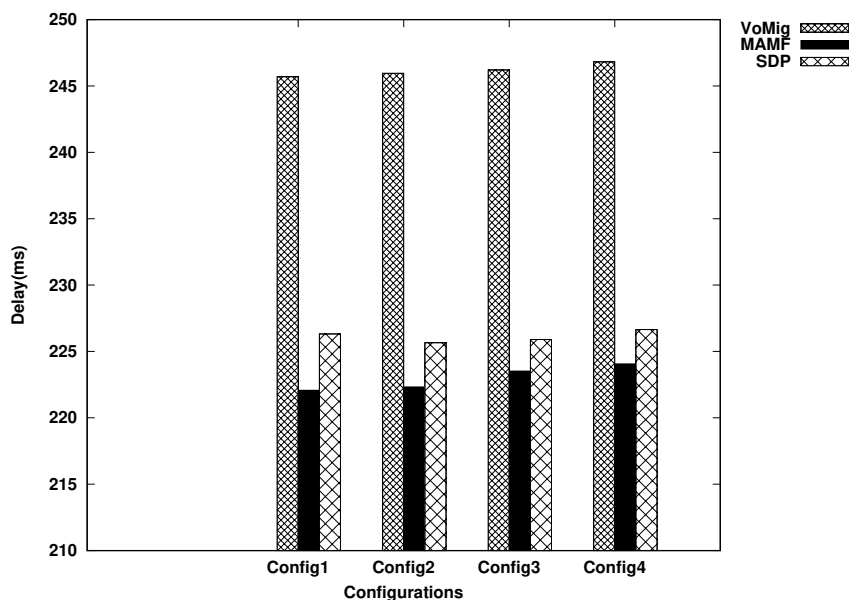


Figure 5.5: Average Loop Delay

5.5.3 Average Tuple Execution Delay

IoT datastreams emitted from the end devices are in the form of sequence of values, which is referred to as a tuple. Figure 5.6 shows the average execution delay of the request tuples. The distance between the user and the Fog node imposes some additional communication delay on the response time for each request. Enabling migration reduces the execution delay drastically in SDP. The proposed MAMF further reduces this delay by migrating the application module to a suitable Fog node which is not only located nearer to the user but also capable of efficient execution of the requests and sending quick responses to the user.

5.5.4 Execution cost in Cloud

The service requests which cannot be processed in the Fog are ultimately transferred to the Cloud. An efficient orchestrator for Fog environments must try to reduce the amount of data transfer to the Cloud and thus try to reduce the execution cost in Cloud. Figure 5.7 shows the comparison of cost of execution in Cloud for all the three approaches.

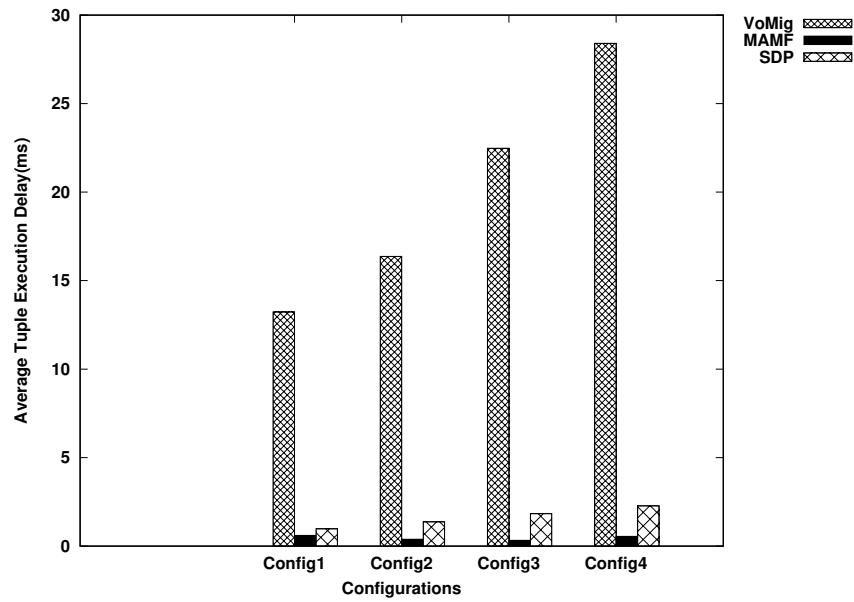


Figure 5.6: Average Tuple Delay

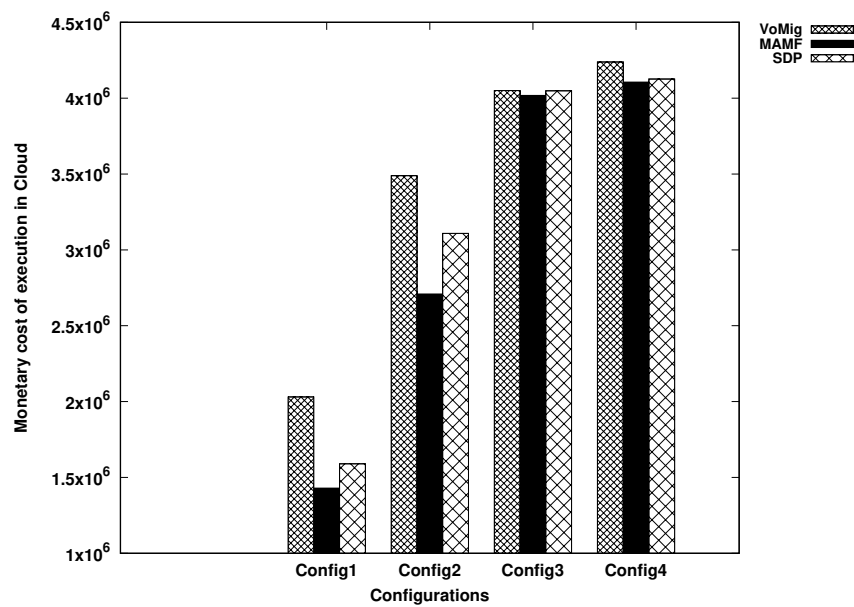


Figure 5.7: Execution Cost in Cloud

The proposed MAMF makes effective utilization of Fog environment through the timely migration of application modules to support user mobility, rather than transporting all the data to the Cloud as in the case of VoMig. The MAMF offloads modules to the Cloud, only when there are absolutely no possibilities of hosting the application module in the Fog.

The SDP approach also tries to migrate the application module to the nearest available Fog node, thus reducing the data transfer to the Cloud. But since it determines the destination based only on the distance without regard for the processing rate, new requests arriving at the destination node after the migration may suffer from longer processing delays.

5.6 DISCUSSION

In this section, we present the inferences and analysis results based on the experiments conducted.

5.6.1 Location value at time $(t + 1)$

In order to forecast the location at time $X_{(t+1)}$ from the history of location values upto time t , a time series forecasting method called Double Exponential Smoothing (DES) was used. Several methods such as Moving average (MA), Exponential Smoothing (ES), Double Exponential Smoothing (DES), AutoRegressive Integrated Moving Average (ARIMA) and AutoRegressive (AR) model, were compared. The different methods were evaluated using the metrics given in the subsequent section. The comparison results are shown in Figure 5.8. It is noted that ARIMA and DES techniques fare better when compared to all the other approaches, and among these two, DES performs exceptionally well. Thus, in MAMF, DES is used to calculate the location value at time $(t + 1)$.

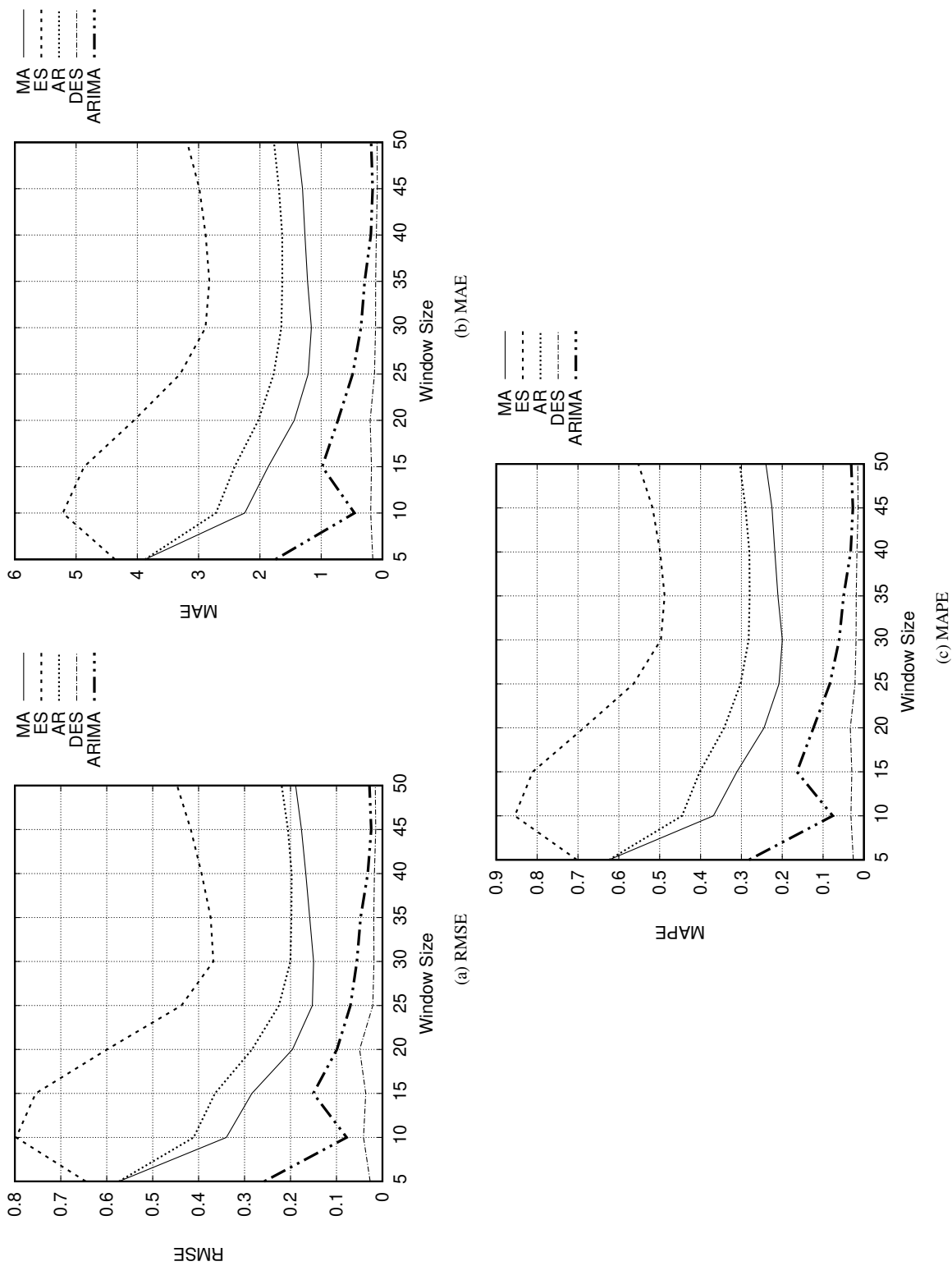


Figure 5.8: MAE, MAPE and RMSE value comparison for different prediction techniques

5.6.1.1 Evaluation criteria

The accuracy of the forecasted location values was evaluated based on several metrics, namely Mean Absolute Percentage Error (MAPE), Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE).

- Mean Absolute Error:

MAE is defined as the mean of the absolute errors. It is one of the simplest measures of accuracy. The absolute value is measured as the difference between the forecasted value and the expected value (Shcherbakov et al. 2013).

- Mean Absolute Percentage Error:

MAPE gives the accuracy of the forecasted values (Makridakis et al. 1982). The accuracy is expressed as percentage, and it is defined in Equation 5.12.

$$MAPE = \frac{1}{n} \sum_{t=1}^n \left| \frac{A_t - F_t}{A_t} \right| \quad (5.12)$$

where A_t indicates the actual value, F_t indicates the forecasted value and n is the number of observations based on which the predictions are made. A lower MAPE value indicates better accuracy.

- Root Mean Squared Error:

RMSE (Chai and Draxler 2014) is a common measure to quantify the difference between two values. It is defined in Equation 5.13.

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (\hat{Y}_i - Y_i)^2}{n}} \quad (5.13)$$

where \hat{y}_i indicates the forecasted value, y_i indicates the actual output and n is the number of observations used for prediction.

5.6.2 Evaluation of GA

This section presents the performance analysis of the GA module in the MAMF.

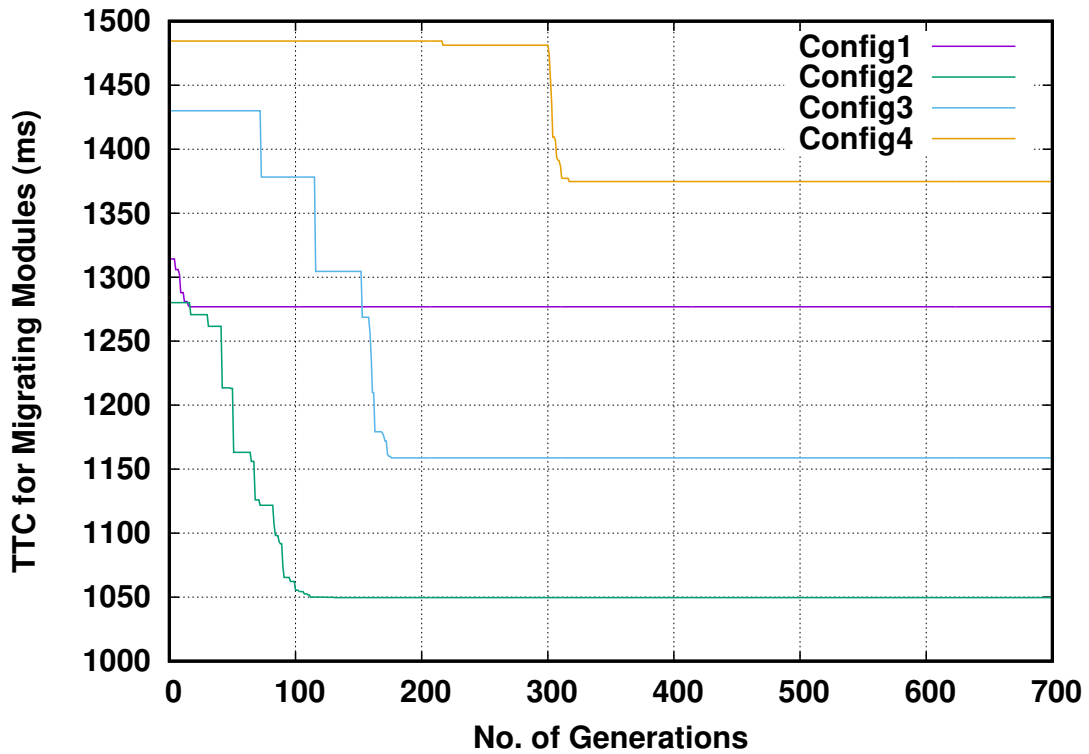


Figure 5.9: Convergence of Time to Completion (TTC) across generations for different configurations

5.6.2.1 Evolution of Objective Function

The optimal value of the objective function is obtained at the termination of the optimization algorithm. The number of generations required for the GA to converge to an optimum was fixed after several trials. The convergence of the objective function with varying number of generations is plotted in Figure 5.9. It is evident that stabilization of the objective function is achieved in different generations for different configurations of the Fog computing environment. For all the four configurations, convergence was attained before the 400th generation. It is observed that the configuration 1 shows an early convergence, which may be due to the small size of search space. Convergence in configuration 4 occurs later, that is only after 300 generations which indicates that the increase in the number of Fog nodes and application modules to migrate increases the search space thus, requiring more number of generations to converge.

Figure 5.10 shows the evolution of population across varying generations. Initially, the individuals were in random positions. It is observed that, after 200 generations,

5. Mobility aware autonomic approach for the migration of application modules in Fog Computing Environment

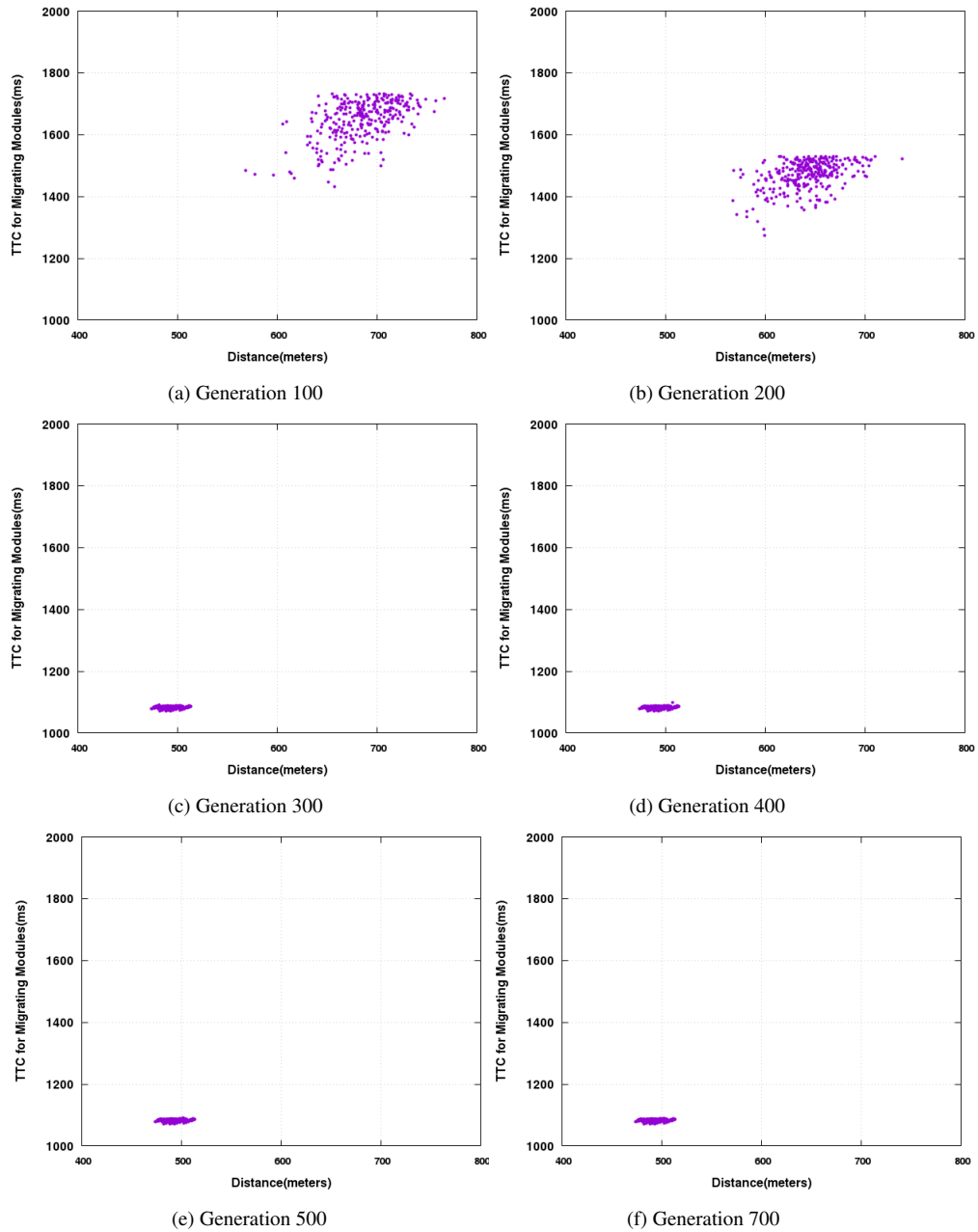


Figure 5.10: Evolution of Time to Completion with Distance across Generations

the individuals are shifted from random positions to the corresponding solution space. After 300 generations, the individuals do not undergo much changes. The distance parameter is directly proportional to the Time to completion. Through all the generations, it is observed that the distance keeps varying till the population converges to the final solution. The distance decreases with an increase in the number of generations. All the solutions are concentrated mostly at the same values of distance and TTC from the 300th generation onwards.

Figure 5.11 shows the variation of TTC of the application modules against their delays in the four different configurations. The red line indicates the tolerable delay of each of the application modules for processing the user request. TTC is the response time of a migrated application module, which includes the time to migrate and also the time to process a request in the destination Fog node. Fog computing environments with heterogeneous Fog nodes and application modules are considered. All the configurations completed the processing of requests within the tolerable delay. Thus, through the migration of application modules, MAMF ensures that the Fog computing environment fulfils the delay requirements of the users. When the total amount of resources available in the Fog layer is not sufficient to satisfy the requirements of the users, the processing will be transferred to the Cloud. The requests are immediately executed in the Cloud because of unlimited capacity of resources in the Cloud. However, it imposes additional overheads in the form of communication delay (between the user and the Cloud) due to the distance and execution cost. This may lead to violation of delay requirements of the user.

5. Mobility aware autonomic approach for the migration of application modules in Fog Computing Environment

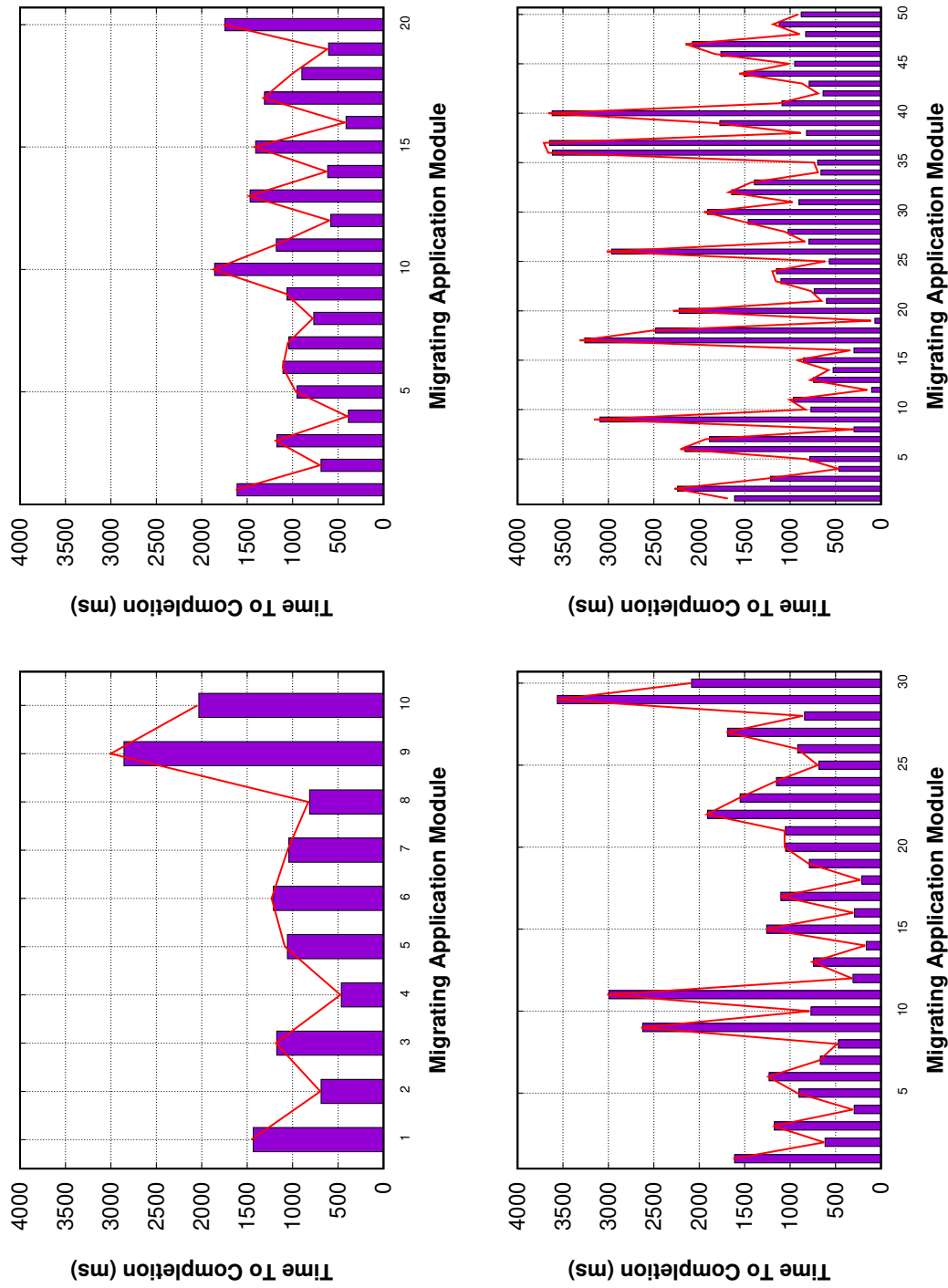


Figure 5.1.1: Variation of Time to Completion and deadline(delay) with number of migrating modules

Parameter	Solution based on ILP	Solution based on Meta-heuristic
x_{11}	1	1
x_{12}	0	0
x_{21}	0	0
x_{22}	1	1
x_{31}	0	0
x_{32}	1	1

Table 5.4: Comparison of Solutions

5.6.3 Integer Linear Programming & Heuristic Approach

In order to identify a suitable destination Fog node for the migrating application module, we have proposed approaches based on ILP and meta-heuristics. A real time example scenario was modelled as an optimization problem and the mapping of application modules to suitable Fog nodes was solved using (0/1) ILP, in Section 5.2.2. The same problem scenario is also solved using the well known meta-heuristic approach, Genetic Algorithm and the results are shown in Table 5.4.

It is observed that the heuristic-based approach approximates the ILP very well by providing the same results for the considered scenario. When the problem size increases, that is when the Fog environment consists of large number of Fog nodes and application modules, it is challenging to model the scenario using the exact technique. Therefore, in MAMF the meta-heuristic approach based on genetic algorithm is used.

5.7 SUMMARY

In order to ensure uninterrupted services to the Fog users irrespective of their mobility patterns, techniques such as application module migration may be employed. In this chapter, a mobility aware autonomic migration framework based on a combination of autonomic computing and genetic algorithm, has been proposed. MAMF ensures that the QoS requirements of the end users are satisfied, through the migration of container based application modules which works based on the concepts of control MAPE loop. The approach can be used to predetermine the migration of application modules and thus reduce the service down time. An ILP model was developed to find the destination

5. Mobility aware autonomic approach for the migration of application modules in Fog Computing Environment

Fog nodes for the migrating modules. The performance of the proposed approach was evaluated under real world mobility traces using different metrics using the iFogsim toolkit. Results show that the average delay of execution, network usage, execution delay per tuple and the cost of execution can be significantly reduced by using the proposed approach.

CHAPTER 6

CONCLUSIONS AND FUTURE SCOPE

The fog computing paradigm has emanated as a widespread computing technology to support the execution of the IoT applications. The distributed and heterogeneous features of Fog environments deem it imperative to devise efficient orchestration mechanisms. In this work, we presented dynamic decision making schemes for the optimal placement and migration of application modules in the Fog computing environment.

We proposed a service placement policy which addresses the conflicting criteria of service reliability and monetary cost. A multi-objective optimisation problem is formulated and a novel placement policy, CREW, is proposed to provide placement decisions ensuring timely service responses. Considering the exponentially large solution space, CREW adopts eagle strategy based Multi-Objective Whale Optimisation for taking placement decisions. We have considered real time microservice applications for validating our approaches, and CREW has been experimentally shown to outperform the existing popular multi-objective meta-heuristics such as NSGA-II and MOWOA based placement strategies.

The dynamic changes in the Fog computing environment and the mobile nature of users induces additional challenges and renders the initial placement infeasible. To handle these problems and challenges faced by Fog computing environments and to provide mobility support, a mobility aware autonomic migration framework (MAMF)

is proposed. MAMF performs the migration of application modules in the Fog computing environment while satisfying the QoS requirements. The hybrid framework is based on a combination of autonomic computing and genetic algorithm. The proposed approach improves user experience by reducing the delays occurring in service delivery due to user mobility. The framework uses pre-determined values of user location for the next time instant, to initiate the migration process. The relocation problem was also formulated as a integer programming problem. The MAMF framework was developed and evaluated using iFogsim simulator and experimental results indicate that the approach offers an improvement in terms of network usage, execution cost and request execution delay, over the existing approaches.

6.1 FUTURE SCOPE

Research efforts towards orchestrational issues in the domain of Fog computing is still immature and there exists several channels for future research.

- **Enhancing the security and trust by integrating blockchain:** The heterogeneous distributed nature of Fog computing demands decentralised trust and security mechanisms. The integration of blockchain with Fog computing environments can be used as an effective solution for this. However, this integration imposes several additional overheads on the resource constrained Fog computing environments. Orchestration mechanisms must be further modified to incorporate the requirements of blockchain and to effectively harness its advantages.
- **Dynamic Consolidation and Scaling:** The requirements from IoE devices to Fog environment vary over time. Thus, a resource orchestrator must be augmented with capabilities to handle the unprecedented fluctuations in the environment. Two mechanisms that can aid this process are dynamic consolidation and autoscaling. Dynamic consolidation involves re-configuration of the applications running in the Fog environments. The applications may be re-distributed across the Fog nodes to resolve degradations in the performance of the initial configuration. Autoscaling is performed to handle variations in the workload. In response

to an increase/decrease in the workload, the number of application instances may be modified.

- **Energy aware resource orchestration:** The low power characteristics of IoT devices reckon the design of orchestrational mechanisms that can reduce the overall energy consumption. For this, resource orchestration mechanisms must include energy as one of the decision parameters. Techniques such as energy profiling may be incorporated in orchestrational solutions. The concept of brownout which includes switching off the non-mandatory components can also be applied in Fog environments to enhance the energy efficiency (Klein et al. 2014).
- **Application-domain specific orchestration:** The applications from different sectors such as healthcare, agriculture, industries and transport, rely on Fog computing for real time processing of data. Requirements of applications from different sectors will vary and demand customised management. Orchestration mechanisms must leverage domain-specific knowledge while taking orchestrational decisions.

BIBLIOGRAPHY

- Aazam, M. and Huh, E.-N. (2015a). “Dynamic resource provisioning through fog micro datacenter.” *Pervasive Computing and Communication Workshops (PerCom Workshops)*, 2015 IEEE International Conference on, IEEE, 105–110.
- Aazam, M. and Huh, E.-N. (2015b). “Fog computing micro datacenter based dynamic resource estimation and pricing model for iot.” *Advanced Information Networking and Applications (AINA)*, 2015 IEEE 29th International Conference on, IEEE, 687–694.
- Aazam, M. and Huh, E.-N. (2016). “Fog computing: The cloud-iot/ioe middleware paradigm.” *IEEE Potentials*, 35(3), 40–44.
- Aazam, M., St-Hilaire, M., Lung, C.-H. and Lambadaris, I. (2016). “Pre-fog: Iot trace based probabilistic resource estimation at fog.” *Consumer Communications & Networking Conference (CCNC)*, 2016 13th IEEE Annual, IEEE, 12–17.
- Abawajy, J. H. and Hassan, M. M. (2017). “Federated internet of things and cloud computing pervasive patient health monitoring system.” *IEEE Communications Magazine*, 55(1), 48–53.
- Abbas, N., Zhang, Y., Taherkordi, A. and Skeie, T. (2018). “Mobile edge computing: A survey.” *IEEE Internet of Things Journal*, 5(1), 450–465.
- Abolfazli, S., Sanaei, Z., Ahmed, E., Gani, A. and Buyya, R. (2014). “Cloud-based augmentation for mobile devices: motivation, taxonomies, and open challenges.” *IEEE Communications Surveys & Tutorials*, 16(1), 337–368.

- Ahmed, E., Ahmed, A., Yaqoob, I., Shuja, J., Gani, A., Imran, M. and Shoaib, M. (2017). “Bringing computation closer towards user network: Is edge computing the solution?.” *IEEE Communications Magazine*, 55, 138–144.
- Akpakwu, G. A., Silva, B. J., Hancke, G. P. and Abu-Mahfouz, A. M. (2018). “A survey on 5g networks for the internet of things: communication technologies and challenges.” *IEEE Access*, 6, 3619–3647.
- Alrawais, A., Alhothaily, A., Hu, C. and Cheng, X. (2017). “Fog computing for the internet of things: Security and privacy issues.” *IEEE Internet Computing*, 21(2), 34–42.
- Alsaffar, A. A., Pham, H. P., Hong, C.-S., Huh, E.-N. and Aazam, M. (2016). “An architecture of iot service delegation and resource allocation based on collaboration between fog and cloud computing.” *Mobile Information Systems*, 2016, 15.
- Alsheikh, M. A., Niyato, D., Lin, S., Tan, H.-P. and Han, Z. (2016). “Mobile big data analytics using deep learning and apache spark.” *IEEE network*, 30(3), 22–29.
- Amendola, D., Cordeschi, N. and Baccarelli, E. (2016). “Bandwidth management vms live migration in wireless fog computing for 5g networks.” *Cloud Networking (Cloudnet), 2016 5th IEEE International Conference on, IEEE*, 21–26.
- Babu, A., Hareesh, M., Martin, J. P., Cherian, S. and Sastri, Y. (2014). “System performance evaluation of para virtualization, container virtualization, and full virtualization using xen, openvz, and xenserver.” In *Advances in Computing and Communications (ICACC), 2014 Fourth International Conference on, IEEE*, 247–250.
- Bao, L., Wu, C., Bu, X., Ren, N. and Shen, M. (2019). “Performance modeling and workflow scheduling of microservice-based applications in clouds.” *IEEE Transactions on Parallel and Distributed Systems*.
- Barcelo, M., Correa, A., Llorca, J., Tulino, A. M., Vicario, J. L. and Morell, A. (2016). “Iot-cloud service optimization in next generation smart environments.” *IEEE Journal on Selected Areas in Communications*, 34(12), 4077–4090.

- Bellavista, P. and Zanni, A. (2017). “Feasibility of fog computing deployment based on docker containerization over raspberrypi.” In *Proceedings of the 18th international conference on distributed computing and networking*, ACM, 16.
- Bi, Y., Han, G., Lin, C., Deng, Q., Guo, L. and Li, F. (2018). “Mobility support for fog computing: An sdn approach.” *IEEE Communications Magazine*, 56(5), 53–59.
- Birke, R., Giurgiu, I., Chen, L. Y., Wiesmann, D. and Engbersen, T. (2014). “Failure analysis of virtual and physical machines: patterns, causes and characteristics.” In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, IEEE, 1–12.
- Bitam, S., Zeadally, S. and Mellouk, A. (2017). “Fog computing job scheduling optimization based on bees swarm.” *Enterprise Information Systems*, 1–25.
- Bittencourt, L. F., Diaz-Montes, J., Buyya, R., Rana, O. F. and Parashar, M. (2017). “Mobility-aware application scheduling in fog computing.” *IEEE Cloud Computing*, 4(2), 26–35.
- Bittencourt, L. F., Lopes, M. M., Petri, I. and Rana, O. F. (2015). “Towards virtual machine migration in fog computing.” P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2015 10th International Conference on, IEEE, 1–8.
- Bonomi, F., Milito, R., Natarajan, P. and Zhu, J. (2014). “Fog computing: A platform for internet of things and analytics.” *Big data and internet of things: A roadmap for smart environments*, Springer, 169–186.
- Bonomi, F., Milito, R., Zhu, J. and Addepalli, S. (2012). “Fog computing and its role in the internet of things.” *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, ACM, 13–16.
- Brogi, A. and Forti, S. (2017). “Qos-aware deployment of iot applications through the fog.” *IEEE Internet of Things Journal*, 4(5), 1185–1192.

- Brogi A, Forti S, I. A. (2019). “Deploying fog applications:how much does it cost, by the way?.” In *2019 International conference on cloud computing and services science*, IEEE, 68–77.
- Carrillo, E., Benitez, V., Mendoza, C. and Pacheco, J. (2015). “Iot framework for smart buildings with cloud computing.” *Smart Cities Conference (ISC2), 2015 IEEE First International*, IEEE, 1–6.
- Chai, T. and Draxler, R. R. (2014). “Root mean square error (rmse) or mean absolute error (mae)?—arguments against avoiding rmse in the literature.” *Geoscientific model development*, 7(3), 1247–1250.
- Chen, T. and Guestrin, C. (2016). “Xgboost: A scalable tree boosting system.” In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 785–794.
- Cisco (2015). “Fog computing and the internet of things: Extend the cloud to where the things are.” Technical report, Tech. Rep.
- Cisco (2016). “Openfog architecture overview.” *White Paper, February*.
- Cisco, F. C. S. (2014). “Unleash the power of the internet of things, white paper.”).
- Cisco, V. (2018). “Cisco visual networking index: Forecast and trends, 2017–2022.” *White Paper*, 1.
- Ciuffoletti, A. (2015). “Automated deployment of a microservice-based monitoring infrastructure.” *Procedia Computer Science*, 68, 163–172.
- Dastjerdi, A. V. and Buyya, R. (2016). “Fog computing: Helping the internet of things realize its potential.” *IEEE Computer*, 49(8), 112–116.
- Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. (2002). “A fast and elitist multiobjective genetic algorithm: Nsga-ii.” *IEEE transactions on evolutionary computation*, 6(2), 182–197.

- Deng, R., Lu, R., Lai, C., Luan, T. H. and Liang, H. (2016). “Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption.” *IEEE Internet of Things Journal*, 3(6), 1171–1181.
- Dinh, T. Q., Tang, J., La, Q. D. and Quek, T. Q. (2017). “Offloading in mobile edge computing: Task allocation and computational frequency scaling.” *IEEE Transactions on Communications*, 65(8), 3571–3584.
- Do, C. T., Tran, N. H., Pham, C., Alam, M. G. R., Son, J. H. and Hong, C. S. (2015). “A proximal algorithm for joint resource allocation and minimizing carbon footprint in geo-distributed fog computing.” Information Networking (ICOIN), 2015 International Conference on, IEEE, 324–329.
- Dsouza, C., Ahn, G.-J. and Taguinod, M. (2014). “Policy-driven security management for fog computing: Preliminary framework and a case study.” Information Reuse and Integration (IRI), 2014 IEEE 15th International Conference on, IEEE, 16–23.
- Eismann, S., Kistowski, J., Grohmann, J., Bauer, A., Schmitt, N. and Kounev, S. (2019). “Teastore-a micro-service reference application.” In *2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS* W)*, IEEE, 263–264.
- Fan, C.-T., Wu, Z.-Y., Chang, C.-P. and Yuan, S. M. (2016). “Web resource cacheable edge device in fog computing.” Parallel and Distributed Computing (ISPDC), 2016 15th International Symposium on, IEEE, 432–439.
- Felter, W., Ferreira, A., Rajamony, R. and Rubio, J. (2015). “An updated performance comparison of virtual machines and linux containers.” In *Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium on*, IEEE, 171–172.
- Friis, H. T. (1946). “A note on a simple transmission formula.” *Proceedings of the IRE*, 34(5), 254–256.

- Garcia Lopez, P., Montresor, A., Epema, D., Datta, A., Higashino, T., Iamnitchi, A., Barcellos, M., Felber, P. and Riviere, E. (2015). “Edge-centric computing: Vision and challenges.” *ACM SIGCOMM Computer Communication Review*, 45(5), 37–42.
- Gavvala, S. K., Jatoth, C., Gangadharan, G. and Buyya, R. (2019). “Qos-aware cloud service composition using eagle strategy.” *Future Generation Computer Systems*, 90, 273–290.
- Gearhart, J. L., Adair, K. L., Detry, R. J., Durfee, J. D., Jones, K. A. and Martin, N. (2013). “Comparison of open-source linear programming solvers.” *Tech. Rep. SAND2013-8847*.
- Ghobaei-Arani, M., Jabbehdari, S. and Pourmina, M. A. (2016). “An autonomic approach for resource provisioning of cloud services.” *Cluster Computing*, 19(3), 1017–1036.
- Gia, T. N., Jiang, M., Rahmani, A.-M., Westerlund, T., Liljeberg, P. and Tenhunen, H. (2015). “Fog computing in healthcare internet of things: A case study on ecg feature extraction.” *Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM)*, 2015 IEEE International Conference on, IEEE, 356–363.
- Giang, N. K., Blackstock, M., Lea, R. and Leung, V. C. (2015). “Developing iot applications in the fog: a distributed dataflow approach.” *Internet of Things (IOT)*, 2015 5th International Conference on the, IEEE, 155–162.
- Goldsmith, S. F., Aiken, A. S. and Wilkerson, D. S. (2007). “Measuring empirical computational complexity.” In *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, ACM, 395–404.
- Gozálvez, J., Sepulcre, M. and Bauza, R. (2012). “Ieee 802.11 p vehicle to infrastructure communications in urban environments.” *IEEE Communications Magazine*, 50(5).

- Gross, D. (2008). *Fundamentals of queueing theory*, John Wiley & Sons.
- Gu, L., Zeng, D., Guo, S., Barnawi, A. and Xiang, Y. (2017). “Cost efficient resource management in fog computing supported medical cyber-physical system.” *IEEE Transactions on Emerging Topics in Computing*, 5(1), 108–119.
- Guan, Y., Shao, J., Wei, G. and Xie, M. (2018). “Data security and privacy in fog computing.” *IEEE Network*, (99), 1–6.
- Guerrero, C., Lera, I. and Juiz, C. (2018). “Genetic algorithm for multi-objective optimization of container allocation in cloud architecture.” *Journal of Grid Computing*, 16(1), 113–135.
- Guerrero, C., Lera, I. and Juiz, C. (2019). “A lightweight decentralized service placement policy for performance optimization in fog computing.” *Journal of Ambient Intelligence and Humanized Computing*, 10(6), 2435–2452.
- Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K. and Buyya, R. (2017). “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*, 47(9), 1275–1296.
- He, J., Wei, J., Chen, K., Tang, Z., Zhou, Y. and Zhang, Y. (2018). “Multitier fog computing with large-scale iot data analytics for smart cities.” *IEEE Internet of Things Journal*, 5(2), 677–686.
- Hevner, A. R., March, S. T., Park, J. and Ram, S. (2004). “Design science in information systems research.” *MIS quarterly*, 75–105.
- Hong, H.-J., Tsai, P.-H. and Hsu, C.-H. (2016). “Dynamic module deployment in a fog computing platform.” In *2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, IEEE, 1–6.
- Hong, K., Lillethun, D., Ramachandran, U., Ottenwalder, B. and Koldehofe, B. (2013). “Mobile fog: A programming model for large-scale applications on the internet of

- things.” Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing, ACM, 15–20.
- Hong, Y., Liu, W. M. and Wang, L. (2017). “Privacy preserving smart meter streaming against information leakage of appliance status.” *IEEE transactions on information forensics and security*, 12(9), 2227–2241.
- Hou, X., Li, Y., Chen, M., Wu, D., Jin, D. and Chen, S. (2016). “Vehicular fog computing: A viewpoint of vehicles as the infrastructures.” *IEEE Transactions on Vehicular Technology*, 65(6), 3860–3873.
- Hu, P., Dhelim, S., Ning, H. and Qiu, T. (2017). “Survey on fog computing: architecture, key technologies, applications and open issues.” *Journal of Network and Computer Applications*, 98, 27–42.
- Hu, Y. C., Patel, M., Sabella, D., Sprecher, N. and Young, V. (2015). “Mobile edge computing a key technology towards 5g.” *ETSI White Paper*, 11(11), 1–16.
- Huang, C., Lu, R. and Choo, K.-K. R. (2017). “Vehicular fog computing: architecture, use case, and security and forensic challenges.” *IEEE Communications Magazine*, 55(11), 105–111.
- Islam, M., Razzaque, A. and Islam, J. (2016). “A genetic algorithm for virtual machine migration in heterogeneous mobile cloud computing.” In *Networking Systems and Security (NSysS), 2016 International Conference on*, IEEE, 1–6.
- Jacob, B., Lanyon-Hogg, R., Nadgir, D. K. and Yassin, A. F. (2004). “A practical guide to the ibm autonomic computing toolkit.” *IBM Redbooks*, 4, 10.
- Jingtao, S., Fuhong, L., Xianwei, Z. and Xing, L. (2015). “Steiner tree based optimal resource caching scheme in fog computing.” *China Communications*, 12(8), 161–168.
- Jutila, M. (2016). “An adaptive edge router enabling internet of things.” *IEEE Internet of Things Journal*, 3(6), 1061–1069.

- Kapsalis, A., Kasnesis, P., Venieris, I. S., Kaklamani, D. I. and Patrikakis, C. Z. (2017). “A cooperative fog approach for effective workload balancing.” *IEEE Cloud Computing*, 4(2), 36–45.
- Karagiannis, G., Altintas, O., Ekici, E., Heijenk, G., Jarupan, B., Lin, K. and Weil, T. (2011). “Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions.” *IEEE communications surveys & tutorials*, 13(4), 584–616.
- Kattepur, A., Dohare, H., Mushunuri, V., Rath, H. K. and Simha, A. (2016). “Resource constrained offloading in fog computing.” Proceedings of the 1st Workshop on Middleware for Edge Clouds & Cloudlets, ACM, 1.
- Khan, S., Parkinson, S. and Qin, Y. (2017). “Fog computing security: a review of current applications and security solutions.” *Journal of Cloud Computing*, 6(1), 19.
- Klein, C., Maggio, M., Årzén, K.-E. and Hernández-Rodríguez, F. (2014). “Brownout: Building more robust cloud applications.” In *Proceedings of the 36th International Conference on Software Engineering*, 700–711.
- Kniep, C. (2014). “Containerization of high performance compute workloads using docker.” Technical report.
- Kochar, V. and Sarkar, A. (2016). “Real time resource allocation on a dynamic two level symbiotic fog architecture.” Embedded Computing and System Design (ISED), 2016 Sixth International Symposium on, IEEE, 49–55.
- Kozhircbayev, Z. and Sinnott, R. O. (2017). “A performance comparison of container-based technologies for the cloud.” *Future Generation Computer Systems*, 68, 175–182.
- Laurila, J. K., Gatica-Perez, D., Aad, I., Bornet, O., Do, T.-M.-T., Dousse, O., Eberle, J., Miettinen, M. et al. (2012). “The mobile data challenge: Big data for mobile computing research.” Workshop on the Nokia Mobile Data Challenge, in Conjunction with the 10th International Conference on Pervasive Computing, 1–8.

- Lèbre, M.-A., Le Mouël, F. and Ménard, E. (2015). “Microscopic vehicular mobility trace of europarc roundabout, creteil, france.” <http://vehicular-mobility-trace.github.io/>). Open data trace, v1.0, Creative Commons Attribution-NonCommercial 4.0 International License.
- Lee, W., Nam, K., Roh, H.-G. and Kim, S.-H. (2016). “A gateway based fog computing architecture for wireless sensors and actuator networks.” *Advanced Communication Technology (ICACT), 2016 18th International Conference on, IEEE*, 210–213.
- Lera, I., Guerrero, C. and Juiz, C. (2018). “Availability-aware service placement policy in fog computing based on graph partitions.” *IEEE Internet of Things Journal*, 6(2), 3641–3651.
- Li, Z., Zhou, X., Liu, Y., Xu, H. and Miao, L. (2017). “A non-cooperative differential game-based security model in fog computing.” *China Communications*, 14(1), 180–189.
- Liu, L., Chang, Z., Guo, X., Mao, S. and Ristaniemi, T. (2017). “Multiobjective optimization for computation offloading in fog computing.” *IEEE Internet of Things Journal*, 5(1), 283–294.
- Liu, Y., Lee, M. J. and Zheng, Y. (2016). “Adaptive multi-resource allocation for cloudlet-based mobile cloud computing system.” *IEEE Transactions on Mobile Computing*, 15(10), 2398–2410.
- Lopes, M. M., Higashino, W. A., Capretz, M. A. and Bittencourt, L. F. (2017). “Myifogsim: A simulator for virtual machine migration in fog computing.” In *Companion Proceedings of the 10th International Conference on Utility and Cloud Computing*, ACM, 47–52.
- Machen, A., Wang, S., Leung, K. K., Ko, B. J. and Salonidis, T. (2018). “Live service migration in mobile edge clouds.” *IEEE Wireless Communications*, 25(1), 140–147.
- Mahmud, R., Ramamohanarao, K. and Buyya, R. (2019a). “Latency-aware application module management for fog computing environments.” *ACM Transactions on Internet Technology (TOIT)*, 19(1), 9.

- Mahmud, R., Srirama, S. N., Ramamohanarao, K. and Buyya, R. (2019b). “Quality of experience (qoe)-aware placement of applications in fog computing environments.” *Journal of Parallel and Distributed Computing*, 132, 190–203.
- Makridakis, S., Andersen, A., Carbone, R., Fildes, R., Hibon, M., Lewandowski, R., Newton, J., Parzen, E. and Winkler, R. (1982). “The accuracy of extrapolation (time series) methods: Results of a forecasting competition.” *Journal of forecasting*, 1(2), 111–153.
- Marín-Tordera, E., Masip-Bruin, X., García-Almiñana, J., Jukan, A., Ren, G.-J. and Zhu, J. (2017). “Do we all really know what a fog node is? current trends towards an open definition.” *Computer Communications*, 109, 117–130.
- Masip-Bruin, X., Marín-Tordera, E., Alonso, A. and Garcia, J. (2016). “Fog-to-cloud computing (f2c): The key technology enabler for dependable e-health services deployment.” *Ad Hoc Networking Workshop (Med-Hoc-Net)*, 2016 Mediterranean, IEEE, 1–5.
- Mayer, R., Gupta, H., Saurez, E. and Ramachandran, U. (2017). “The fog makes sense: Enabling social sensing services with limited internet connectivity.” *Proceedings of the 2nd International Workshop on Social Sensing*, ACM, 61–66.
- Mekki, T., Jabri, I., Rachedi, A. and ben Jemaa, M. (2017). “Vehicular cloud networks: Challenges, architectures, and future directions.” *Vehicular Communications*, 9, 268–280.
- Mijumbi, R., Serrat, J., Gorricho, J.-L., Bouten, N., De Turck, F. and Boutaba, R. (2016). “Network function virtualization: State-of-the-art and research challenges.” *IEEE Communications Surveys & Tutorials*, 18(1), 236–262.
- Mirjalili, S. and Lewis, A. (2016). “The whale optimization algorithm.” *Advances in engineering software*, 95, 51–67.
- Mitchell, M. (1998). *An introduction to genetic algorithms*, MIT press.

- Moreno-Vozmediano, R., Montero, R. S., Huedo, E. and Llorente, I. M. (2017). “Cross-site virtual network in cloud and fog computing.” *IEEE Cloud Computing*, 4(2), 46–53.
- Mukherjee, M., Matam, R., Shu, L., Maglaras, L., Ferrag, M. A., Choudhury, N. and Kumar, V. (2017). “Security and privacy in fog computing: Challenges.” *IEEE Access*, 5, 19293–19304.
- Natesha, B. and Guddeti, R. M. R. (2018). “Heuristic-based iot application modules placement in the fog-cloud computing environment.” In *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, IEEE, 24–25.
- Ni, L., Zhang, J., Jiang, C., Yan, C. and Yu, K. (2017). “Resource allocation strategy in fog computing based on priced timed petri nets.” *IEEE Internet of Things Journal*, 4(5), 1216–1228.
- Ottenwälder, B., Koldehofe, B., Rothermel, K. and Ramachandran, U. (2013). “Migcep: operator migration for mobility driven distributed complex event processing.” Proceedings of the 7th ACM international conference on Distributed event-based systems, ACM, 183–194.
- Oueis, J., Strinati, E. C., Sardellitti, S. and Barbarossa, S. (2015). “Small cell clustering for efficient distributed fog computing: A multi-user case.” Vehicular Technology Conference (VTC Fall), 2015 IEEE 82nd, IEEE, 1–5.
- Pallewatta, S., Kostakos, V. and Buyya, R. (2019). “Microservices-based iot application placement within heterogeneous and resource constrained fog computing environments.” In *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*, 71–81.
- Pan, J. and McElhannon, J. (2018). “Future edge cloud and edge computing for internet of things applications.” *IEEE Internet of Things Journal*, 5(1), 439–449.

- Pang, Z., Sun, L., Wang, Z., Tian, E. and Yang, S. (2015). “A survey of cloudlet based mobile computing.” *Cloud Computing and Big Data (CCBD)*, 2015 International Conference on, IEEE, 268–275.
- Pathan, A.-M. K. and Buyya, R. (2007). “A taxonomy and survey of content delivery networks.” *Grid Computing and Distributed Systems Laboratory, University of Melbourne, Technical Report*, 4.
- Peng, M., Yan, S., Zhang, K. and Wang, C. (2016). “Fog-computing-based radio access networks: issues and challenges.” *IEEE Network*, 30(4), 46–53.
- Preden, J. S., Tammemäe, K., Jantsch, A., Leier, M., Riid, A. and Calis, E. (2015). “The benefits of self-awareness and attention in fog and mist computing.” *IEEE Computer*, 48(7), 37–45.
- Qiu, T., Qiao, R. and Wu, D. O. (2018). “Eabs: An event-aware backpressure scheduling scheme for emergency internet of things.” *IEEE Transactions on Mobile Computing*, 17(1), 72–84.
- Qiu, T., Zheng, K., Song, H., Han, M. and Kantarci, B. (2017). “A local-optimization emergency scheduling scheme with self-recovery for a smart grid.” *IEEE Transactions on Industrial Informatics*, 13(6), 3195–3205.
- Sabella, D., Vaillant, A., Kuure, P., Rauschenbach, U. and Giust, F. (2016). “Mobile-edge computing architecture: The role of mec in the internet of things.” *IEEE Consumer Electronics Magazine*, 5(4), 84–91.
- Santoro, D., Zozin, D., Pizzolli, D., De Pellegrini, F. and Cretti, S. (2017). “Foggy: a platform for workload orchestration in a fog computing environment.” *Cloud Computing Technology and Science (CloudCom)*, 2017 IEEE International Conference on, IEEE, 231–234.
- Sarkar, S., Chatterjee, S. and Misra, S. (2015). “Assessment of the suitability of fog computing in the context of internet of things.” *IEEE Transactions on Cloud Computing*, 6(1), 46–59.

- Sarkar, S. and Misra, S. (2016). “Theoretical modelling of fog computing: a green computing paradigm to support iot applications.” *IET Networks*, 5(2), 23–29.
- Satyanarayanan, M., Bahl, P., Caceres, R. and Davies, N. (2009). “The case for vm-based cloudlets in mobile computing.” *IEEE pervasive Computing*, 8(4).
- Saurez, E., Hong, K., Lillethun, D., Ramachandran, U. and Ottenwalder, B. (2016). “Incremental deployment and migration of geo-distributed situation awareness applications in the fog.” Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems, ACM, 258–269.
- Shang, W., Yu, Y., Droms, R. and Zhang, L. (2016). “Challenges in iot networking via tcp/ip architecture.” *Technical Report NDN-0038. NDN Project*.
- Sharma, P. K., Chen, M.-Y. and Park, J. H. (2018). “A software defined fog node based distributed blockchain cloud architecture for iot.” *IEEE Access*, 6, 115–124.
- Sharma, S. K. and Wang, X. (2017). “Live data analytics with collaborative edge and cloud processing in wireless iot networks.” *IEEE Access*, 5, 4621–4635.
- Sharma, Y., Javadi, B., Si, W. and Sun, D. (2016). “Reliability and energy efficiency in cloud computing systems: Survey and taxonomy.” *Journal of Network and Computer Applications*, 74, 66–85.
- Shcherbakov, M. V., Brebels, A., Shcherbakova, N. L., Tyukov, A. P., Janovsky, T. A. and Kamaev, V. A. (2013). “A survey of forecast error measures.” *World Applied Sciences Journal*, 24(2013), 171–176.
- Shi, W., Cao, J., Zhang, Q., Li, Y. and Xu, L. (2016). “Edge computing: Vision and challenges.” *IEEE Internet of Things Journal*, 3(5), 637–646.
- Shih, Y.-Y., Chung, W.-H., Pang, A.-C., Chiu, T.-C. and Wei, H.-Y. (2017). “Enabling low-latency applications in fog-radio access networks.” *IEEE Network*, 31(1), 52–58.
- Singh, S., Chiu, Y.-C., Tsai, Y.-H. and Yang, J.-S. (2016). “Mobile edge fog computing in 5g era: Architecture and implementation.” Computer Symposium (ICS), International, IEEE, 731–735.

- Skarlat, O., Nardelli, M., Schulte, S., Borkowski, M. and Leitner, P. (2017). “Optimized iot service placement in the fog.” *Service Oriented Computing and Applications*, 11(4), 427–443.
- Skarlat, O., Schulte, S., Borkowski, M. and Leitner, P. (2016). “Resource provisioning for iot services in the fog.” *Service-Oriented Computing and Applications (SOCA)*, 2016 IEEE 9th International Conference on, IEEE, 32–39.
- Stanciu, A. (2017). “Blockchain based distributed control system for edge computing.” In *Control Systems and Computer Science (CSCS), 2017 21st International Conference on*, IEEE, 667–671.
- Stojmenovic, I. and Wen, S. (2014). “The fog computing paradigm: Scenarios and security issues.” *Computer Science and Information Systems (FedCSIS)*, 2014 Federated Conference on, IEEE, 1–8.
- Tan, B., Ma, H., Mei, Y. and Zhang, M. (2018). “Evolutionary multi-objective optimization for web service location allocation problem.” *IEEE Transactions on Services Computing*.
- Tandon, R. and Simeone, O. (2016). “Harnessing cloud and edge synergies: toward an information theory of fog radio access networks.” *IEEE Communications Magazine*, 54(8), 44–50.
- Tang, B., Chen, Z., Hefferman, G., Pei, S., Wei, T., He, H. and Yang, Q. (2017a). “Incorporating intelligence in fog computing for big data analysis in smart cities.” *IEEE Transactions on Industrial Informatics*, 13(5), 2140–2150.
- Tang, M., Gao, L., Pang, H., Huang, J. and Sun, L. (2017b). “Optimizations and economics of crowdsourced mobile streaming.” *IEEE Communications Magazine*, 55(4), 21–27.
- Thiebaut, R. (2019). “Ai revolution: How data can identify and shape consumer behavior in ecommerce.” In *Entrepreneurship and Development in the 21st Century*, Emerald Publishing Limited, 191–229.

- Tomovic, S., Yoshigoe, K., Maljevic, I. and Radusinovic, I. (2017). “Software-defined fog network architecture for iot.” *Wireless Personal Communications*, 92(1), 181–196.
- Touati, L. and Challal, Y. (2016). “Collaborative kp-abe for cloud-based internet of things applications.” *Communications (ICC), 2016 IEEE International Conference on*, IEEE, 1–7.
- Truong, N. B., Lee, G. M. and Ghamri-Doudane, Y. (2015). “Software defined networking-based vehicular adhoc network with fog computing.” *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, IEEE, 1202–1207.
- Urgaonkar, B., Rosenberg, A. L. and Shenoy, P. (2007). “Application placement on a cluster of servers.” *International Journal of Foundations of Computer Science*, 18(05), 1023–1041.
- van Lingen, F., Yannuzzi, M., Jain, A., Irons-Mclean, R., Lluch, O., Carrera, D., Perez, J. L., Gutierrez, A., Montero, D., Marti, J. et al. (2017). “The unavoidable convergence of nfv, 5g, and fog: A model-driven approach to bridge cloud and edge.” *IEEE Communications Magazine*, 55(8), 28–35.
- Vaquero, L. M. and Rodero-Merino, L. (2014). “Finding your way in the fog: Towards a comprehensive definition of fog computing.” *ACM SIGCOMM Computer Communication Review*, 44(5), 27–32.
- Velasquez, K., Abreu, D. P., Curado, M. and Monteiro, E. (2017). “Service placement for latency reduction in the internet of things.” *Annals of Telecommunications*, 72(1-2), 105–115.
- Venticinque, S. and Amato, A. (2019). “A methodology for deployment of iot application in fog.” *Journal of Ambient Intelligence and Humanized Computing*, 10(5), 1955–1976.
- Verbelen, T., Simoens, P., De Turck, F. and Dhoedt, B. (2012). “Cloudlets: Bringing the cloud to the mobile user.” *Proceedings of the third ACM workshop on Mobile cloud computing and services*, ACM, 29–36.

- Vilalta, R., Mayoral, A., Casellas, R., Martínez, R. and Muñoz, R. (2016). “Experimental demonstration of distributed multi-tenant cloud/fog and heterogeneous sdn/nfv orchestration for 5g services.” *Networks and Communications (EuCNC), 2016 European Conference on, IEEE*, 52–56.
- Wang, S., Urgaonkar, R., Zafer, M., He, T., Chan, K. and Leung, K. K. (2015). “Dynamic service migration in mobile edge-clouds.” *IFIP Networking Conference (IFIP Networking), 2015, IEEE*, 1–9.
- Wen, Z., Yang, R., Garraghan, P., Lin, T., Xu, J. and Rovatsos, M. (2017). “Fog orchestration for internet of things services.” *IEEE Internet Computing*, 21(2), 16–24.
- Wu, J., Zhang, Z., Hong, Y. and Wen, Y. (2015). “Cloud radio access network (c-ran): a primer.” *IEEE Network*, 29(1), 35–41.
- Xia, F., Ding, F., Li, J., Kong, X., Yang, L. T. and Ma, J. (2014). “Phone2cloud: Exploiting computation offloading for energy saving on smartphones in mobile cloud computing.” *Information Systems Frontiers*, 16(1), 95–111.
- Xiao, Y. and Zhu, C. (2017). “Vehicular fog computing: Vision and challenges.” *Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on, IEEE*, 6–9.
- Xu, Y., Mahendran, V. and Radhakrishnan, S. (2016). “Towards sdn-based fog computing: Mqtt broker virtualization for effective and reliable delivery.” *Communication Systems and Networks (COMSNETS), 2016 8th International Conference on, IEEE*, 1–6.
- Yan, Y. and Su, W. (2016). “A fog computing solution for advanced metering infrastructure.” *Transmission and Distribution Conference and Exposition (T&D), 2016 IEEE/PES, IEEE*, 1–4.
- Yang, X.-S. and Deb, S. (2010). “Eagle strategy using lévy walk and firefly algorithms for stochastic optimization.” In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, Springer, 101–111.

- Yangui, S., Ravindran, P., Bibani, O., Glitho, R. H., Hadj-Alouane, N. B., Morrow, M. J. and Polakos, P. A. (2016). “A platform as-a-service for hybrid cloud/fog environments.” *Local and Metropolitan Area Networks (LANMAN), 2016 IEEE International Symposium on*, IEEE, 1–7.
- Yi, S., Li, C. and Li, Q. (2015a). “A survey of fog computing: concepts, applications and issues.” *Proceedings of the 2015 Workshop on Mobile Big Data*, ACM, 37–42.
- Yi, S., Qin, Z. and Li, Q. (2015b). “Security and privacy issues of fog computing: A survey.” *International Conference on Wireless Algorithms, Systems, and Applications*, Springer, 685–695.
- Yousefpour, A., Patil, A., Ishigaki, G., Kim, I., Wang, X., Cankaya, H. C., Zhang, Q., Xie, W. and Jue, J. P. (2018). “Qos-aware dynamic fog service provisioning.” *arXiv preprint arXiv:1802.00800*.
- Zeng, D., Gu, L., Guo, S., Cheng, Z. and Yu, S. (2016). “Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system.” *IEEE Transactions on Computers*, 65(12), 3702–3712.
- Zhang, H., Qiu, Y., Chu, X., Long, K. and Leung, V. (2017a). “Fog radio access networks: management, interference mitigation and resource optimization.” *arXiv preprint arXiv:1707.06892*.
- Zhang, H., Xiao, Y., Bu, S., Niyato, D., Yu, F. R. and Han, Z. (2017b). “Computing resource allocation in three-tier iot fog networks: A joint optimization approach combining stackelberg game and matching.” *IEEE Internet of Things Journal*, 4(5), 1204–1215.
- Zhang, H., Zhang, Y., Gu, Y., Niyato, D. and Han, Z. (2017c). “A hierarchical game framework for resource management in fog computing.” *IEEE Communications Magazine*, 55(8), 52–57.
- Zhu, J., Chan, D. S., Prabhu, M. S., Natarajan, P., Hu, H. and Bonomi, F. (2013). “Improving web sites performance using edge servers in fog computing architecture.”

Service Oriented System Engineering (SOSE), 2013 IEEE 7th International Symposium on, IEEE, 320–323.

RESEARCH OUTCOMES

PUBLICATIONS

Journal Papers

1. Martin, J. P. , A. Kandasamy & K. Chandrasekaran (2018). Exploring the support for high performance applications in the container runtime environment. Springer Human-centric Computing and Information Sciences, 2018, 8(1), 1-15. (DOI: <https://doi.org/10.1186/s13673-017-0124-3>, URL: <https://link.springer.com/article/10.1186/s13673-017-0124-3>) **[SCI Indexed]**
2. Martin, J. P. , A. Kandasamy , K. Chandrasekaran & C. T. Joseph. (2019). Elucidating the challenges for the praxis of fog computing: An aspect-based study. Wiley International Journal of Communication Systems, 2019, 32(7), 1-28. (DOI: <https://doi.org/10.1002/dac.3926>, URL: <https://onlinelibrary.wiley.com/doi/epdf/10.1002/dac.3926>) **[SCI Indexed]**
3. Martin, J. P. , A. Kandasamy & K. Chandrasekaran (2019). Mobility aware autonomic approach for the migration of application modules in fog computing environment. Springer Journal of Ambient Intelligence and Humanized Computing, 2020, 1-20 (DOI: <https://doi.org/10.1007/s12652-020-01854-x>, URL: <https://link.springer.com/article/10.1007/s12652-020-01854-x>) **[SCI Indexed]**
4. Martin, J. P. , A. Kandasamy & K. Chandrasekaran. CREW: Cost and Reliability aware Eagle-Whale optimizer for Service Placement in Fog. Wiley Software:

Practice and Experience, 2020. ("Accepted")(DOI: <https://doi.org/10.1002/spe.2896> ,)[**SCI Indexed**]

Conference Papers

1. Martin, J. P. , A. Kandasamy & K. Chandrasekaran (2019). Unraveling the challenges for the application of fog computing in different realms: A multifaceted study. *Integrated Intelligent Computing, Communication and Security*, vol. 771, pp. 481-492. Springer, Singapore. (DOI: https://doi.org/10.1007/978-981-10-8797-4_49, URL: https://link.springer.com/chapter/10.1007/978-981-10-8797-4_49)

BIODATA

Name : John Paul Martin
Email : johnpm12@gmail.com
Date of Birth : 12th January 1990
Permanent address : John Paul Martin,
S/o Martin P John,
Pottakka House,
Azhakam, Kodakara Post, Thrissur District,
Kerala-680684

Educational Qualifications :

Degree	Year	Institution / University
B.Tech.(CSE)	2011	University of Calicut, Kerala.
M.Tech.(CSE)	2014	Mahatma Gandhi University, Kerala.