FOG BASED FRAMEWORKS FOR IOT/IIOT SERVICE PLACEMENT AND DATA ANALYTICS IN SMART APPLICATION ENVIRONMENTS

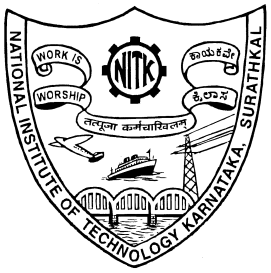
Thesis

Submitted in partial fulfilment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

Mr. NATESHA B V



DEPARTMENT OF INFORMATION TECHNOLOGY

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA
SURATHKAL, MANGALORE - 575 025, INDIA

May, 2022

Declaration

I hereby *declare* that the Research Thesis entitled " Fog Based Frameworks for IoT/IIoT Service Placement and Data Analytics in Smart Application Environments" 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 Information Technology is a *bonafide report of the research work carried out by me*. The material contained in this thesis has not been submitted to any University or Institution for the award of any degree.

Place : NITK Surathkal
Date : 09-05-2022

Mr. Natesha B V (Reg.No.: IT16FV05)
Department of Information Technology

Certificate

This is to *certify* that the Research Thesis entitled ”Fog Based Frameworks for IoT/IIoT Service Placement and Data Analytics in Smart Application Environments” submitted by Mr. Natesha B V (Register Number: 165032IT16FV05) as the record of the research work carried out by him, is *accepted as the Research Thesis submission* in partial fulfilment of the requirements for the award of degree of Doctor of Philosophy.

Prof. Ram Mohana Reddy Guddeti
Research Guide and Senior Professor
Department of Information Technology
NITK Surathkal - 575 025, INDIA

Chairman - DRPC
(Signature with Date and Seal)

This thesis is dedicated to

My Parents and Family

Acknowledgements

Foremost, my sincere thanks to my research guide Prof. Ram Mohana Reddy Guddeti, Senior Professor, Department of Information Technology, NITK Surathkal, for his valuable guidance, enthusiasm, inspiration, and dedication throughout my research. This research work is a result of his timely suggestions and firm decisions.

My sincere thanks to the RPAC members, Prof. Gangadharan K V and Prof. Shashidhar G Koolagudi, for their timely and valuable suggestions. I also thank all the teaching, technical, administrative, and non-teaching staff who have been very kind and helpful throughout my research work.

I convey my sincere thanks to my beloved grandparents, parents, sisters, and family members for their continuous moral support throughout my life, encouragement at every point of my research work.

I extend my sincere thanks to my dear one's Dr. Shiva Darshan, Dr. Karthik N, Dr. Ashwin who has been part of my research journey and made it more comfortable and enjoyable. I extend my sincere thanks to Mrs. Rashmi M, Mr. Sanket S Salvi, and Mr. Rahul M V for their valuable suggestions, inputs. Also, I thank all my friends who have been helpful and supportive during my research.

I extend my sincere thanks to the Department of MeitY, Government of India, for providing the financial support under Visvesvaraya Ph.D. Scheme for Electronics and IT to carry out the research work.

Finally, my sincere thanks to NITK Surathkal, Karnataka, India, the most exciting place for making this research more memorable and possible by facilitating updated infrastructure to carry out my research.

Natesha B V

Abstract

There is an exponential increase in Internet of Things (IoT) devices in smart environments to monitor and control activities. The use of IoT devices in these environments increased the computational and storage resources requirement. Cloud computing provides computational and storage resources, but it requires entire data to be transferred to the cloud. Using cloud computing for all IoT/Industrial IoT (IIoT) applications is not feasible as some of these applications are delay-sensitive and require service in real-time to avoid significant failures. Hence, a distributed fog computing architecture is developed to provide the computational and storage resources at the network edge to process and analyze the data. The main research challenges in a fog computing environment are: to realize the fog computing infrastructure on resource constrained devices using the virtualization technique to provide the computational resources. Further, it is challenging to use these fog nodes for service placement and deploy a machine learning model for real-time data analytics. This research work focuses on developing fog frameworks for IoT/IIoT service placement and the machine learning model deployment to process and analyze the sensor data to reduce the service time and resource consumption and thus enable real-time monitoring of the smart environments.

The Fog-Cloud computing environment is used to place the IoT/IIoT services based on the resource availability and deadline to address the above research challenges. The service placement problem in the fog-cloud computing environment is formulated as a multi-objective optimization problem and a novel cost-efficient deadline-aware service placement algorithm is developed to place the services on the Fog-Cloud resources to ensure the QoS of the IoT/IIoT services in terms of deadline, service cost and resource availability. Using simulators or virtual machines based resource provisioning framework is not feasible as it takes more time and consumes more resources. Hence, the container-based fog computing framework is developed on 1.4 GHz 64-bit quad-core processor devices to realize the fog computing architecture on the resource constrained devices. Further, the service placement problem in the fog computing environment is formulated as a multi-objective optimization problem and the meta-heuristic algorithms such as Elite Genetic Algorithm (EGA), Modified Genetic Algorithm with Particle Swarm Optimization (MGAPSO) and EGA with Particle Swarm Optimization (EGAPSO) are developed for IoT/IIoT service placement in the fog computing environment. The experimental results show that using a hybrid EGAPSO based service placement on the fog nodes reduces service time, cost and energy consumption.

Using fog nodes for deploying the machine learning models to analyze the data reduces the size of the data to be transferred to the cloud, which might reduce the network congestion, reduce the service time and thus enable to make quick decisions. The fog server-based framework is developed as a prototype for intelligent machine malfunction monitoring in the Industry 4.0 environment. The various supervised machine learning models are developed and deployed on the fog server at the network edge to analyze the data and thus enable real-time monitoring in the smart industry/Industry 4.0 environment. The fog server framework is used for industrial machine monitoring at Smart Industry/Industry 4.0 to detect and classify the machine as normal and abnormal using the machine operating sounds. The experimental results show the machine learning models' performance for the various machines' sounds recorded with different Signal to Noise Ratio levels for normal and abnormal operations using Linear Prediction Coefficients and Mel Frequency Cepstral Coefficient audio features. Using fog server prototype for monitoring will reduce the total time and thus avoids the significant machines failures in the industrial environment.

Keywords: Abnormal, Containers, Energy Consumption, Industry 4.0, Internet of Things, IIoT, Malfunction Monitoring, Meta-heuristic, MFCC, Normal, Resource Provisioning, Service Placement.

Contents

List of Tables

List of Figures

List of Abbreviations

Abbreviation	Meaning
IoT	Internet of Things
IIoT	Industrial Internet of Things
PMs	Physical Machines
OS	Operating System
VMs	Virtual Machines
SDN	Software Defined Networking
NFV	Network Function Virtualization
MDC	Micro Data Center
DVFS	Dynamic Voltage and Frequency Scaling
SaaS	Software as a Service
PaaS	Platfrom as a Service
IaaS	Infrastructure as a Service
CaaS	Communication as a Service
XaaS	Anything as a Service
ML	Machine Learning
MILP	Mixed Integer Linear Programming
CNN	Convolutional Neural Network
API	Application Programming Interface
MEC	Mobile Edge Computing
MINLP	Mixed Integer Non-Linear Programming
QoS	Quality of Service
QoE	Quality of Experience
DASP	Deadline Aware Service Placement
MQTT	Message Queuing Telemetry Transport
FDK	Fog Development Kit
IoV	Internet of Vehicles
GENI	Global Environment for Network Innovations
MIPS	Millions of Instructions per Second
PSO	Particle Swarm Optimzation
GA	Genetic Algorithm
FFD	First-Fit Decreasing
BLB	Benchmark Lower Bound
FMN	Fog Master Node
FC	Fog Cells

FCR	Fog Cell Registry
FSRSA	Fog Service Registry and Service Allocation
FCC	Fog Cell Controller
IP	Internet Protocol
EGA	Elitism-based Genetic Algorithm
DMS	Double Matching System
DEBTS	Delay Energy based Task Scheduling
GAPSO	Genetic Algorithm and Particle Swarm Optimization
MGAPSO	Modified Genetic Algorithm and Particle Swarm Optimizaion
EGAPSO	Elitism-based Genetic Algorithm and Particle Swarm Optimizaion
BB	Branch-and-Bound
FF	First-Fit
SA	Simulated Annealing
LPC	Linear Prediction Coefficients
MFCC	Mel Frequency Cepstral Coefficients
ICU	Industrial Controller Unit
FFT	Fast Fourier Transform
DCT	Discrete Cosine Transform
RF	Random Forest
SVM	Support Vector Machine
LR	Logistic Regression
AdaB	AdaBoost Classifier
MLP	Multi-Layer Perceptron
SNR	Signal to Noise Ratio
MIMII	Malfunctioning Industrial Machine Investigation and Inspection
ANN	Artificial Neural Network
ROC	Receiver Operating Characteristic
TPR	True Positive Rate
FPR	False Positive Rate

Chapter 1

Introduction

1.1 Internet of Things

Internet of Things (IoT) is defined as the group of physical devices connected over the internet. IoT has become the most pivotal technology since many devices such as home and kitchen appliances, thermostats, etc., are connected to the internet and enable seamless communication between humans, processes, and things. These IoT devices allow a new way of interconnection and communication between the devices and thus generate a huge amount of data. It is projected that the number of IoT devices will reach over 75.44 billion, and these devices will generate enormous data by the year 2025 (Alavi et al., 2018). Using IoT devices in different environments automates the processes, reduces human intervention, cost, service time, and improves the user experience (Atzori et al., 2010). IoT enables the use of the resources/infrastructure efficiently, enhances people's awareness about their cities, and encourages them to actively manage the resources and create new applications using IoT devices.

The things in the IoT are classified into the following:

- **Collect data and send:** The connected sensor devices such as temperature, motion, air quality, moisture sensors, etc., collect data and transfer it to process and analyze to make intelligent decisions.
- **Receive data and act:** These are the connected devices that receive data and perform some actions. For example, the signal from the car key opens or locks the car door.
- **Things perform both send and act:** The more powerful things can perform both operations, i.e., can send, receive data, and act. These devices will collect the data and, based on the data, decide to send the control signal to act. For example, in the automated irrigation control system, the moisture sensor collects the data. Then the control system automatically decides to watering or not to water the crops based on the sensor data.

1.1.1 IoT Applications

The IoT applications bring more value into our lives and reduce time, human intervention, and cost. Many real-world applications make use of more IoT devices to automate monitoring and controlling operations. Figure 1.1 shows diverse real-world smart environments that use the IoT.

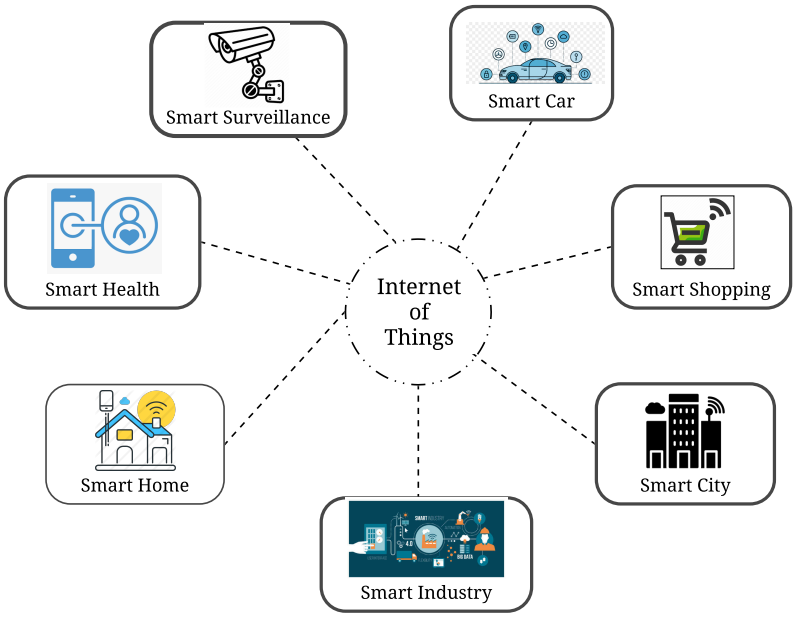


Figure 1.1: Smart Environments where large number of IoT devices are deployed to monitor and control the environment

1.1.1.1 Smart City Applications

The increase in urbanization and population in the cities requires ensuring a clean, decent life quality for the citizens. Hence, governments are considering developing cities as smart cities using technologies (Chatterjee et al., 2018). A smart city is a public area that uses IoT devices to monitor and control the city's environment. Using IoT devices enables smart governance, smart economy, smart mobility, smart environment, and smart living (Arasteh et al., 2016; Simmhan et al., 2018). In the smart city, the IoT devices are used for automated and smart light control (Ouerhani et al., 2016), cameras and other connected devices are used for intelligent traffic control and parking monitoring (Al-Turjman and Malekloo, 2019), traffic congestion and vehicle movement, smart water supply management, waste management, etc. (Zanella et al., 2014). Using IoT in smart waste management will connect the end devices such as intelligent waste containers, waste collecting trucks, and the waste controller system, thus identifying the best possible truck routes and reducing the waste collection cost. Hence, IoT devices in smart waste management enable significant savings in terms of time and cost.

1.1.1.2 Smart Healthcare

Healthcare is an essential part of life as the aging population and chronic diseases increases, requiring more hospital resources, from doctors to beds. Thus, there is a requirement to reduce the pressure on the healthcare systems and provide a better quality

of care for the patients at high risk. In this situation, using IoT devices to monitor the patients remotely will reduce the pressure on the healthcare systems (Baker et al., 2017; Catarinucci et al., 2015). The IoT devices are used at the patient's, doctor's, and hospital's side for different purposes.

IoT for Patients: Using IoT devices for patient monitoring or elder monitoring transforms the healthcare system and enables a new way of delivering healthcare solutions (Park et al., 2017; Saraubon et al., 2018). The wearable IoT devices for patients will continuously collect the heart rate, blood pressure, and glucometer readings as the patient's data. It can suggest a calorie check, exercise, or a doctor's appointment.

IoT for Physician: Using IoT for patient monitoring will enable physicians to remote monitor, keep track of the patient's health effects, and proactive patient treatment.

IoT for Hospitals: The sensors and other IoT devices track the medical equipment, wheelchairs, and the medical staff's locations.

The advantages of using IoT in the smart healthcare system are (Satija et al., 2017; Dhanvijay and Patil, 2019):

- Faster disease diagnosis.
- Proactive and improved treatment.
- Drugs and equipment management.
- Low cost.

1.1.1.3 Smart Surveillance System

Nowadays, the video-based surveillance system is gaining more popularity (Kim and Ben-Othman, 2018; Tsakanikas and Dagiuklas, 2018). These systems are beneficial for government, organizations, and residential societies for monitoring safety and security activities. These surveillance systems can be used for vehicle monitoring, activity recognition, and monitoring in public places (Alam et al., 2019), student activity monitoring in the smart campus environment (Rashmi et al., 2020). The video feeds from the surveillance cameras are processed and analyzed using machine learning techniques to monitor and control the activities in real-time and thus ensure the people's safety and security in public areas, classroom environments, and residential environments (Mabrouk and Zagrouba, 2018).

1.1.1.4 Smart Manufacturing Industry (Industry 4.0)

The recent advancement in technologies has transformed the industrial environment to increase the industries' efficiency and productivity (Tao and Qi, 2017; Kang et al., 2016; Yang et al., 2019). IoT devices are used in the industrial environment to monitor and control manufacturing devices in real-time and reduce significant failures. The smart industries are also referred to as Industry 4.0, which has revolutionized the industries that focus on interconnectivity, automation, machine learning, and real-time optimization in manufacturing and supply chain management. Industry 4.0 allows manufacturers to predict the problems before they happen and thus reduce the manufacturing machines' potential failures (Tange et al., 2020; Aazam et al., 2018a). The Industrial IoT (IIoT) applications such as Pump Condition Monitoring, Pipeline leak, Corrosion detection, and Manufacturing machine monitoring are delay-sensitive and require machine monitoring in real-time to reduce significant failures (Aceto et al., 2019; Diez-Olivan et al., 2019).

1.1.2 Components of IoT Architecture

The components of IoT architecture consist of IoT sensor and actuator devices, data acquisition or data collection node, gateway devices, and the cloud data center for processing and analyzing the collected data is shown in Figure 1.2.

- **Sensors and Actuators:** The sensors are the devices that detect change in the environment. These sensor devices continuously sense and collects the data and then transfer it as a electrical signal. For example, the agriculture field sensor will collect the moisture, soil temperature, and pH level data. Actuators are the devices that receive the control signal and act on the received control signal. For example, the actuators will open or close the water pump valve after receiving the control signal based on the sensor data (Ray, 2017; Krishna et al., 2017).

Figure 1.2: Components of IoT Architecture

- **Data Acquisition System and Gateway Devices:** These are the devices deployed close to the sensor and actuator devices to collect the data from the sensor. The collected sensor data is stored in these devices and then transferred to the cloud for further processing and analysis to make intelligent decisions on the data.

- **Cloud Data Center:** The data collected from the sensor devices should be processed, analyzed to decide, and then send the control signal to actuators. Computational and storage resources are needed to process and store the data. The cloud data center provides the computational and storage resources as the services and machine learning techniques for analyzing the sensor data and makes the decisions to transfer the control signal (Aazam et al., 2016; Pham et al., 2015; Mital et al., 2015). The sensor data can be processed at the device level, at the network level using network devices such as smart gateways, routers, and remote cloud data centers are used for complex data processing and storage (Rahimi et al., 2020; Zahoor et al., 2018).

1.2 Computing Architectures

IoT devices deployed in the different smart environments generate an unprecedented amount of data. The computing resources are required to process and extract the insights from the data, which is helpful to make the decisions. There are two types of computing architecture used to process and analyze the IoT data, namely: Cloud-centric (Centralized) and Device-centric computing (Simmhan, 2017b; Varshney and Simmhan, 2017).

- **Cloud-centric Computing:** It uses centralized cloud computing architecture to process, store and send back decision to the actuators.
- **Device-centric Computing:** It uses the end devices or network devices to process the data close to the data source devices. The two types of device centric computing are Edge Computing and Fog Computing.

The Cloud, Edge, and the Fog Computing architectures and the use-cases are discussed in the following sub-sections.

1.2.1 Cloud Computing

Cloud computing is a centralized computing architecture that provides the processing, storage, and network resources as services remotely over the internet (Rimal et al., 2009; Beloglazov et al., 2011). The cloud offers on-demand services and uses the pay-as-you-go price model for charging cloud service users. The complete overview of cloud computing architecture is shown in Figure 1.3. The most crucial cloud computing characteristics are (Buyya et al., 2010; Garg et al., 2013; Buyya et al., 2009):

- **On Demand Services**: Cloud users can access and use cloud resources based on their requirements. Cloud service allows users to increase or decrease the cloud resources when they need them and charge based on a pay-as-you-go basis.

- **Rapid Elasticity:** On-demand services ensure rapid elasticity, which is essential for successful resource allocation. Rapid elasticity allows the cloud user to scale up or down the cloud resources based on the requirement.
- **Resource Pooling:** The computing, storage, memory, and network resources are pooled and allocated to multiple users based on user demand.
- **Measured Services:** The usage of cloud resources is measured and metered by the cloud service provider. The measurement is used to track resource usage and the cost of resource usage according to their demands.

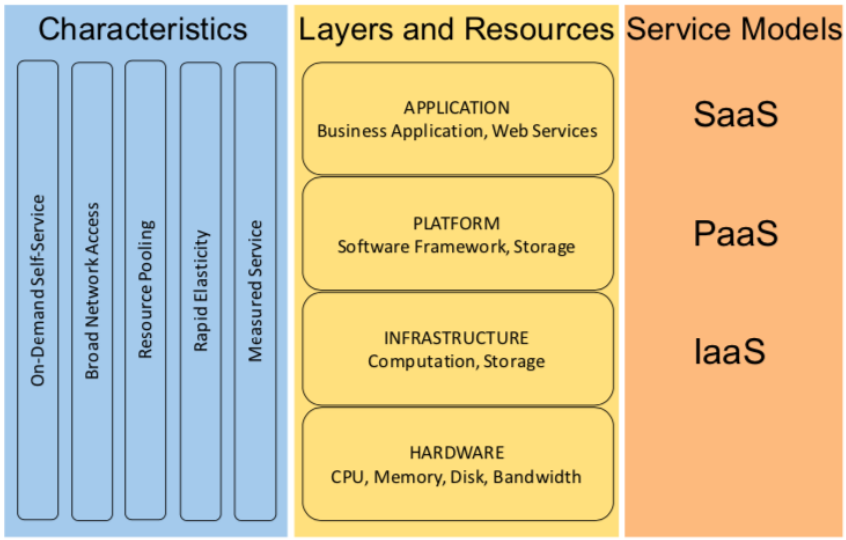


Figure 1.3: Overview of an Cloud Computing Architecture (Botta et al., 2016)

1.2.1.1 Cloud Service Models

The cloud services are classified into different models such as Infrastructure as a Service (IaaS), Software as a Service (SaaS), Platform as a Service (PaaS), and Anything as a Service (XaaS) (Buyya et al., 2010; Kavis, 2014).

Infrastructure as a Service (IaaS): In IaaS service model, the cloud service provider offers storage, processing (Virtual Machines), memory, load balancers and other computing resources on demand to the end users (Ghosh et al., 2013). It enables rapid scale-up and scale-down of the resources based on the user demand and avoids users managing the physical infrastructure such as the servers and the data center. Using IaaS, a cloud user can rent the Virtual Machines (VMs), servers, storage, network resources from the cloud service provider based on the pay-as-you-go basis. Examples of IaaS are Amazon Web Services and Open Stack.

Platform as a Service (PaaS): PaaS provides the on-demand cloud environment for developing, testing, delivering, and managing software applications. In PaaS, the

cloud service providers host the application and software development tools accessible through APIs and web portals for the cloud end users. Examples of PaaS are Google App Engine and Windows Azure.

Serverless Computing: Serverless computing is defined as the architecture and programming model where small code snippets are deployed and executed without any control over the resources (Baldini et al., 2017; Fox et al., 2017). Serverless computing allows developers to purchase the backend services on a pay-as-you-go basis and ensures the flexibility to the developers to pay for what they use. Serverless computing eliminates infrastructure management and simplifies the developer experience. The benefits of using serverless computing are:

- No Infrastructure Management: The cloud service provider automatically manages the resources and let the user focus on core business logic.
- Dynamic and Simplified Scalability: Serverless computing ensures the dynamic and simplified scale-up and scale-down of the resources based on the end-user or developer requirement.
- More Efficient Use of Resources: Using serverless computing helps to use the available computing resources efficiently and allows the resources reallocation to accelerate the innovations.

Software as a Service (SaaS): The cloud service provider provides the software services hosted on a server and makes them available for end user based on subscription. The SaaS model is used for the software liscensing over the internet (Choudhary, 2007). Hence, it allows the end-users to use software applications over the internet instead of having to install the required software on the end user machines. Examples of SaaS are Microsoft 365 and Google Apps.

Anything as a Service (XaaS): Nowadays cloud is capable of providing any services, and it is termed as the Anything as a Service. The managed service providers install the hardware on the customer premises on demand, and it is called the Hardware as a Service (HaaS) and other services include Communication as a Service (CaaS), Security as a Service (SECaaS), etc.

1.2.1.2 Cloud Deployment Models

The different cloud deployment models are: Public, Private, and Hybrid cloud (Buyya et al., 2010; Voorsluys et al., 2011).

Public Cloud: Public cloud is owned by a third party called the cloud service provider. It provides computing resources, storage resources, and network resources over the internet. Anyone can use the public cloud, and it is not dedicated to any single organization or community. One of the main challenges in the public cloud is ensuring data and service security since anybody can use cloud services.

Private Cloud: The private cloud provides the cloud resources which are explicitly dedicated to the particular organization or the community. The ownership or cloud resource management is done by the organization, third party, or a combination of these two. For example, some companies use the private cloud in their onsite data center, and the cloud service is maintained over the private network.

Hybrid Cloud: A hybrid cloud combines private and public clouds, allowing data and application sharing between them. Hybrid cloud provides greater flexibility, deployment options.

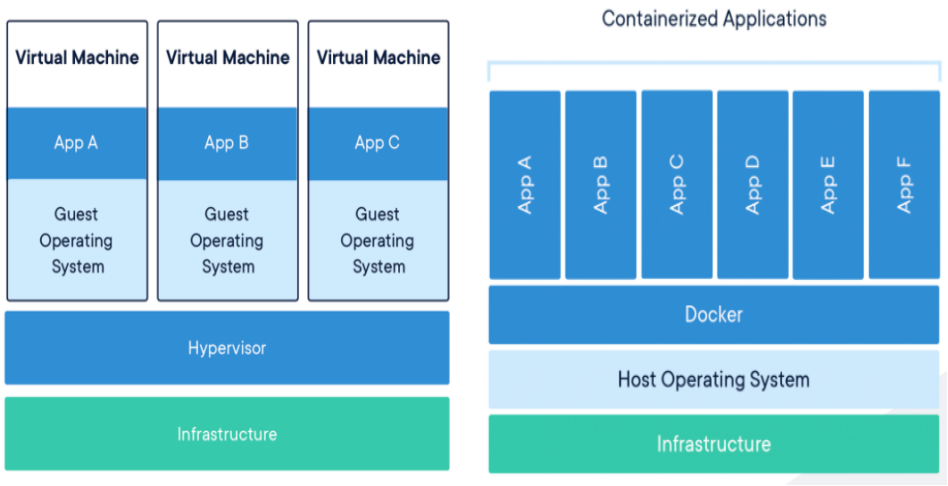
1.2.1.3 Virtualization

Virtualization is a process that allows the software to create an abstraction layer over the physical computer hardware to use the underlying computer resources effectively. It splits the existing resources such as processors, memory, and storage as multiple components. It uses an independent virtual computer called Virtual Machines (VMs), and each of these VMs runs a separate operating system (Xing and Zhan, 2012). Cloud service providers use virtualization for provisioning resources for multiple users. Some of the benefits of using virtualization are: efficient resource usage, easy management, minimal downtime, and faster provisioning.

The different types of virtualization are: full virtualization and operating system virtualization (Containers) (Sharma et al., 2016; David, 2014; Felter et al., 2015).

- **Full Virtualization:** It enables the existing Physical Machines (PMs) or servers to run many machines called VMs, and each VM contains its operating systems, applications, and necessary binaries. For deploying many VMs on the physical machines, the user must mention the amount of resources that VM should possess CPU, RAM, and storage. The managing and controlling of deployed VMs on PMs is done using the middleware called Virtual Machine Monitoring or hypervisor, which prevents the direct access of the PMs' hardware resources. One of the main problems of using VMs is, it consumes more system resources as it runs the guest operating system on each VMs and also, the booting time of VMs is very high.

- **Operating System Virtualization (Containers):** The OS-level virtualization called containers is developed to overcome the problems of using VMs. Containers are the executable software packages that contain the code and required dependencies together, which run on the host OS kernel. Container consumes less space than the VMs and provides the rapid scalability and management of IoT services (Tang et al., 2018). One of the most used container techniques is using the docker container. Docker container is the lightweight and executable package that contains the code and the required dependencies to run the application (Hoque et al., 2017). Container images are referred to as containers at runtime and in the case of Docker containers – the images become containers when they run on Docker Engine. Docker containers that run on Docker Engine are portable, more efficient, more secure and do not require a separate OS per application. Figure 1.4 illustrates the comparison between the VMs and the Containers.



(a) (b)

Figure 1.4: Comparision between VMs and Containers (Chamberlain, 2018)

These virtualization techniques can provide computing resources for applications based on user or application requests. Resource provisioning using VMs is not adaptable on resource constrained devices (fog nodes) since VMs boot time is high, consumes more resources, and hosting more VMs in a single physical machine degrades the physical machine's performance. Thus using VMs to host and process the delay-sensitive IoT/IIoT service requests might increase the service time. A container is the preferable lightweight virtualization technique which virtualizes the OS instead of underlying computer. The container not only reduces the resource management overhead but also provides rapid and high scalability (Fayos-Jordan et al., 2020) based on the application's requirement. Hence, docker containers are used for resource provisioning and process the delay-sensitive IoT applications in real-time.

Using cloud computing as the computing architecture for all types of IoT applications is not feasible. It requires entire data to be transferred to the cloud; thus, it increases communication time, service response time, consumes more computing, storage, and network resources (Yi et al., 2015; Hu et al., 2017). Hence, device-centric computing architecture is developed to provide service for delay-sensitive IoT applications in real-time. The two types of device-centric computing techniques are: Edge and Fog Computing.

1.2.2 Edge Computing

One of the best methods to address low latency and the bandwidth issues of centralized computing architecture is by pushing the computation away from the centralized cloud to the devices. Performing computation on the devices to make real-time decisions is called Edge Computing (Garcia Lopez et al., 2015; Satyanarayanan, 2017). The processing and analytics in edge computing are limited to a single device, and edge computing does not extend across different domains and the cloud. Thus using edge computing for data processing ensures low latency, reduced resource consumption, and bandwidth cost for the IoT applications (Shi et al., 2016).

The IoT applications, such as security systems in smart homes or smart buildings, smart health monitoring, autonomous cars, etc., prefer edge computing to process and make decisions quickly.

Benefits of using the Edge computing for processing the data are as follows (Shi and Dustdar, 2016; Premsankar et al., 2018):

- Security: Data accessed and analysed locally on the devices hence the number of hops of data transfer is reduced to make the data more secured and protected.
- Cost saving: Bandwidth and other resources cost is minimized.
- Rapid Scalability: Computations on the IoT devices allow quick scale-up and scale-down of their operations.

1.2.3 Mobile Edge Computing (MEC)

Mobile edge computing is defined as pushing the computation to the base stations of the mobile operator to host and process the data at the network edge. Gupta et al. (2016) defined MEC as a highly distributed computing architecture that provides the computing,

storage resources to deploy both computationally intensive and delay-sensitive applications on the base stations. Characteristics of the MEC architecture are: proximity, optimized utilization of the radio and the network resource, ensures low latency, location awareness (Hu et al., 2015). Li et al. (2016) defines the critical benefits of MEC in terms of its ability to virtualize and programmability of the network to provide the resources and host the applications.

1.2.4 Fog Computing

A centralized computing architecture was preferred to process, store, and analyze the data in the last few years or decades ago. It provides greater computational and storage resources in the cloud. Nowadays, there is a transition towards using decentralized computing architecture to address the high latency, location unawareness, high service cost, and more resource usage issues of the centralized computing environment. Hence, a decentralized computing architecture called Fog Computing is developed to move the computation close to the data source devices (Solutions, 2015; Puliafito et al., 2019; Varshney and Simmhan, 2020).

Bonomi et al. (2012) defined fog computing as a decentralized computing architecture that provides computational, storage, and network resources close to the data source devices at the edge of the network. Hence, fog and cloud computing environments share some of the standard features such as virtualization, multi-tenancy, etc. (Bonomi et al., 2014). In the fog computing environment, devices such as smart gateways, routers, and micro data centers, servers are used as the fog nodes at the edge of the network to provide computational resources to host, process, and store the IoT service requests in real-time (Abdulkareem et al., 2019; Aazam and Huh, 2014, 2015a; Chang et al., 2017).

Fog computing helps to address the following issues (Aazam et al., 2018b; Yi et al., 2015):

- To carry Machine to Machine (M2M) and Human to Machine (H2M) communication in IoT applications.
- To enable the data aggregation at the edge and push and pull the data selectively from the cloud.
- To adapt and scale upon the geographical expansion.
- To provide the data analysis capabilities for making real-time decisions on many IoT application domains.

- High latency and the location awareness.

The introduction of fog computing adds a new computation layer between the lower IoT and the Cloud layers. Hence, using fog computing nodes to process the data at the network's edge will reduce communication time and reduce the total service time and resource requirement. Figure 1.5 shows three layers of computing architecture consisting of IoT, Fog, and Cloud layers.

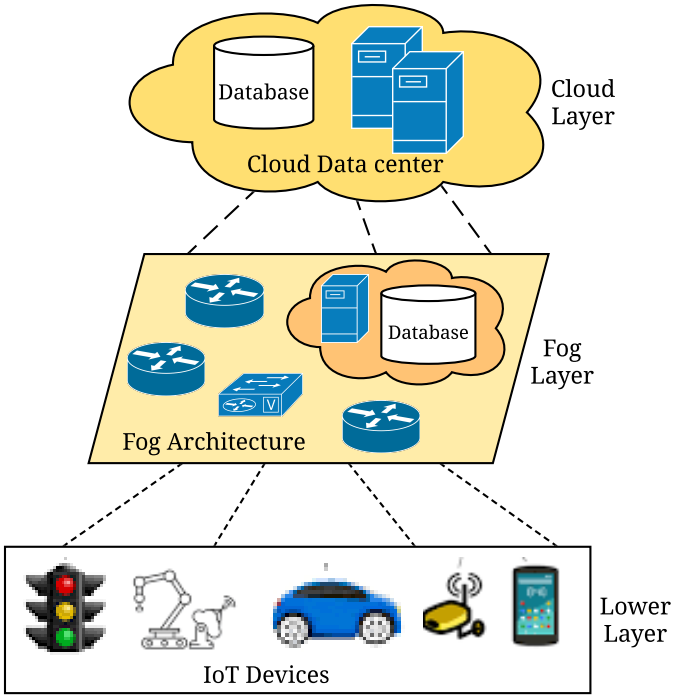


Figure 1.5: Three-Layer Computing Architecture

The lower layer consists of IoT sensor devices and actuators deployed in a different smart environment. These devices generate the data and transmit it to the fog or cloud to process. The middle layer consists of fog devices, which provide computational and transient storage resources to process and store the IoT data. Fog nodes will process and, if required, decide and send back the control signal to activate the actuators. Finally, the top layer is the cloud layer provides more considerable computational resources to process, aggregate, and store the data from the different fog nodes for further analysis (Systems, 2015).

1.2.5 Fog Computing Use-cases

Fog computing ensures quick service for the delay-sensitive applications such as Smart grid, Industry 4.0 or Smart Manufacturing Industry, Smart Healthcare (Kumari et al.,

2018), Connected Vehicles. Some of the use-cases are described as follows.

1.2.5.1 Use-case 1: Connected Cars/Autonomous Cars

Connected cars/autonomous cars have huge number of sensor devices and generate massive amount of data. These connected cars communicate with road side units or smart traffic light controller which are used as the fog nodes to process the data (Vilalta et al., 2018).

1.2.5.2 Use-case 2: Smart Home/Building

The various IoT devices are deployed to monitor and control the different activities remotely. These IoT devices generate massive data, and it is transferred to the fog nodes. The fog nodes process and analyze to make the quick decisions to send back the control signals to the actuators. Using fog computing architecture to process this data in the smart home or smart building offers more security, low latency, minimizes cost and energy consumption (Rahimi et al., 2020).

1.2.5.3 Use-case 3: Smart Grid

Smart grid has a huge number of sensor devices that are connected to provide an intelligent power supply. Using Fog computing in the smart grid addresses challenges such as latency issues, handling device failures, and ensuring network security. Fog computing allows distributed control by using location awareness, performing real-time analytics, etc., in a smart grid environment to provide uninterrupted power supply (Ruan et al., 2020). Hence, using fog computing will help to make the real-time decision in the smart grid environment (Okay and Ozdemir, 2016).

1.2.6 Challenges in Fog Computing Environment

Some of the challenges in the fog computing environment are as follows (Liu et al., 2019; Wang et al., 2015; Mahmud et al., 2018; Simmhan, 2017a):

- **Resource Management:** As these devices are resource constrained and geographically distributed, resource management is one of the main challenges in the fog computing environment. The use of virtualization techniques using VMs and containers addresses the resource provisioning and management issue in the fog computing environment.

- **Energy Management:** Fog nodes are distributed, and these are battery-operated; hence, efficient energy management is required to increase these fog nodes' reliability and availability.

- **Fault Tolerance:** The fog nodes are prone to software or hardware failures and handling these device failures, and load management is a challenge in the fog computing environment.
- **Security and Access Control:** The fog nodes operate at the network level; hence preserving the data integrity and developing the security protocol is required to protect the fog system from malicious users or nodes.
- **Applications Placement:** Using fog nodes to host and process the IoT application requests is a challenge since the fog nodes are geographically distributed and computationally less powerful.

1.2.7 Comparison of the Computing Architectures

Edge, Fog, and Cloud computing are the three computing architectures that can process the IoT data based on the applications' requirements in a smart application environment. Table 1.1 shows the comparison between three computing architectures in terms of different attributes.

Table 1.1: Comparison between Edge, Fog and Cloud Computing Architectures

Attributes	Edge Computing	Fog Computing	Cloud Computing
Number of hops	one	more than one	many
Data processing location	on same device	Micro-data centers, servers, network devices etc.	distant cloud server
Latency	low	low	high
Security	high	high	low
Processing capacity	low	medium and higher than edge devices	high
Location awareness	yes	yes	no
Mobility	yes	yes	no
Bandwidth requirement	less	less	high

The edge and fog computing architectures use the on-device and more than one hop device at the network level to process the data at the edge and the micro data centers, servers are used as the fog nodes, whereas cloud computing requires the multiple hops transfer to host the IoT data at the cloud. Hence, using cloud computing may increase the latency due to more hops of data transfer, and thus using fog and edge devices for processing the data may reduce the latency. Further, the bandwidth requirement for data transfer to the cloud computing architecture is very high compared to the one-hop and more than one hop data transfer at the edge and fog computing environment. The devices' processing capacity is less in the Edge devices. Fog nodes have typically higher processing capacity than edge devices and Cloud computing architecture provides greater computing resources. The edge devices perform computations at the edge

of the network resulting in less data transfer across the network and hence the security threat is less than the cloud. Using fog nodes the number of hops of data transfer gets reduced and hence the security threat is less than the cloud. Further, Edge and Fog computing architectures not only provide the location awareness but also handle the mobility of the devices for data processing. But, cloud computing does not provide location awareness and the mobility of the devices as it is centralized computing architecture (Solutions, 2015; Kim, 2016).

1.3 Motivation

Many applications are delay sensitive in different smart environments such as smart healthcare monitoring, Smart industry/Industry 4.0 environment, Smart/Autonomous cars, etc. Using a centralized cloud as the computing architecture is not feasible as it might increase the service time and the service cost (Dastjerdi and Buyya, 2016; Yi et al., 2015). Hence, fog computing is developed to address latency, bandwidth, cost, and resource consumption issues (Bonomi et al., 2012). Using the resource constrained fog nodes to deploy the IoT services will reduce the service time, cloud resource consumption and thus delay sensitive IoT services can be placed on the fog nodes to ensure the QoS. One way to create the fog computing environment is to use the virtualization technique on resource constrained devices. There are several existing works using simulations and the deployment of VMs on the edge devices or fog nodes to host and process the IoT data (Salman et al., 2018; Peng et al., 2016). Using VMs on resource constrained devices might not be feasible as it consumes more host machine resources, boot time is high. Hence, using the lightweight OS-level virtualization technique will reduce start-up/booting time, PMs resource consumption and further enhances the scalability and management of the IoT services. Thus, using containers over VMs in the fog computing environment will help to address the latency-sensitive applications (Hoque et al., 2017; Tang et al., 2018). Therefore, using the fog computing environment minimizes the communication time and processes the delay-sensitive IoT/IIoT applications in real-time. Further, the challenge is using resource constrained devices to develop the fog framework for real-time monitoring to analyze the sensor data and make quick decisions in smart environments.

With this motivation, a novel QoS-aware service placement strategy is developed to place the IoT/IIoT service requests in the Fog-Cloud environment to ensure the QoS of the application in terms of deadline, cost and resource availability. A container-based two-level resource provisioning fog framework is developed to host and process the

IoT/IIoT applications requests. The service placement strategies are proposed to place the service requests on the developed fog framework to minimize service time, service cost, and energy consumption in the fog computing environment. Pushing intelligence close to the data source devices will reduce the service time and thus enable a real-time monitoring in the smart environments. This motivated to develop the fog server-based framework as a prototype to deploy the machine learning models on the fog server to analyze the various sensor data and enable real-time monitoring. The developed prototype considers the intelligent malfunctioning machine monitoring system for classifying the machines as normal and abnormal machines based on the analysis of machines' operating sounds and thus enable real-time monitoring in the smart industry/Industry 4.0 environment.

1.4 Organization of Thesis

Figure 1.6 shows the detailed organization of thesis with the formulated research objective and the respective research contribution chapters. The thesis is organized as follows.

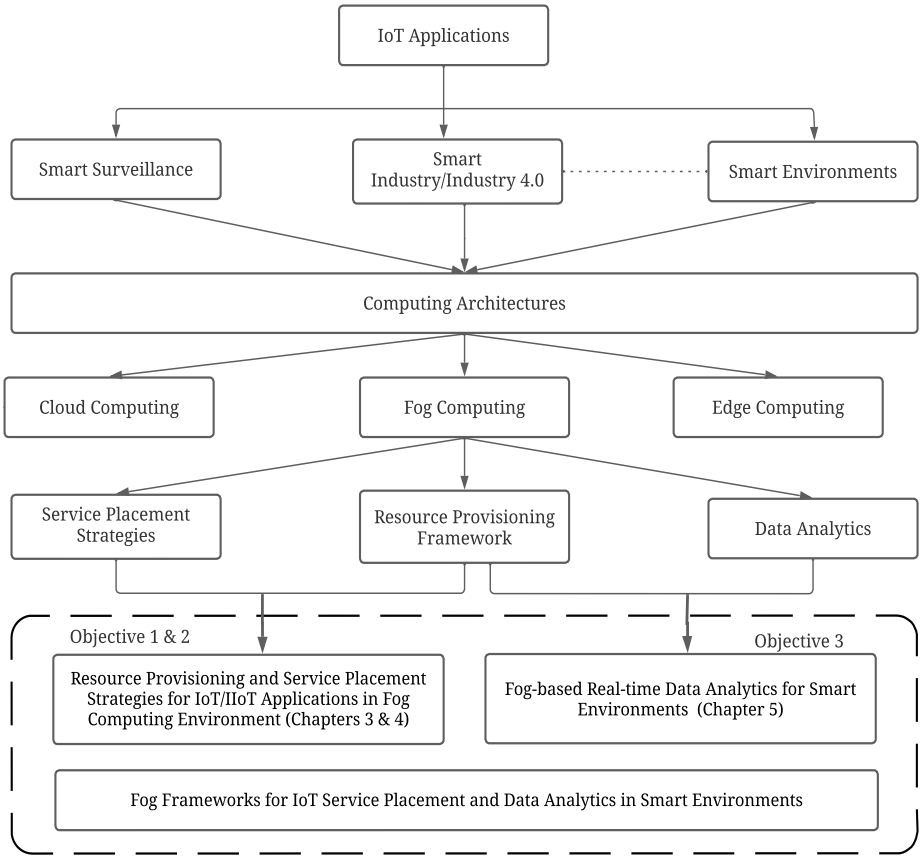


Figure 1.6: Thesis Organization with respect to Research Objectives and the Contributions

Chapter 2 presents the state-of-the-art resource provisioning frameworks, the service placement strategies on the fog and cloud computing architectures, data analytics, and the monitoring systems in smart environments. Further, the outcomes of the literature survey are discussed to find out the research gaps in the fog and cloud computing environments. Finally, the problem statement and research objectives of the work are given in detail.

In Chapter 3, a detailed explanation about the QoS-aware service placement strategies to place the IoT/IIoT services in the Fog-Cloud computing environment and the performance evaluation of the service placement strategies are discussed in detail.

Chapter 4 describes the development of a container-based two-level resource provisioning fog framework using the resource constrained 1.4 GHz processor devices. Further, the meta-heuristics based service placement strategies to place IIoT service requests on the developed fog framework to minimize service time, service cost, and the fog node energy consumption are discussed in detail.

Chapter 5 discusses the cost-efficient fog server-based framework for real-time data analytics in smart environments. The fog server-based prototype is developed to deploy the supervised machine learning techniques to analyze data in smart environments. An intelligent industrial machine monitoring system is considered for identifying the normal and abnormal machines based on the analysis of the machines' operating sounds in the Industry 4.0/Smart Industrial Environment.

Finally, Chapter 6 presents the concluding remarks based on the thesis contributions with the future directions in the fog computing environment.

1.5 Summary

In this chapter, a detailed explanation for the Internet of Things, IoT applications is presented. The basic concepts of the computing architectures used to process the IoT data, such as Cloud Computing, Edge Computing, and Fog Computing, are discussed. The possible use-cases and the comparison of these computing architectures with respect to the different attributes are given in detail. Further, the motivation for doing this research and the detailed organization of the thesis is presented in this chapter. The detailed literature survey, identified research gaps, the problem statement, and the objectives are discussed in the next chapter.

Chapter 2

Literature Survey

Nowadays IoT has gained a lot of attention in every aspect of day-to-day life, and it is used everywhere for remote and automated monitoring. These IoT devices generate a massive amount of data; thus, efficient computing architectures are essential to provide the computational resources to process and analyze data.

In this chapter, the existing works on the resource provisioning framework and service placement strategies in the Fog-Cloud environment, and the data analytics in smart environments such as smart surveillance, smart industry (Industry 4.0) applications are discussed in detail.

2.1 Resource Provisioning Frameworks and Service Placement Strategies in Fog and Cloud Computing Environments

Using the centralized cloud as a computing architecture to process the data may experience more delay. Hence, the flexible indie-fog infrastructure is developed by Chang et al. (2017) using the customer premises equipment as fog nodes to provide the services for the IoT applications. Thus using wi-fi access points, routers, and other network devices as fog nodes not only provides the computational resources for servicing the IoT applications in real-time but also reduces the delay. Aazam et al. (2018b) designed the layered architecture for the placement of fog infrastructure between the existing Cloud-IoT infrastructure and thus provides the computational resources to handle the urgent tasks and then offload them to the cloud for further processing. Lee et al. (2016) developed a gateway and micro server-based fog infrastructure for Wireless Sensor and Actuator Networks. The developed architecture consists of a master layer that handles control functionalities. The worker layer handles resource management, the flow management based on the Software Defined Networking (SDN), and the virtual gateway at the lower level of the architecture. The new simulation environment called FogDirSim is developed for simulating the actual management policies using the RESTFul APIs for industrial fog computing management platform (Forti et al., 2020). The developed management platform can decide when and where to migrate specific services and reduce the fog node's energy consumption. But, they did not consider the container-based multi-level fog computing framework and the multi-objective optimization for IoT/IIoT service placement in the fog computing environment.

IoTSim-Osmosis framework is developed to support osmotic computing (Alwasel et al., 2020). In osmotic computing, the tasks/services are migrated from the centralized cloud to the edge devices or IoT devices based on events. The developed simulation framework evaluates the smart electrical billing application and assesses the execution time, network transmission time, and energy consumption. Further, simulations were carried out for enabling the edge and cloud framework but did not consider the container-based fog framework. Aazam and Huh (2015a) developed a micro data center (MDC) based fog resource management model for efficient and effective resource management. The resource estimation for the customers is done by relinquishing the customer service type and price to provide the resources for processing the IoT applications. Battula et al. (2020) developed a stochastic model for selecting the resources and behavior in single and multi-location fog nodes using the Markov chain. Further, a nearest location best-fit strategy is proposed for resource selection by considering all the fog characteristics. However, the current work did not consider the container-based fog framework for IoT service placement.

A hierarchical and secure fog to cloud (mF2C) continuum architecture is developed for effectively utilizing the computing resources at the different architecture levels. The developed architecture can host the IoT applications requests to process, analyze the data, and ensure the resource coordination between the hierarchies (Masip-Bruin et al., 2018). A fog development kit (FDK) is developed to create a fog environment for computing and network resource allocation and service placement on the fog nodes (Powell et al., 2020). The developed kit uses the SDN and virtualization techniques for allocating resources. Further, the FDK is integrated with the fog emulators to create a flexible fog environment to build the application prototypes with less additional cost. Luo et al. (2019) developed a container-based multi-cloud to multi-fog architecture. The temporary and long-term service model is designed to reduce the IoT applications' service delay for efficient fog resource utilization. Further, they developed an energy-aware task scheduler to improve network nodes' lifetime and the tasks' delay constraints in the fog computing environment.

Zhang et al. (2017) developed a cooperative fog computing-based architecture for the Internet of Vehicle (IoV) applications for handling the big data from the smart city applications. The developed architecture for IoV consists of an edge and fog layer and supports mobility, multi-source data acquisition, multi-path data transmission, distributed data computation, and storage. Further, they considered the hierarchical

model for resource management between the Inter-fog and Intra-fog. The SDN-based edge framework is developed for efficient resource allocation in the vehicular networks (Goudarzi et al., 2020). The reinforcement learning algorithm is designed for handling the resource allocation problem at the edge using cloud assistance. The computation and network resource allocation in the developed three-tier architecture is done using the experience reply information.

A smart gateway-based communication between the Cloud-IoT infrastructure is developed to enable data processing at the network level to decide what to send to the cloud for further processing (Aazam and Huh, 2014). Thus, using the smart gateway-based fog infrastructure at the network level will optimize network and cloud resource usage, power consumption. The autonomic generic computing model is developed by Etemadi et al. (2020) to scale up or scale down the fog resources using the Bayesian learning technique. The developed model will provide the resources to accommodate IoT services' workload in the fog computing environment. A micro cluster-based fog computing framework is developed for collaborative task execution among the devices by sharing the incentive information to form a microcluster (Luo et al., 2020). The developed framework is used to host and process the delay-sensitive or energy-intensive services to optimize the service time and the devices' energy consumption.

The Ubiquitous Resource Management for Interference and Latency-Aware (URMILA) middleware solution is developed to manage the resources across the edge, fog, and cloud for handling mobility and minimize the service level objective violations of the delay-sensitive IoT applications (Shekhar et al., 2019). The developed middleware is responsible for choosing adequate resources based on deterministic user mobility patterns. Further, a fog server selection algorithm is developed based on the user's mobility and evaluated for network latency and cost. Peralta et al. (2020) developed a flexible and dynamic architecture by combining the fog and cloud architecture to reduce data download time at the end node or the fog nodes. An M/M/1 queuing system is modeled at the multi-level computing architecture for reducing the service delay at the fog node and uses network coding to enable the distributed data storage. But, they did not consider ensuring the QoS in the fog and cloud computing environment.

The different service placement strategies are developed to place the IoT service /application modules on the fog and cloud computing environments. Taneja and Davy (2017) proposed resource-aware-based service placement in the fog and cloud computing environment to place the IoT application modules using the Benchmark Lower

Bound (BLB) algorithm. The application module placement using BLB is done based on the resource availability to effectively utilize the fog resources at the network level to minimize the network usage of the IoT applications. Desikan et al. (2017) developed a distributed latency-aware data processing model in the fog computing environment. The gateway devices share the available resource information with other gateways and then decide data forwarding for processing in the fog layer. Therefore using fog-enabled gateway devices not only reduces the transmission latency but also reduces the response time. But, they did not consider the multi-objective-based service placement in the fog computing environment.

Mishra et al. (2018) developed the nature-inspired meta-heuristic algorithms for industrial applications placement in the heterogeneous fog computing environment. The distinct VMs are considered the server to place the service requests to optimize the execution time and the energy consumption in the fog environment. Skarlat et al. (2017) developed a heuristic algorithm to solve the fog service placement problem. The service requests are placed on the fog node based on resource availability, thus minimizing the service cost and satisfying the IoT applications' deadline. Liu et al. (2020) proposed a heuristic-based Horae task scheduling algorithm in the Mobile Edge Computing (MEC) environment for scheduling task requests on the edge server for efficient resource utilization and further satisfy the edge server placement constraints in the MEC. The developed heuristic algorithm considers minimizing the processor slack timing and meets the processor topologies and placement constraints of the MEC environment. But, they did not consider the multi-level fog computing architecture to service the IoT applications.

Hassan et al. (2020) developed a heuristic approach for IoT service placement based on the network priority, energy consumption in the Fog-Cloud computing environment. The proposed method considers the IoT applications with a maximum of two services placed on Fog-Cloud architecture. Souza et al. (2018) developed service placement strategies in the Fog-Cloud computing scenarios based on the service specification and resource availability. The services are atomized and then placed on the Fog-Cloud computing resources for parallel execution using Best-fit and Best-fit with Queue methods in the Fog-Cloud computing environment. Murtaza et al. (2020) developed an adaptive and intelligent task scheduling method in the Fog-Cloud environment to improve the QoS in terms of delay and energy consumption. A smart layer is introduced between the IoT and cloud to process the data using the learning-based policies. Further, the ser-

vice cost and available resources in the fog computing environment are not considered for servicing the IoT applications. Deng et al. (2016) considered the framework consisting of the Fog-Cloud computing devices for optimal placement of the workload such that the power consumption and the delay for the IoT applications are minimized. The mathematical model is developed for optimal workload allocation among the devices in the Fog-Cloud environment. Further, the problem is decomposed into subproblems and then solved using the approximation method to get the optimal solutions. However, the current work did not consider multi-objective optimization in the fog computing environment.

Bozorgchenani et al. (2020) developed a mathematical model for energy consumption and energy harvesting in the fog computing environment. A prediction-based energy harvesting approach is proposed for selecting the fog nodes in the cluster to reduce fog node energy consumption and thus enhance the fog nodes' lifetime. Yang et al. (2018) developed the energy-efficient homogeneous fog framework by exploiting the user devices. Further, carried out a mathematical analysis to develop a maximal energy-efficient task scheduling algorithm in the homogeneous fog network to optimize the energy efficiency. Goudarzi et al. (2020) developed a weighted cost model to optimize the execution time and the energy consumption in the considered Edge-Fog-Cloud computing environment. Further, a concurrent IoT application placement using the memetic-based algorithm and the pre-scheduling algorithm is developed to increase the number of parallel task execution in the Edge-Fog-Cloud computing environment.

Mahmud et al. (2018) proposed a latency-aware application module placement in the fog computing environment for satisfying the service deadline and further considered application module forwarding to optimize the number of active fog nodes that violates the QoS of the applications in the fog computing environment. Ghanavati et al. (2020) formulated the application placement problem as a bi-objective optimization model by considering both makespan and energy consumption. Further, proposed Ant-Mating Optimization algorithm for optimal task placement on the fog nodes minimizes the makespan and the energy consumption in the fog computing environment. Mahmoud et al. (2018) proposed an energy-aware-based application placement in the fog computing environment. Further, smart healthcare-based task/service placement is done based on the fog nodes' resource availability and energy consumption to process the health sensor data in real-time. However, the existing works did not consider the service cost and deadline in the two-level fog computing architecture.

Table 2.1: Summary of Key Existing Works on Resource Provisioning Framework

Sl. No.	Authors	Framework	Remarks
1	Tuli et al. (2019)	FogBus work as PaaS Model	FogBus framework integrates both fog and cloud for resource monitoring, application deployment to process the IoT data in real-time.
2	Merlino et al. (2014)	Stack4Things-PaaS based Fog Framework	The fog framework is developed for deploying multiple service requests.
3	Concone et al. (2019)	Fog Framework	A three-tier (Sensor-Fog-Cloud) architecture is developed for human activity reco -gnition. The user devices such as smartwatch and smartphone are used as the fog node for initial preprocessing of the sensor data, and activity models are maintained at the cloud.
4	Ananthanarayanan et al. (2017)	Rocket:The software stack for video analytics	A multi-tier architecture is developed for video analytics using the cloud, private clusters, and the edge devices to address the service time, bandwidth issues for real-time video analytics.
5	Patman et al. (2018)	Fog Framework for Predictive Analytics	A fog sandbox in a GENI infrastructure is developed for predictive analytics using machine learning techniques. GENI provides heterogeneous computing resources to process computationally intensive tasks. The predictive analysis using the GENI fog sandbox is crucial for delay-sensitive applications.
6	Isa et al. (2020)	Energy Efficient Fog Framework	Gigabit Passive Optical access network-based fog framework is developed for health monitoring. Further, MILP based mathematical model is formulated for optimizing the number and the location of the fog nodes/servers at the network edge to optimize the energy consumption.

Table 2.2: Summary of Key Existing Works on Service Placement Strategies in Fog and Cloud Computing Environment

Sl. No.	**Authors**	**Methodology**	**Objective**	**Environment**	**Testbed**	**Remarks**
1	Ramírez et al. (2017)	First-Fit and Random Fit based service placement	Single Objective	Fog-Cloud	No	Used first-fit and random fit algorithms for the dynamic service placement in the continuous Fog-Cloud computing environment based on the devices' resource availability.
2	Xavier et al. (2020)	Heuristic based resource allocation	Single Objective	Cloud of Things	No	The proposed approach considers the heterogeneity of the devices at the edge and the cloud environment for efficient resource allocation using collaboration between the edge and cloud.
3	Naas et al. (2017)	Divide and conquer method for IoT data placement	Single	Fog	No	Designed a generalized iFogstor data placement strategy for fog architecture on the geographical zone to reduce the end to end latency. The network usage and nature of data is considered to store and process on the fog nodes.
4	Zhang et al. (2018)	Adaptive bacterial foraging algorithm	Multi-Objective	SDN	No	A multi-objective optimization problem is formulated for controller placement in the SDN to optimize network reliability and load balancing. Further, solved the problem using an adaptive bacterial foraging algorithm.
5	Yu et al. (2020)	Dependency-aware Task Placement	Single	Fog	No	Task allocation problem is formulated as a non-linear integer optimization problem and then fragmented into subproblems to allocate on both the IoT and the fog nodes to optimize the network lifetime.
6	Ruan et al. (2018)	Dynamic resource allocation	Single Objective	Fog	No	Developed a three-layer fog architecture and then designed a dynamic resource allocation and up-link offloading strategy using the differential game and bipartite graph multiple matching method.
7	Djemai et al. (2019)	Discrete-PSO based service placement	Single	Fog	No	Developed meta-heuristic based discrete-PSO algorithm for IoT service placement in the fog computing environment for optimizing the energy consumption of the devices.

Tables 2.1 and 2.2 summarize the key existing works on the Resource Provisioning Framework and Service Placement Strategies, respectively. Most of these existing works considered simulations and deployed VMs on the devices to develop the fog frameworks, but these are not efficient as they might take more time for booting and consume more PMs resources. Also, the existing works on service placement strategies in the fog and cloud computing environment were evaluated using simulators or used VMs based resource provisioning frameworks. This motivated to consider the containers for developing the multi-level resource provisioning fog framework on resource constrained devices to host and process the IoT/IIoT service requests. Hence, using containers on the network edge or resource constrained devices to create the fog framework will enable servicing of delay-sensitive IoT applications in the fog computing environment. Further, developing the multi-objective optimization-based service placement and other meta-heuristic-based hybrid algorithms is essential to optimize the fog nodes' service time, cost, and energy consumption for servicing the IIoT applications in the fog computing environment. As these fog nodes are battery operated and reducing energy consumption will increase the fog nodes' reliability and availability, optimizing the service time will avoid significant industrial environment failures.

2.2 Data Analytics in the Fog Computing Environment

With an increase in the number of IoT devices in smart cities or IIoT devices in the smart manufacturing environment, there is an exponential increase in data generated. Using distributed computing architecture to process and analyze the data at the network level will address the latency issues in the smart city/Industry 4.0 environment. Some of the existing works on data analytics in the fog computing architecture are outlined in this section.

Tang et al. (2017) developed a hierarchical distributed fog architecture using different devices and integrated the intelligence into the architecture to analyze latency-sensitive and latency tolerant applications data. The sequential learning method is developed and deployed on the hierarchical fog architecture for anomaly detection and avoids the smart pipeline monitoring system's critical failures. Liu et al. (2020) developed an intelligent system for numerical control machine tools monitoring and data processing system for the smart manufacturing industry. The developed framework considers the bi-directional data and control flow between the machine and the software system used for analyzing the machine data. Li et al. (2018) proposed a fog computing-based DeepIns system for defect detection and inspection in the smart manufacturing

environment. The developed models are deployed on the fog nodes for real-time defect and the degree of defect detection in the images captured from the sensor devices in the manufacturing environments.

Tuli et al. (2020) proposed a deep learning ensemble-based edge system architecture called HealthFog for real-time heart disease analysis. The proposed system is deployed on the FogBus (Tuli et al., 2019) framework for evaluating the system's performance in terms of execution time, jitter, power consumption, and the accuracy in the fog computing environment for processing the heart patients data considered as the user requests. A Fog-IBDIS system is developed to integrate and share the industrial data with the fog nodes to ensure the security of the data, also reduces the network traffic load by using the fog architecture and thus reducing the network resource consumption (Wang et al., 2019). The data analysis is done using the task flow graph, and each graph has the modules which are integrated with the fog architecture. The developed Fog-IBDIS system can use the edge devices' to preprocess the industrial data, and other more extensive analytics is performed at the cloud server.

He et al. (2017) developed a multi-tier fog computing architecture using both ad-hoc and the dedicated fog nodes to perform the data analytics for smart city applications. Further, QoS-based resource allocation strategies are developed to provide computational resources for real-time data analytics services. Yassine et al. (2019) proposed a new platform for IoT data analytics using fog and cloud resources for smart home applications. The proposed system is used to address the fog and cloud computing environment's issues, such as computing resources, storage resource requirements for the online and offline data processing, task allocation, and classification analysis. Further, the system is used for IoT device management and admission authentication in the smart home environment.

Chen et al. (2016) developed a dynamic video surveillance system for real-time car detection and tracking using fog computing nodes. In the developed system divide and conquer strategy is used for processing the video. The moving vehicle detection algorithm is deployed on the fog node to detect the vehicle. The video frames of the detected vehicle are used for tracking the vehicle. Thus utilizing the fog nodes to process the video will reduce the service time and the network load of real-time video surveillance systems. Diro and Chilamkurti (2018) developed a deep learning model for a distributed attack detection scheme for the IoT environment. The deep learning models are trained using the fog nodes and implemented on edge for attack detection.

The parallel and distributed network attack scheme with a centralized attack scheme is considered to compare the performance of the developed model. Further, compared the developed model with the traditional ML approaches.

Foukalas (2020) developed a cogni-IoT platform for providing distributed intelligence using fog computing for IIoT applications. The developed platform is used for the machines' predictive maintenance by deploying machine learning models on the devices. Thus, it intelligently monitors the machine conditions based on the sensor data in the smart factory environment. EdgeEye, an edge computing-based framework, is developed for real-time video analytics (Liu et al., 2018). The developed framework provides the deep neural network models for processing and analyzing the videos on the edge devices, reducing computation and bandwidth resources. Khochare et al. (2017) developed a distributed video analytics system using the Orchestration Platform for Hybrid Dataflows across Edge and Cloud (ECHO) platform (Ravindra et al., 2017). The ECHO platform is used on the Edge, Fog, and Cloud resources to deploy the deep learning model for real-time automated parking billing and the urban scene classification in a smart city environment.

Fog-based smart gateway prototype is developed for data conditioning, data filtering, and performing data analytics on sensor data from wearable devices (Constant et al., 2017). The developed smart gateway is also responsible for deciding what data to be transferred to the cloud for further analysis and long-term storage. Further, the developed prototype can be used for smart data analytics in the health environment for analyzing the wearable sensor data in real-time. Zeng et al. (2020) developed a distributed framework using the cameras and the edge cluster architecture for dynamic workload management. The developed model considers the workload balancing among the cameras and the workload partition between the cameras and the edge clusters for efficient usage of the edge resources. Hence, it maximizes the throughput, reduces the latency without affecting the accuracy of the video analytics methods.

Table 2.3 shows the summary of key existing works on data analytics in the fog computing environment. Many of these current works used fog and cloud to perform the video data analysis, sensor data analysis in urban video surveillance, industrial machine monitoring applications based on vibrations, and other sensor data. But, it is also essential to develop an intelligent machines' monitoring system based on the operating sounds to detect and classify the industrial machines as normal and abnormal (malfunctioning) using the fog computing architecture.

Table 2.3: Summary of Key Existing Works on Data Analytics in the Fog Computing Environment

Sl. No.	**Authors**	**Methodology**	**Data**	**Environment**	**Remarks**
1	Iqbal et al. (2019)	Fault Detecion and Isolation using Deep learning approach	Control and inspection sensor data	–	Used a deep learning model for fault detection and isolation and thus automated the manufacturing machine monitoring in industrial environment. But, did not consider machine sounds data to detect the malfunctioning machines.
2	Saucedo-Dorantes et al. (2020)	Novel fault detection using incremental learning	Machine Vibrations Data	–	Used linear discriminant analysis and the self-organizing map for the novel fault detection and classification in the industrial environment. The machines operating sound for machine fault detection is not considered.
3	Mahmud et al. (2019)	Context-aware application placement for Industry 4.0 oriented applications	Images	Fog	The robot assistance-based monitoring is considered in the industrial environment. The robot will process the images from the IP cameras to extract the features of the emergency events and assist in deciding real-time to avoid the significant failures in the industrial environment.
4	Tuli et al. (2019)	EdgeLense framework for real-time object detection	Videos	Fog- Cloud	Developed EdgeLense framework which utilizes the available computational resources at the edge to deploy the deep learning model (YOLO) for real-time object detection. Thus optimize the cost, response time, bandwidth requirement.
5	Neto et al. (2018)	Realtime Crime Object Detection	Video	Fog	A fog based crime assistance prototype is developed for crime object detection in the smart transportation system.

Table 2.3: Summary of Key Existing Works on Data Analytics in the Fog Computing Environment

Sl. No.	**Authors**	**Methodology**	**Data**	**Environment**	**Remarks**
6	Wang et al. (2019)	CNN based Sailent Object Detection	Images	Fog- Cloud	SaliencyGAN, an adversarial learning framework is developed to deploy the CNN models on the fog-cloud environment for the salient object detection.
7	Teerapittayanon et al. (2017)	Distributed deep neural network on the distributed Fog-Cloud architecture	Sensor Data	Fog-Cloud	Deployed DNN models on the distributed Fog-Cloud computing resources to process and analyze the data at the network edge level. The developed framework minimizes the communication and computational resource usage which allows low latency.
8	Yang et al. (2019)	Multi-layer ocean of things framework for marine environment information monitoring	Multi-sensor Data	Fog-Cloud	A fog layer between the cloud and the sensor is introduced in the marine environment to process the multi-sensor data at the fog layer to make the real-time decision, and the neural network-based model is used in the cloud for further data analysis.
9	Ashjaei and Bengtsson (2017)	Predictive Maintenance Framework for Smart Factory	Sensor	Fog-Cloud	A multi-level framework consisting of Fog-Cloud is developed for predictive maintenance in the smart factory environment. The developed framework is more reliable and ensures data privacy and security, and also it reduces response time.
10	Liang et al. (2019)	Prognosis system for machine processing optimization using CNN	Sensor	Fog-Cloud	A dynamic prognosis system is developed consisting of Fog-Cloud for machine processing optimization. The system uses the CNN model to analyze the machine data at the fog level and thus reduce the data traffic and improve the system energy efficiency.

2.3 Outcome of Literature Survey

The extensive literature survey of the existing works on the different resource provisioning frameworks and the service placement strategies on the Fog-Cloud computing environment is done in section 2.1. Using fog nodes as the computing resources to perform the data analytics in the smart city, the smart industrial environment is done in the section 2.2 and thus identified some of the following research gaps in the fog and cloud computing environment to carry out the research work.

Research Gaps

1. **Resource Provisioning Framework and IoT/IIoT Service Placement**
    - Developing heuristic or meta-heuristic approaches for IoT/IIoT application placement in fog and fog-cloud architecture is essential to ensure the applications' QoS.
    - Most of the existing works used VM-based virtualization or simulators to simulate resource provisioning and service placement strategies. Hence, using Docker and containerization techniques at the edge/fog facilitates faster resources and faster deployment of applications/services.
    - Many of the authors proposed single objective or bi-objective optimization techniques for IoT application placement. Hence, there is a need to develop multi-objective optimization-based service placement strategies on the Multi-level fog infrastructure for placing the IoT/IIoT applications.
    - There is a need for learning-based workload prediction and allocation in the fog computing environment to achieve low latency and efficient utilization of the fog computing resources.

2. **Data Analytics in the Fog Computing Environment**
    - Some of the works considered using the cloud environment to deploy the machine learning and deep learning techniques to analyze the data. Using fog nodes for deploying the machine learning models on the resource constrained fog nodes to analyze IIoT data in real-time is helpful for the delay-sensitive IoT/IIoT applications.
    - Many of the existing works considered sensor-based machine monitoring to detect the defects in a smart factory environment. There is a need for Intelligent Machine monitoring using the machines' operating sound to monitor the malfunctioning machine in the Smart Manufacturing Industry (Industry 4.0) environment.

3. **Service Migration and Load Balancing on the Fog Nodes**
    - There is a need for handling the fog nodes failure in the fog computing environment.
    - There is a need for handling the migration of tasks or services in the fog environment for load balancing due to unavailability of resources or node failure in the fog computing environment. Hence, there is a requirement for handling the runtime service migration between the fog nodes if any anomaly is predicted in the fog computing environment.

2.3.1 Problem Statement

This research aims to design and develop fog frameworks for IoT/IIoT Service Placement and Data analytics for smart environments. Accordingly, the research problem and objectives are defined as follows.

Problem Statement:

"To Design and Develop the Fog Frameworks for IoT/IIoT Service Placement and Data Analytics in Smart Application Environments"

2.3.2 Research Objectives

1. Design and develop the QoS-aware Service Placement Strategies in Fog-Cloud Computing Environment.
2. Design and develop a container-based Framework and QoS-aware Service Placement strategies in the Fog Computing Environment.
3. Design and develop a Cost-effective Fog server framework for Real-time Data Analytics in Smart Application Environments.

2.4 Summary

In this chapter, the existing computing architectures/frameworks developed for resource provisioning to host and process IoT service requests in the fog-cloud and fog computing environments are presented in detail. Further, the state-of-the-art service placement strategies were discussed to place the IoT/IIoT service requests in the fog-cloud computing environment. Also, the existing works on data analytics for smart environments are discussed in detail.

Finally, the challenging issues based on the outcomes of the literature survey, the problem statement, and the research objectives are discussed in detail. The QoS-aware Service Placement Strategies in the Fog-Cloud computing environment are explained in the next chapter.

Chapter 3

QoS-aware IoT/IIoT Service Placement in Fog-Cloud Environment

Cloud computing provides scalable resources to process and store the data. Using cloud computing for delay-sensitive IoT applications is not preferable as it requires entire data to be transferred to the cloud for processing, which will lead to network congestion and increase communication time. Hence, the service delay and the service cost are high as the cloud resources are billed based on the pay-as-you-go approach (Buyya et al., 2009). Therefore, using Fog-Cloud computing architecture will provide the service in real-time by hosting a few service in the fog computing environment (Bonomi et al., 2012).

Using both Fog and Cloud computing to process the data might reduce the network congestion and the service time. But, the placement of the services on the fog nodes is the key challenge because of the geographical distribution and the limited computing capabilities of the fog nodes. The node that can host the service requests should satisfy the Quality of Service (QoS) in terms of service cost, deadline, and resource requests. This motivated to conisder the available resources at the network edge to host and process the IoT services to minimize the cloud resource usage and thus reduce the network resource usage, service cost and energy consumption for the IoT services. Hence better placement strategies are required to place the service requests on the fog nodes to minimize the service delay, service cost, and energy consumption. This chapter discusses service placement strategies for the Fog-Cloud Computing environment. The First Fit Decreasing (FFD) algorithm (Baker, 1985) is applied for service placement in the Fog-Cloud computing environment based on the resource availability on the fog nodes to minimize the service delay and network resource usage. Further, a novel cost-efficient deadline aware service placement approach is proposed to place the services in the Fog-Cloud computing environment and thus ensures the QoS in terms of deadline and the service cost of the IoT services in the Fog-Cloud environment.

3.1 Three-Tier Architecture for Servicing IoT/IIoT Applications

The three-tier computing architecture with IoT, Fog, and Cloud layers is considered for processing the IoT data. Fog computing is used as the intermediate computing architecture between the IoT and Cloud layers, as shown in Figure 1.5 (Page No. 12, Chapter 1).

Table 3.1: List of Notations

Notations	Description
V_M	set of cloud servers
S_K	set of all services
F_N	set of all fog nodes
R	total number of computing resources
t^f_{pro}	processing time in fog node in ms
t^c_{pro}	processing time in cloud server in ms
t^f_{com}	communication time in fog include both request and response in ms
t^c_{com}	communication time in cloud including both request and response in ms
t^f_{av}	service availability time in fog nodes in ms
t^c_{av}	service availability time in cloud server in ms
T_t	total service time in ms
D_d	service deadline defined for the IoT applications in ms
C^f_{pro}	processing cost in $ for service **s** deployed in fog node
C^c_{pro}	processing cost in $ for service **s** deployed in cloud server
C^f_{sto}	storage cost in $ for service **s** deployed in fog node
C^c_{sto}	storage cost in $ for service **s** deployed in cloud server
C^{CPU}_n	unit cost in $ for processing resource in mips at fog node
C^{CPU}_m	unit cost in $ for processing resource in mips at cloud server
C^{sto}_n	unit cost in $ for storage resource in mips at fog node (per byte per second)
C^{sto}_m	unit cost in $ for stoarge resource at cloud server (per byte per second)
C^s_{max}	maximum service cost in $
S^f_{cost}	service cost for the applications deployed on the fog nodes in $
S^c_{cost}	service cost for the applications deployed on the cloud server in $
S_{cost}	total service cost in $
s^{size}	service size
γ_j	processing capability of a fog node j
β_f	bandwidth between nodes in Mbps
γ_c	processing capability of a cloud server
$\Gamma_{(n,s)}$	service availability time in fog nodes in ms
β_c	bandwidth between cloud server and IoT in Mbps
λ_{sn}	input rate/ arrival rate of service requests on fog node
λ_{sm}	input rate/ arrival rate of service requests on cloud server
L^{CPU}_a	required amount of processing elements (in mips)
L^{sto}_a	storage size of service in bytes
$L^{e_i}_s(t)$	total amount of data generated by end devices (in bytes)
E_{max}	maximum energy consumption in joules
E^{Total}_{eng}	total energy consumption of fog-cloud environment in joules
E^{eng}_{fj}	energy consumption of fog node j in joules
E^{eng}_{ft}	energy consumption during transmission in fog in joules
E^{eng}_{fp}	energy consumption during processing in fog in joules
E^{eng}_{ct}	energy consumption during transmission in cloud server in joules
E^{eng}_{cp}	energy consumption during processing in cloud server in joules
P^{idle}_f	power consumption of fog node in idle state in watts
P^{idle}_c	power consumption of cloud server in idle state in watts
P_{trans}	maximum power consumption during transmission in watts
P_{proc}	maximum power consumption during processing in watts
$time$	total time duration in ms

Tier 1 (IoT Layer) contains the end devices such as sensor nodes, camera devices, and other IoT devices that continuously sense and generate the data. Tier 2 (Fog Layer) consists of a router, smart gateways as fog nodes, and independent Micro Data Center (MDC) as fog servers (Simmhan, 2017a). These devices can use VMs or container techniques to provide the computational resources at the network's edge and host IoT applications. These devices are used to process the data and then forward data to the cloud, thus reducing the network congestion by reducing data size. Tier 3 (Cloud Layer) is a centralized cloud data center that provides substantial processing and storage resources. A Cloud data center performs computationally intensive operations and stores the data for a longer duration.

3.2 Service Placement Problem Formulation

Table 3.1 shows the list of mathematical notations used in this work. The service placement problem in the Fog-Cloud computing environment is considered as the NP-hard problem (Brogi et al., 2020; He et al., 2018). The service (S_K) placement on the available computing resources (R) is considered as mapping each service requests on the fog and the cloud resources as given by Equation (3.1).

$$S_K \rightarrow R \tag{3.1}$$

The available computing resources in Fog-Cloud environment is given by R=$\{F_N \cup V_M\}$, where F_N is the number of fog nodes and V_M is the number of cloud server in the three-tier architecture. The set of all services to be deployed on the computing devices is denoted by $S_K = \{s_1, s_2, ...s_k\}$.

3.2.1 Resource Constraints in Fog-Cloud

The computing resources considered for deploying the IoT service requests (s$\in S_K$) are: fog nodes (n$\in F_N$) and the cloud server (m$\in V_M$) in the Fog-Cloud computing environment.

The deployment of IoT service requests on fog nodes should consider the available resources on the resource constrained fog nodes such that the fog nodes are not overloaded. Thus, the resource requirement of service should be less than the fog nodes'

available computing resources in terms of CPU (in millions of instructions per second (MIPS)), RAM and storage capacity of the fog nodes defined by the Equation (3.2).

$$\forall f_j \in F_N \begin{cases} \sum_{\forall s_i}^{S_K} CPU^{req}(s_i) * x_{ij} \leq CPU_{available}(f_j) \\ \sum_{\forall s_i}^{S_K} RAM^{req}(s_i) * x_{ij} \leq RAM_{available}(f_j) \\ \sum_{\forall s_i}^{S_K} storage^{req}(s_i) * x_{ij} \leq storage_{available}(f_j) \end{cases}$$

$$\begin{aligned} &where, \\ &\quad s_i \in S_K, \;\; i \in [1, K] \;\; services \\ &\quad f_j \in F_N, \;\; j \in [1, N] \;\; fog\ nodes \\ &\quad v_j \in V_M, \;\; j \in [1, M] \;\; cloud\ servers \end{aligned} \tag{3.2}$$

$$x_{ij} = \begin{cases} 1, & \text{if } s_i \text{ is placed on fog node } f_j \\ 0, & \text{if } s_i \text{ is placed on cloud server } v_j \end{cases} \tag{3.3}$$

The deployment of IoT applications on the fog node f_j and the cloud server v_j is indicated by x_{ij} given by Equation (3.3).

3.2.2 Service Time and Energy Consumption in Fog-Cloud

The IoT applications, such as smart healthcare applications, autonomous cars, and industrial applications, are delay-sensitive. Thus, using the fog nodes at the network level will reduce the total service time. The delay-sensitive applications are deployed on the fog node and delay-tolerant on the cloud server. The master-worker IoT application model is considered, consisting of the independent modules to be placed on the Fog-Cloud environment such that the service deadline of the application is satisfied. The total service time T_t depends on the processing time, service availability time, and communication time. Equation (3.4) defines the total service time (T_t) for the IoT applications in the Fog-Cloud computing environment.

$$T_t = \left(((t^f_{pro} + t^f_{av} + t^f_{com}) * x_{ij}) + ((t^c_{pro} + t^c_{av} + t^c_{com}) * (1 - x_{ij}))\right) \tag{3.4}$$

The processing time (t^f_{pro}) in the fog node F_N and the processing time (t^c_{pro}) in cloud server V_M depends on the processing capacity of the VMs on which the service requests are deployed, and the communication time (t^f_{com}, t^c_{com}) depends on the data transfer rate between the IoT devices and the computing devices (fog node and cloud server). Communication time consists of both service request time and the response time from the devices. Service availability time (t^f_{av}, t^c_{av}) in the Fog-Cloud environment for the applications/services is defined as the total waiting time to prepare data and computing resources and waiting time in the queue to get the computing resources in the Fog-Cloud environment. Thus waiting time of the service requests depends on the processing time and the arrival time of the IoT applications. The placement of IoT application requests on the fog nodes should satisfy the deadline of the IoT application as given by Equation (3.5).

$$T_t \leq D_d \tag{3.5}$$

The total energy consumption (E^{Total}_{eng}) in the Fog-Cloud computing environment depends on the energy consumed by the fog and cloud devices. The energy consumption is calculated during the processing and communication in both fog and cloud computing environments as defined by Equation (3.6). The energy consumption in the fog and cloud devices for the deployed services is given by Equations (3.7)-(3.10).

$$E^{Total}_{eng} = \left((E^{eng}_{ft} + E^{eng}_{fp}) * x_{ij} + (E^{eng}_{ct} + E^{eng}_{cp}) * (1 - x_{ij})\right) \tag{3.6}$$

The energy consumption during processing (E^{eng}_{ft}) in the fog node (Equation (3.7)) depends on the total power consumption over the period of time for transferring (P_{trans}) the data and the power consumption when the fog node is idle (P^{idle}_f).

$$E^{eng}_{ft} = \sum_{t=0}^{time} \left(P^{idle}_f + \left(\frac{S^{size}_i}{\beta_f} * P_{trans} \right) * x_{ij} \right) \tag{3.7}$$

The energy consumption during processing E^{eng}_{fp} in the fog node (Equation (3.8)) depends on the total power consumption over the period of time for processing service

requests and the power consumption when the fog node is idle.

$$E_{fp}^{eng} = \sum_{t=0}^{time} \left(P_f^{idle} + \left(\frac{S_i^{size}}{\gamma_f} * P_{proc} \right) * x_{ij} \right) \tag{3.8}$$

Similarly, energy consumption in the cloud environment is calculated using the Equations (3.9)-(3.10). The energy consumption in the cloud environment (E_{ct}^{eng}) depends on the power consumption of the cloud devices over the time duration of processing and communication.

$$E_{ct}^{eng} = \sum_{t=0}^{time} \left(P_c^{idle} + \left(\frac{S_i^{size}}{\beta_c} * P_{trans} \right) * (1 - x_{ij}) \right) \tag{3.9}$$

$$E_{cp}^{eng} = \sum_{t=0}^{time} \left(P_c^{idle} + \left(\frac{S_i^{size}}{\gamma_c} * P_{proc} \right) * (1 - x_{ij}) \right) \tag{3.10}$$

3.2.3 Service Cost in Fog-Cloud Environment

The total service cost for the IoT applications deployed in the Fog-Cloud environment is given as follows. The total service cost is defined as the sum of the processing and storage cost. It depends on the number of resources utilized for providing the service for the deployed IoT applications in the fog and cloud environment. Equations (3.11) and (3.12) give the processing cost of both the fog (C_{pro}^{f}) and cloud computing resources, respectively. The processing cost of the IoT applications in both fog and cloud computing environments (C_{pro}^{c}) depends on the amount of CPU resources used, and the unit cost for the processing elements in both fog (C_n^{CPU}) and cloud (C_m^{CPU}) environments is given by Equations (3.11) and (3.12), respectively.

$$C_{pro}^{f} = \sum_{i=1}^{K} \sum_{j=1}^{N} (C_n^{CPU} * L_a^{CPU} * \lambda_{an} * x_{ij}) \tag{3.11}$$

$$C_{pro}^{c} = \sum_{i=1}^{K} \sum_{j=1}^{M} (C_m^{CPU} * L_a^{CPU} * \lambda_{am} * (1 - x_{ij})) \tag{3.12}$$

Equations (3.13) and (3.14) define the storage cost in the fog and cloud environments, respectively. The storage cost for the application requests on fog (C^f_{sto}) and cloud (C^c_{sto}) depends on the amount of storage resources utilized and the unit cost for the storage resources in both the fog (C^{sto}_n) and cloud (C^{sto}_m) environments for all service requests as given by Equations (3.13) and (3.14), respectively.

$$C^f_{sto} = \sum_{i=1}^{K} \sum_{j=1}^{N} \left(C^{sto}_n * L^{sto}_a * x_{ij} \right) \tag{3.13}$$

$$C^c_{sto} = \sum_{i=1}^{K} \sum_{j=1}^{M} \left(C^{sto}_m * L^{sto}_a * (1 - x_{ij}) \right) \tag{3.14}$$

The service cost for the IoT applications deployed in the Fog (S^f_{cost}) and Cloud (S^c_{cost}) computing environments is given by the Equations (3.15) and (3.16), respectively. The service cost is defined as sum of the processing and storage cost in both the fog and cloud computing environment.

$$S^f_{cost} = (C^f_{pro} + C^f_{sto}) \tag{3.15}$$

$$S^c_{cost} = (C^c_{pro} + C^c_{sto}) \tag{3.16}$$

3.2.4 Optimization Model for Service Placement

The master-worker IoT application model is considered, which consists of modules that will process the service requests independently and then send back the response to each request. Each of these application modules is regarded as an independent service request. Thus, minimizing the service time of these applications by deploying in the Fog-Cloud computing environment minimizes service cost by reducing the cloud resource usage. The total service cost for the applications deployed in the Fog-Cloud computing environment is given by the Equation (3.17).

$$S_{cost} = S^f_{cost} + S^c_{cost} \tag{3.17}$$

IoT/IIoT service placement problem in Fog-Cloud computing environment is formulated as the multi-objective optimization problem. The optimization model is defined for minimizing the service cost and energy consumption for the IoT/IIoT services deployed in the Fog-Cloud computing environment such that it ensures the QoS of the IoT/IIoT services given by Equation (3.18). The total service cost for the IoT/IIoT applications depends on the processing cost and storage cost in the Fog-Cloud computing environment. Hence, considered minimizing the total service cost and energy consumption of the IoT/IIoT services in the Fog-Cloud computing environment should satisfy the defined constraints. The service time should be less than or equal to the deadline for the services defined by Equation (3.18), and the resource requests should satisfy the resources available with the fog nodes and the cloud to place the services in the Fog-Cloud computing environment as defined by the Equation (3.2).

$$
\begin{aligned}
&Minimize \sum_{i=1}^{K} \sum_{j=1}^{R} \; S_{cost}(s_i, R_j) \\
&Minimize \sum_{i=1}^{K} \sum_{j=1}^{R} \; E_{eng}^{Total}(s_i, R_j) \\
&\quad Subject \;\; to: \\
&\qquad T_t \leq D_d \\
&\qquad 0 \leq E_{eng}^{Total} \leq E_{max} \\
&where, \\
&\quad s_i \in S_K, \;\; i \in [1, K] \;\; services \\
&\quad \forall j \in R, where \;\; R = \{F_N \cup V_M\}
\end{aligned}
\tag{3.18}
$$

The constraints defined for the objective functions are as follows:

- Total service time for the IoT/IIoT service should be less than the service deadline defined for the deployed services in the Fog-Cloud environment.
- The energy consumption should be less than the maximum energy consumed in the Fog-Cloud computing environment.
- Each service request for the resources and placement decision should satisfy the resource requirement defined by Equation (3.2). The service should also be placed only on one computing node in the Fog-Cloud computing environment.

3.3 Service Placement Strategies in the Fog-Cloud Environment

In this section, the existing First-Fit Decreasing (FFD) algorithm is applied for service placement in a fog-cloud environment and the details are shown in subsection 3.3.1. Further, a new deadline aware based service placement (DASP) algorithm is proposed to place services in the Fog-Cloud computing environment and the details are given in subsection 3.3.2.

3.3.1 Strategy 1: FFD based Service Placement

The placement of services on the fog nodes and the cloud is referred to as the bin packing problem (Salaht et al., 2019). But the main challenge is to find the eligible node which can host and provide the service. The placement of these services on the fog nodes should consider the nodes distributed over geographical locations with limited computing capability so that the service delay is minimal. Hence, a heuristic-based First-Fit Decreasing (FFD) algorithm (Baker, 1985) is applied to place the services on the fog nodes while considering its available resources. This work on the applied FFD algorithm for service placement in fog environment is based on our publication (Natesha and Guddeti, 2018).

The deployment of services $s \in S_K$ on the fog node should satisfy the resources required of the applications in the fog environment. The application requests which meet the resources requirement are deployed on the fog nodes at the network edge. First-Fit Decreasing (FFD) approach is applied to deploy services on the distributed fog node at Tier 2 of the architecture based on the availability of the resources of the fog nodes. Using FFD, services are deployed on the fog nodes, and thus it reduces the number of services to be transferred to the cloud. Hence, using the fog nodes to process the data reduces communication time, thus minimizing the service delay and the network resource usage for IoT applications.

3.3.2 Strategy 2: Deadline Aware Service Placement

The IoT service placement in the Fog-Cloud environment is referred to as the NP-hard problem (Brogi et al., 2020; He et al., 2018). But the primary challenge is to find a suitable fog node that can host the services and thus provide the service in real-time. Hence, we propose a novel cost-efficient Deadline Aware Service Placement (DASP)

algorithm to place the service requests in the Fog-Cloud computing environment. Using the DASP algorithm, the service time and the service cost and energy consumption are minimized, thus ensuring the QoS of IoT applications in terms of deadline and resources availability. Figure 3.1 shows the service allocation procedure for deploying the IoT service requests in the Fog-Cloud computing environment.

Algorithm 3.1 Deadline Aware Service Placement

Input: Services s $\in S_K$
Resource_list include both Fog (F_N) and Cloud (V_M) resources
Result: Placement_list $\leftarrow$ []

Intialize: Placement_list$\leftarrow$ []
for *each service* $s \in S_K$ **do**
  for *each set of all Resources in fog nodes* $n \in F_N$ **do**
    // check for resource availability constraints in fog nodes n$\in$ F_N
    if(Resources_Availability(s,n))
      $T_{total} \leftarrow$ ServiceTime(s,n) using Equation (3.4)
    $S^f_{cost} \leftarrow$ ServiceCost(s,n) using Equation (3.15)
      if($T_t \leq D_d$ and $S^f_{cost} \leq C^s_{max}$)
    // Update the placement list; place service s $\in S_K$ on fog node n $\in F_N$
    Placement_list$\leftarrow (s, n)$
    break

    else
      // Place IoT service s on Cloud server
    $T_t \leftarrow$ ServiceTime(s,m) using Equation (3.4)
    $S^c_{cost} \leftarrow$ ServiceCost(s,m) using Equation (3.16)
    // Update the placement list; place service s on cloud server m $\in V_M$
    Placement_list$\leftarrow (s, m)$
  end
end
update Services and Resources_list
return Placement_list

For each service requests, Algorithm 3.1 checks for the resources demand and the availability of resources in fog nodes. If the availability of the resources is satisfied for the service requirements in the fog nodes, then the expected service completion time and the service cost are calculated based on the amount of resources used as per given cost parameters defined by Equations (3.11)-(3.14). If the total expected service completion time and service cost are less than the defined service deadline and the maximum service

cost for each service, then the service *'s'* can be placed on the fog node n. If resources are not available with the fog nodes, such requests could be forwarded to the cloud server m, and the service cost for such application is calculated. The delay-sensitive applications can be serviced in real-time using fog computing nodes, thus minimizing the number of applications deployed in the cloud, leading to less resource consumption in the cloud server. Hence, using the Fog-Cloud computing architecture to place the delay-sensitive applications that satisfy the resource constraints and service deadline minimizes the service cost, service time for the IoT applications.

3.3.2.1 Mathematical Analysis and Example for DASP Algorithm in the Fog-Cloud

The mathematical analysis for the proposed approach is done by considering one example scenario (Liu et al., 2017; Kherraf et al., 2019). The three-tier architecture is considered for servicing the IoT applications, as shown in Figure 1.5 (Page No. 12, Chapter 1). The middle layer consists of distributed network devices that can process data and send control signals back to the actuators. Transferring the huge data from end devices to the cloud consumes more time. Thus using fog nodes at the network level will reduce the transmission/communication time, network traffic, service time, and service cost by reducing cloud resource consumption.

Let, $L_s^{e_i}(t)$ be the total amount of data generated by the sensor and other end devices, which is to be stored and processed on the computing nodes. The expected service time in the fog environment is defined as the sum of the service availability time, processing time, and communication time defined by Equation (3.4). Service availability in a fog environment depends on the waiting time for the service in fog nodes. The waiting time for the services given by Equation (3.19) depends on the service request arrival time and the fog node's processing speed. The service time should be greater than the arrival time to maintain the queue's stability at the fog node given by the Equation (3.20).

$$\Gamma_{(n,s)} = \delta_{(n,s)} - \lambda_{sn} \tag{3.19}$$

$$\lambda_{sn} \geq 0 \tag{3.20}$$

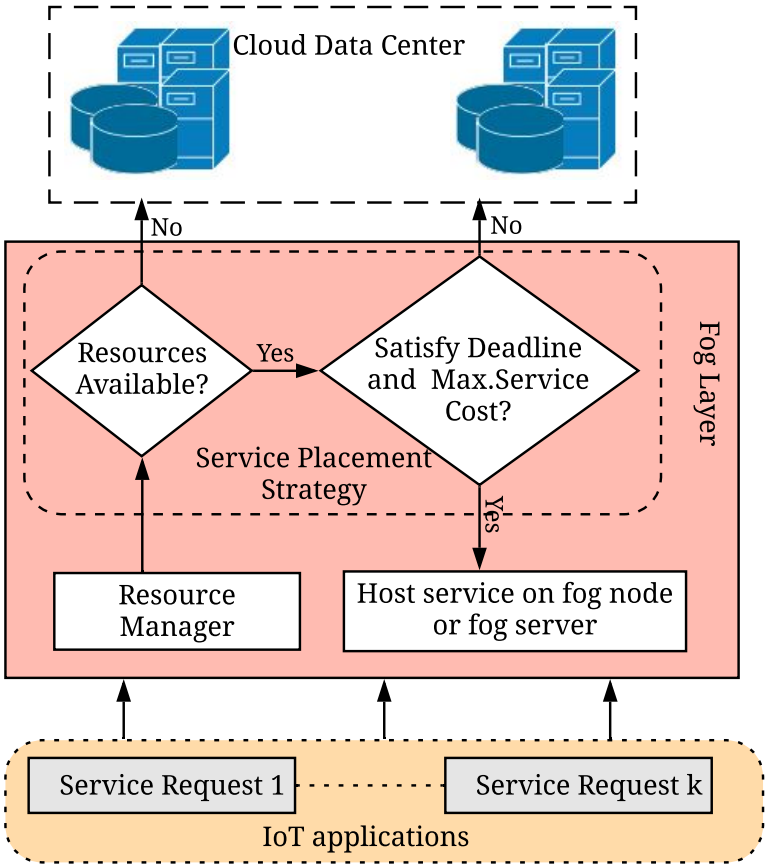


Figure 3.1: Service Allocation in the Fog-Cloud Computing Environment

If any of these service constraints in Equations (3.5) and (3.17) fail, and if resources are not available, then the service requests can be forwarded to the centralized cloud. The service cost for the application deployed in the cloud environment depends on the resources used to process the deployed applications. The service cost should be less than the maximum service cost defined by Equation (3.17).

The architecture considered for the placement of two service requests with two fog gateways and the public cloud is shown in Figure 3.2. The resources are available in the computing nodes described as R={CPU, RAM, Storage}, and the task requests are represented by the set T={CPU, RAM, Size, Deadline, Max.Cost}. The resources available in cloud and fog gateways (G1 and G2) considered for this example are given by Cloud={10000, 4GB, 10000}, G1={4000, 1GB, 3000} and G2={5000, 1GB, 2000}. The application with two tasks, T1 and T2, each task resource requests given as T1={3000, 512MB, 2000, 750ms, $250} and T2={5000, 2GB, 3000, 850ms,

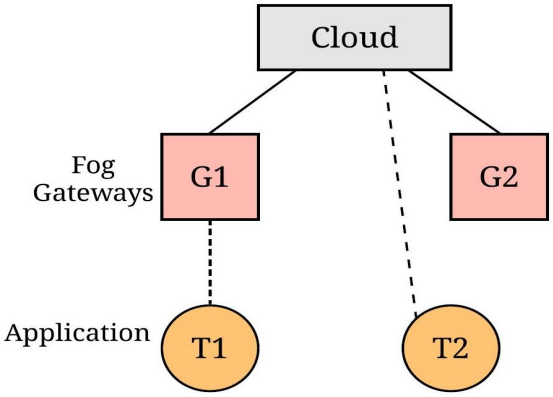


Figure 3.2: Example for Service Placement in the Fog-Cloud Environment

$700} is considered in the example. Further, we can assume that the fog and cloud environments have the communication time of 50 ms and 100 ms, respectively. The decision to allocate the tasks on the fog node is based on the availability of the resources, satisfying the service deadline and service cost. Otherwise, the task request is placed on the cloud data center, which provides substantial computational resources to host the applications.

Table 3.2: Calculation Results

	Fog		Cloud	
	T_t	S_{cost}	T_t	S_{cost}
T1	716ms	$210	767ms	$ 410
T2	-	-	700ms	$650

The expected service time and the service cost of the application, which satisfies the resource requirement in the fog computing environment, are defined by the Equation (3.2). From the considered example, task T1 requirements are satisfied by the gateway G1 and then calculates the expected service time and the service cost for deploying the service on G1 and found that it satisfies the service deadline and cost constraints. Hence, task T1 is deployed on the gateway G1, and a similar procedure is followed for task T2 which will be placed in the cloud environment. Table 3.2 shows the obtained results, and it is observed that Task T1 can be placed on either G1 or cloud but, task T1 has more delay and cost if it is placed in the cloud when compared to using fog gateway. Hence, deploy task T1 on G1 and task T2 hosted in the cloud environment since enough resources for T2 requests are unavailable in fog gateways G1 and G2. The values of T_t and S_{cost} for T1 and T2 are shown in Table 3.2.

3.4 Performance Evaluation

3.4.1 Experimental Simulation Setup

The proposed DASP algorithm is evaluated using the iFogSim simulator (Gupta et al., 2017), which is built on top of the cloudsim (Calheiros et al., 2011). The iFogSim simulator uses the VMs deployed on the data center as the resource provisioning framework. The proposed service placement methodology is implemented by extending the ModuleMapping and ModulePlacement classes. The applications and distribution of workload are made using the Application class and FogDevice class in the simulator. The workload is distributed using a uniform distribution with varying resource requirements. The fog and cloud resources to deploy the IoT/IIoT service requests; the fog nodes are resource constrained and have limited computational and storage resources compared to the cloud servers. Hence, Alibaba cloud resource configurations such as small VMs as the fog nodes and the larger VMs (Resources, 2022) as the cloud server and their pricing model are considered in the simulation as shown in the Table 3.3.

Table 3.3: Simulation Parameters

Computing Nodes and Task	Resource Types	Values
Fog Nodes	RAM	2 GB
	CPU (in MIPS)	2000-6000
	Storage	1-2 GB
	unit cost/ CPU	$ 0.05
	unit cost/ Storage	$ 0.02
Cloud Server	RAM	4 GB
	CPU (in MIPS)	8000-20000
	Storage	3-8 GB
	unit cost/ CPU	$ 0.10
	unit cost/ Storage	$ 0.05
Services	RAM	1-3 GB
	CPU (in MIPS)	2000-15000
	Storage	1-5 GB

The five different configurations, namely: Config 1, Config 2, Config 3, Config 4, and Config 5, with the number of fog nodes as 200, 400, 600, 800, and 1000, respectively, and five cloud servers are considered to deploy the service requests using the service placement algorithms. The number of fog nodes in Tier 2 of the architecture depends on the amount of work done and the fog nodes' resource availability. The data movement can happen between the fog nodes horizontally in the same tier or vertically

between the fog nodes and the cloud. The fog nodes capable of hosting the service will process the data, send control signals to actuators, and send the processed data to the centralized cloud for further analysis. Each gateway device is connected to the end devices, which collect data from the sensor and then forward it to the gateway devices. The computational capabilities of these computing nodes are heterogeneous, and the resource capabilities and cost parameters of these computing devices considered for our simulation are given in Table 3.3.

3.4.2 Application Types for Simulation

The IoT application model is represented by a directed graph, which consists of independent modules. Each module will perform some operations, and these modules should be deployed in the Fog-Cloud computing environment to minimize service time. In the simulation, the master-worker (Mahmud and Buyya, 2019) model is considered, which independently requests the services. Some of the services are delay-sensitive, and a few more are delay tolerant. The model considers the IoT data from the smart building environment used to monitor the building. The MainModule is placed on the end devices, and worker modules are placed on the fog nodes such that the QoS of applications is satisfied in the Fog-Cloud computing environment. Figure 3.3 shows the application types considered for the simulation.

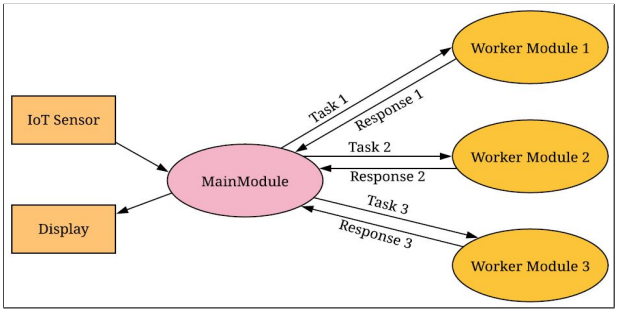


Figure 3.3: Application types considered for simulation

3.4.3 Results and Discussion

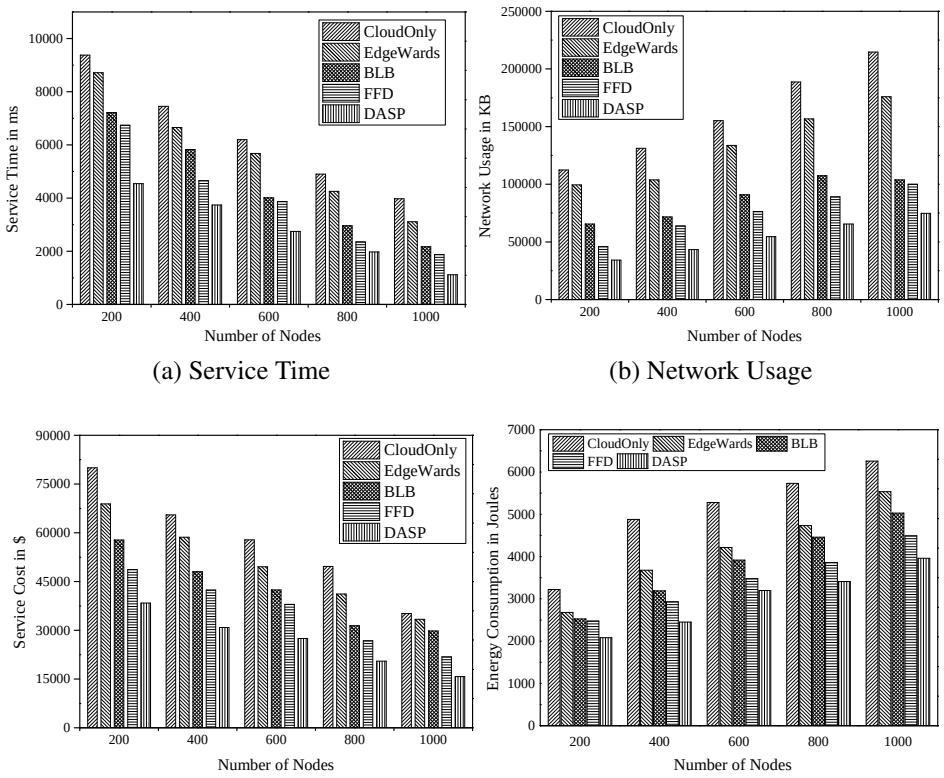
The different network topology configurations are simulated by considering two computing environments: Fog-Cloud and CloudOnly. In the Fog-Cloud computing environment, IoT services are distributed among the fog node and cloud server based on resource availability, service deadline, and cost. In the CloudOnly approach, all service requests are placed on the centralized data centers. All the data is transferred to a cloud

server, where it is processed and sends the control signal to the actuators if any actions are required. These two computing approaches are used for comparing the application service time, service cost, service deadline and the energy consumption. Further, the network resource usage for the IoT/IIoT services is measured by using the different network configurations in Tier 2 of the computing architecture. The five different network topology configurations are considered by varying the number of fog nodes in Tier 2 from 200 to 1000. The number of IoT/IIoT service is varied from 250 to 1500 for performance evaluation of the proposed DASP algorithm in the Fog-Cloud and CloudOnly computing environments.

The proposed DASP is compared with the different service placement methods, such as placing all the modules in the cloud, i.e., CloudOnly, EdgeWards, a default placement method in iFogSim simulator (Gupta et al., 2017) and state-of-the-art Benchmark Lower Bound (BLB) (Taneja and Davy, 2017) algorithm for service placement in the Fog-Cloud computing environment. The EdgeWards approach is the default service placement strategy is used to deploy the services as close to the edge of the network. It considers traversal from the leaf node to the top in the different tiers. BLB deploys the services if resources are available on the network edge and the cloud resources. The BLB approach uses the key-value pair to identify the mapping of service requests on the fog and cloud resources based on resource availability. The simulations are carried out five times for the different configurations of fog nodes in Tier 2 of the architecture and the other service placement strategies considered for performance evaluation in both the Fog-Cloud and CloudOnly computing environments. The average of the simulation results is shown in Figure 3.4.

The 1500 IoT/IIoT service requests are deployed on the different topology configurations using the various service placement strategies. Figure 3.4 (a) shows service time for the deployed services in both CloudOnly and Fog-Cloud computing environments. It is observed from the Figure 3.4 (a) that if all the services are placed in a cloud data center, it needs more time to provide the service due to an increase in the communication time for transferring a large amount of data.

In the Fog-Cloud environment, the simulations are carried out for the default service placement method (EdgeWards), BLB, FFD, and proposed DASP algorithm for ser-



(a) Service Time (b) Network Usage

(c) Service Cost (d) Energy Consumption

Figure 3.4: Performance comparison of DASP with state-of-the-art service placement strategies in terms of (a) Service Time (b) Network Usage (c) Service Cost (d) Energy Consumption in the Fog-Cloud computing environment

vices placement on the iFogSim simulator. Using Fog computing to host applications will reduce the service time. It is observed from Figure 3.4 (a) that the proposed DASP method performs better than CloudOnly, EdgeWards, BLB, and FFD based placement methods in terms of service time. Hence using fog computing for delay-sensitive IoT applications such as Healthcare monitoring, Industrial applications will provide the service very quickly. Figure 3.4 (b) indicates a significant decrease in network usage for the proposed DASP approach when compared to default EdgeWards, BLB, FFD, and CloudOnly service placement approaches considered for the performance evaluation. Figure 3.4 (c) shows service cost for applications in two different computing environments for the proposed and state-of-the-art service placement strategies. It is found that using fog computing nodes to deploy the services will reduce the service cost by reducing the size of data to be transferred and thus minimize the cloud resource consumption. Figure 3.4 (d) shows the energy consumption for the IoT services that are deployed on the Fog-Cloud computing environment. It is observed that using DASP reduces the

energy consumption in the different topology configurations as compared to the state-of-the-art service placement strategies in the Fog-Cloud computing environment.

The 250-1500 number of IoT application requests are deployed on various network topology configurations using the proposed DASP method in the Fog-Cloud computing environment. Further, the number of IoT applications that satisfy the service deadline were calculated. Some of the IoT/IIoT application services are delay-sensitive and should complete the processing within the deadline, and a few application services are delay tolerant. It is observed that using more fog nodes between the IoT and cloud architecture can satisfy the maximum number of application services' deadline.

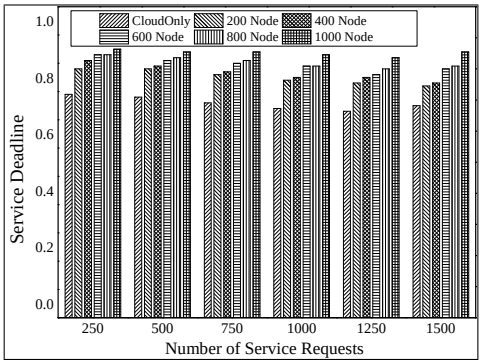


Figure 3.5: Performance evaluation of proposed DASP in terms of Service Deadline in the Fog-Cloud computing environment

Figure 3.5 shows that the percentage of services that meet the service deadline for different topology configurations using fog and cloud allocations. Using fog nodes to host delay-sensitive IoT applications will satisfy the deadline and reduce network, processing, and storage resource usage in the cloud servers.

The use of fog nodes for hosting services will reduce the amount of resources consumed for processing the applications in the cloud and reduce the service cost for the applications. The different applications are considered with the varying resources and deployed on two different computing architectures such as Fog-Cloud and CloudOnly. In Fog-Cloud, used different topology configurations to deploy the services on the available computing resources. Using fog nodes to deploy the services that satisfy the resources requirement and service deadline will provide the service in real-time. Hence, it minimizes the number of service requests deployed in the cloud environment and reduces the service cost by lowering cloud resource usage.

3.4.4 Statistical Hypothesis Analysis

Table 3.4: p-values for t-test analysis

Algorithms	Configurations	p-Values
CloudOnly-DASP	Config 1	0.021
	Config 2	0.011
	Config 3	0.015
	Config 4	0.017
	Config 5	0.019
EdgeWards-DASP	Config 1	0.019
	Config 2	0.012
	Config 3	0.021
	Config 4	0.017
	Config 5	0.019
BLB-DASP	Config 1	0.027
	Config 2	0.029
	Config 3	0.031
	Config 4	0.033
	Config 5	0.026
FFD -DASP	Config 1	0.031
	Config 2	0.023
	Config 3	0.034
	Config 4	0.021
	Config 5	0.024

The statistical hypothesis testing is carried out for the proposed DASP algorithm with the existing benchmark methods for the service placement in the Fog-Cloud computing environment. The t-test analysis is considered to evaluate the service placement strategies in Fog-Cloud environment. The t-test is carried out for the proposed DASP method, and the different benchmark methods (CloudOnly, EdgeWards, BLB, and FFD) for service placement and results of the t-test analysis are shown in Table 3.4. For the null hypothesis, we considered the threshold value of p or the significance level denoted by α. The widely adopted arbitrary value of $\alpha = 0.05$ is considered. From this hypothesis analysis, we found that the value of p for the benchmark methods is less than the threshold value of α, as shown in Table 3.4 and thus rejects the null hypothesis. Hence proposed DASP method is a better approach for IoT service placement in the Fog-Cloud computing environment than the existing benchmark and state-of-the-art service placement approaches.

3.4.5 Time Complexity Analysis

The time complexity for the baseline service placement strategies are given as follows. The service placement using CloudOnly environment depends on the number of cloud servers *(M)* and the number of IoT/IIoT application services *(K)* is given by *O(KM)* and the default EdgeWards service placement strategy considers the available fog nodes *(N)* and it takes *O(KN)*. The time complexity for the BLB approach is $O((|R| + |K| + |K| * |R|) * log|R| + |K| * log|K|)$, where *'R'* is the total computing resources consist of both the fog and cloud resources. The time complexity for the applied FFD approach depends on the number of services and the fog computing resources used in the infrastructure. The resources comparison of the fog nodes and the IoT/IIoT services take *O(KN)* time, and sorting the services in decreasing order of resources requests takes *O(K log K)*. The overall time complexity for the FFD approach is given by *O(KN + K log K)*, where *K* is the number of IoT/IIoT application services and *N* is the number of fog nodes present in the fog layer.

The time complexity for the proposed DASP approach depends on the number of fog nodes used in Layer 2 of the computing infrastructure and IoT/IIoT services to be placed onto the fog nodes. The available resource comparison will take *O(KN)* to compare the service resource requirement with available resources of the fog nodes, where *K* and *N* are the number of IoT/IIoT services and fog nodes present in the considered architecture, respectively. After satisfying the criterion of available resources with the fog nodes, the computation of both service time and the service cost is dependant on the constant time (*C*). Thus the services placement after satisfying the constraints in the Fog-Cloud environment depends on the number of IoT/IIoT services and the constant time given as *O(KC)*. Hence the overall time complexity is *O(KN+KC)*. The final time complexity for the proposed DASP approach is given by *O(K(N+C))*.

3.4.6 Limitations of the Work

The proposed DASP approach considers the service placement in the Fog-Cloud computing environment. iFogSim simulator is used for evaluating the proposed service placement strategies, which uses the VMs based fog framework to provide the resources to host the IoT/IIoT services. Using VMs based fog framework for resource provisioning is not efficient as deploying VMs consumes more PMs resources, and takes high

booting time. The inter-dependent IoT/IIoT services are not considered for evaluating the performance of the proposed service placement strategy using DASP approach.

3.5 Summary

This Chapter discusses two service placement strategies to place the IoT/IIoT service requests in the Fog-Cloud computing environment. The FFD method is applied for service placement and evaluated for different network topologies. Further, the service placement problem is solved using the novel cost-efficient deadline-aware service placement algorithm to reduce the service time, service cost and the energy consumption. Also, it ensures the IoT service' QoS in terms of service deadline, service cost, and resource availability. The proposed DASP approach is evaluated for the various topology configurations in Tier 2 of the computing architecture. The experimental results show that using fog computing to process the data at the edge of the network will reduce the service time, service cost, energy consumption and network resource usage for the IoT/IIoT services and thus reduce the cloud resource usage. The VMs based resource provisioning framework is considered in the simulation, but using VMs for servicing all IoT services may not be feasible as booting time is high and consumes more resources, and deploying more VMs on a single node might reduce the performance of the PM. The next Chapter discusses the container-based fog framework and the QoS aware IoT/IIoT service placement strategies in the fog computing environment.

Chapter 4

Container-based Framework for QoS aware IoT/IIoT Service Placement in Fog Environment

The different virtualization techniques used for resource provisioning are Virtual Machines (VMs) and Containers. The resource provisioning framework in the iFogSim simulator uses the VMs deployed on the PMs in the data center. Resource provisioning using VMs is not adaptable in a resource constrained fog nodes since VMs boot time is high, consumes more resources, and hosting more VMs in a single physical machine degrades the physical machine's performance (He et al., 2012; Nguyen et al., 2020). Thus using VMs in a fog environment increases the service time for the delay-sensitive IoT/IIoT applications. Hence, containers are the preferable lightweight virtualization technique used for resource management in a fog environment due to their fast startup time, reducing the resource management overhead, and providing rapid and high scalability (Fayos-Jordan et al., 2020). This motivated us to develop the fog framework on resource constrained nodes using the docker containers.

The smart manufacturing environment has heterogeneous sensors for monitoring the devices and generates vast data. Using fog computing devices in the industrial environment to process heterogeneous data reduces the service time and avoids significant failures. Also, reducing the total energy consumption increases the lifetime of the battery-operated fog nodes and thus increases the reliability and availability of those devices in the industrial environment. But the primary challenge is to find suitable fog nodes that are distributed and vary significantly in terms of resource availability, data processing speed, and service time to host IIoT service requests to process the data. Hence, better service placement strategies are essential for the placement of IIoT services in the fog computing environment such that the resource-constrained devices are utilized efficiently (Chiu et al., 2018; Lin et al., 2018). Therefore, the placement of IIoT services on the resource-constrained fog nodes is referred to as an NP-hard problem (Brogi et al., 2020). The service placement problem is formulated as the multi-objective optimization problem to minimize the service time, service cost, and energy consumption in the fog computing environment. This chapter explains the proposed work on a two-level resource provisioning fog framework using docker containers on devices with a 1.4 GHz

64-bit quad-core processor. Further, we explain three novel approaches for solving the IIoT service placement problem in the fog computing environment. The proposed work with three novel approaches for solving the service placement problem is based on our publications (Natesha and Guddeti, 2021a, 2022).

4.1 Two-level Architecture for Servicing IoT/IIoT Applications

Fog computing is defined as the decentralized computing architecture for processing the data at the network level using smart gateways, routers, and micro-data centers as the fog nodes combined with the advantages of cloud and virtualization techniques. Thus, fog computing provides the computational and storage resources close to the data source node (Aazam et al., 2018c). Hence, we develop a two-level fog framework to provide the resources using docker and containers on resource constrained 1.4 GHz processor devices.

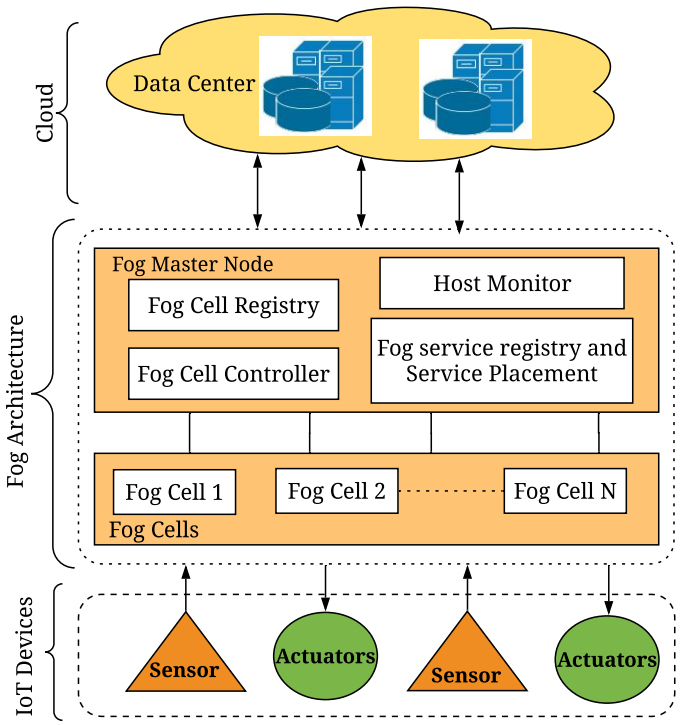


Figure 4.1: Fog computing architecture

Figure 4.1 shows the fog computing architecture considered for provisioning the resources to process the IoT data. The lower layer consists of the physical sensor nodes and actuator devices. The middle layer contains the fog nodes generally considered as

networking devices or independent MDC as fog servers. The two-level fog infrastructure is used to provide the resources and place the service on the fog nodes. The two-level fog infrastructure consists of Fog Master Node (FMN) and Fog Cell (FC). FMN is responsible for continuous monitoring of the topology and deciding the placement of the services based on the designed service placement strategies. FC is the independent node that provides the computational resources and hosts the service. Fog cells process the data, send back the response to actuators, and transfer the data to the upper layer for further processing.

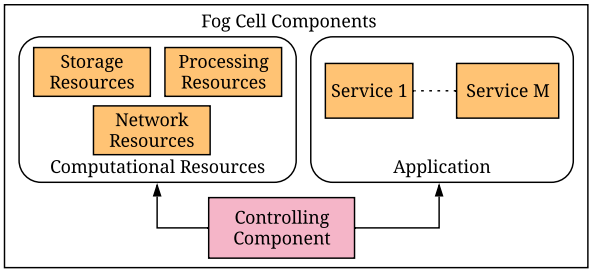


Figure 4.2: Fog Cell architecture

FMN consists of Fog Cell Registry; it registers the fog cell once entered into the topology and monitors the FC. The host Monitor is responsible for continuous monitoring of the available and used resources in the fog nodes. The host monitor controls the status of available resources in FCs and helps decide the placement of services. Fog Cell Controller is responsible for dynamic topology controlling and monitoring the FCs; therefore, many FCs can be added and used to deploy the services dynamically in the fog infrastructure. The fog service registry performs the registering of the services to be deployed. It is also responsible for the placement of services on the fog nodes based on the service placement algorithm.

Fog Cells: Fog Cells are the nodes that provide the computing resources and host the applications. The components of fog cells are shown in Figure 4.2. The applications contain multiple independent services, and these services are deployed on the fog cells based on the available computing resources. The controlling component present in the FC is responsible for managing the usage of resources and deployment of services. The data is processed in the fog environment, and then the decision control signals are transferred to the actuators to take action based on the control signal. Thus, using the fog computing environment reduces the service time for IoT applications.

4.2 Service Placement Problem Formulation

Table 4.1: List of Notations

Notations	Description
S_M	set of all services
F_N	set of all fog nodes
t_{pro}	processing time in seconds
t_{com}	communication time in seconds
t_{av}	service availability time in seconds
T_t	Total service time in seconds
D_d	service deadline in seconds
s^{size}	service size
s^{size}_{req}	size of service request
s^{size}_{res}	size of service response
γ_j	processing capability of a fog node j
β_f	bandwidth between nodes in Mbps
λ_{in}	input rate/arrival rate of service requests on fog node
C_{total}	total service cost in $
C_{max}	maximum service cost in fog given in $
C_{ec}	energy consumption cost of fog node given in $
C_{pro}	processing cost in fog environment given in $
C_{sto}	storage cost in fog environment given in $
L^{cpu}_a	required amount of processing resource (in mips per request)
u^{CPU}_c	unit cost in $ for processing resource in mips at fog node
u^{sto}_c	unit cost in $ for storage resource at fog node (per byte per second)
u^{eng}_c	unit cost in $ for energy consumption per joule in fog environment
r^{sto}_j	amount of storage resource used in fog node (in bytes)
E_{max}	maximum energy consumption in joules
E^{Total}_{eng}	total energy consumption of fog nodes in joules
E^{eng}_{fj}	energy consumption of fog node j in joules
E^{eng}_{tra}	energy consumption during transmission in joules
E^{eng}_{pro}	energy consumption during processing in joules
P^{idle}_f	power consumption of fog node in idle state in watts
P_{trans}	maximum power consumption during transmission in watts
P_{proc}	maximum power consumption during processing in watts
$time$	total time duration in seconds

4.2.1 Resource Constraints in Fog

Table 4.1 shows the list of mathematical notations used in this section. The placement of services on the fog node depends on the dynamic usage and availability of computational resources in the fog nodes. The services hosted on a fog node consume resources such as CPU, RAM, and storage. Thus, the resource demands of the deployed services on these fog nodes must not exceed the fog nodes' available computational resources since these fog nodes have limited computational and storage resources. The following equations give the resource constraints for the fog environment, and the placement of

services on the fog nodes should satisfy the resource constraints of each service request, as defined by Equation (4.1).

$$\forall f_j \in F_N \begin{cases} \sum_{\forall s_i}^{S_M} CPU^{req}(s_i) * x_{ij} \leq CPU_{available}(f_j) \\ \sum_{\forall s_i}^{S_M} RAM^{req}(s_i) * x_{ij} \leq RAM_{available}(f_j) \\ \sum_{\forall s_i}^{S_M} storage^{req}(s_i) * x_{ij} \leq storage_{available}(f_j) \end{cases}$$

$$\begin{aligned} &where, \\ &\quad s_i \in S_M, \;\; i \in [1, M] \;\; services \\ &\quad f_j \in F_N, \;\; j \in [1, N] \;\; fog\ nodes \end{aligned} \tag{4.1}$$

$$x_{ij} = \begin{cases} 1, & \text{if } s_i \text{ is placed on fog node } f_j \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

A set of all applications (services) S_M requests for resources {RAM, CPU, storage}, and to decide for placement of these services, the total available resources on the fog nodes are denoted by f_j should be more than the resource requests by the IIoT service. The allocation of the services on the fog nodes is indicated by Equation (4.2), x_{ij} =1 if the service is placed on the fog node else, x_{ij}=0.

4.2.2 Service Time and Energy Consumption in Fog

The service time is defined as the sum of processing, communication, and service availability time for the services in the fog computing environment given by Equation (4.3). The processing time t_{pro} depends on the service size (data size) s_i^{size} and the processing capacity of the fog node γ_j given by Equation (4.4).

$$T_t = t_{pro} + t_{av} + t_{com} \tag{4.3}$$

$$t_{pro} = \sum_{i=1}^{M} \sum_{j=1}^{N} \left(\frac{s_i^{size}}{\gamma_j} \right) x_{ij} \tag{4.4}$$

$$t_{com} = \sum_{i=1}^{M} \sum_{j=1}^{N} \left(\frac{s_{req}^{size} + s_{res}^{size}}{\beta_f} \right) x_{ij} \tag{4.5}$$

The communication time t_{com} is defined as the total time involved for transferring the service request to fog nodes and response back to the actuators. t_{com} depends on the service size (data size) and the link capacity β_f connected between the nodes is given by Equation (4.5). The service availability time t_{av} involves the time taken for selecting the fog node to deploy the services and data. Also, it depends on the total time of the service request in the waiting queue to get the selected fog nodes' computational resources. Thus t_{av} depends on the amount of time it takes for completing the current service request. The total service time calculated for each of the services should be less than the service deadline defined for the application requests, given by Equation (4.6).

$$T_t \leq D_d \tag{4.6}$$

It is assumed that if the fog node is not hosting any services, these fog nodes will consume some energy during the idle state. The energy consumption of the fog node during active state while hosting/running some services is calculated (Tang et al., 2018; Jalali et al., 2016; Murtaza et al., 2020). The fog node's energy consumption is defined as the sum of the energy consumption during the transmission of data and the energy consumption during the processing of the services defined by Equation (4.7). The Equations (4.8) and (4.9) define the energy consumption during the transmission and processing of the application, respectively.

$$E_{fj}^{eng} = (E_{tra}^{eng} + E_{pro}^{eng}) \tag{4.7}$$

$$E_{tra}^{eng} = \sum_{t=0}^{time} \left(P_f^{idle} + \left(\frac{S_i^{size}}{\beta_f} * P_{trans} \right) \right) \tag{4.8}$$

$$E_{pro}^{eng} = \sum_{t=0}^{time} \left(P_f^{idle} + \left(\frac{S_i^{size}}{\gamma_f} * P_{proc} \right) \right) \tag{4.9}$$

$$E_{eng}^{Total} = \sum_{j=1}^{N} E_{fj}^{eng} \tag{4.10}$$

The energy consumption depends on the maximum power consumed by the fog node per unit time for transmitting and processing the whole data as defined by Equations (4.8) and (4.9), respectively. The total energy consumption of the fog nodes for servicing the IIoT application services is defined by Equation (4.10).

4.2.3 Service Cost in Fog

The service cost for the IIoT applications deployed on the fog node is defined as the sum of processing cost, storage cost, and energy consumption cost in the fog infrastructure is given by Equation (4.11).

$$C_{total} = C_{pro} + C_{sto} + C_{ec} \tag{4.11}$$

The processing cost for the service depends on the amount of CPU/MIPS used for processing the application service request, and the arrival rate of the service request on the fog nodes is given by Equation (4.12). The storage cost depends on the amount of memory used for storage and the size of the service request, as defined by Equation (4.13).

$$C_{pro} = \sum_{i=1}^{M} \sum_{j=1}^{N} \left(u_c^{CPU} * L_a^{cpu} * \lambda_{in} * x_{ij} \right) \tag{4.12}$$

$$C_{sto} = \sum_{i=1}^{M} \sum_{j=1}^{N} \left(u_c^{sto} * r_j^{sto} * x_{ij} \right) \tag{4.13}$$

The fog infrastructure's energy consumption cost depends on the amount of energy consumed by the fog nodes and the time duration. The energy consumption cost is calculated as the product of total energy consumed and the cost of the unit energy consumption in the fog infrastructure as defined by Equation (4.14).

$$C_{ec} = \sum_{i=1}^{M} \sum_{j=1}^{N} \left(E_{eng}^{Total} * u_c^{eng} * x_{ij} \right) \tag{4.14}$$

4.2.4 Optimization Model for Service Placement

The set of available fog nodes denoted by $F_N = \{f_1, f_2, ...f_N\}$ and a set of all services to be deployed on the fog nodes denoted by $S_M = \{s_1, s_2, ...s_M\}$. The main objective is to find the fog nodes for optimal service placement such that the service time, service cost, and energy consumption for the IIoT applications are minimized in the fog computing environment. Hence, the service placement problem in fog computing environment is formulated as the multi-objective optimization problem. The formulated multi-objective optimization functions are given by Equations (4.15)-(4.18).

$$Minimize \sum_{i=1}^{M} \sum_{j=1}^{N} (T_t(s_i, f_j)) \tag{4.15}$$

$$Minimize \sum_{i=1}^{M} \sum_{j=1}^{N} (C^F_{total}(s_i, f_j)) \tag{4.16}$$

$$Minimize \sum_{i=1}^{M} \sum_{j=1}^{N} (E^{Total}_{eng}(s_i, f_j)) \tag{4.17}$$

$$\begin{aligned} &Subject\ \ to: \\ &\quad T_t \leq D_d \\ &\quad C_{total} \leq C_{max} \\ &\quad 0 \leq E^{Total}_{eng} \leq E_{max} \\ &where, \\ &\quad s_i \in S_M,\ \ i \in [1, M]\ \ services \\ &\quad f_j \in F_N,\ \ j \in [1, N]\ \ fog\ nodes \end{aligned} \tag{4.18}$$

The constraints defined for the said multi-objective functions are as follows:

- Total service time for the IIoT application should be less than the service deadline defined for the deployed IIoT application.
- The service cost should be less than the maximum service cost defined for the said application.

- The energy consumption should be less than the maximum energy consumed by the fog nodes.
- Each service request for the resources and placement decision should satisfy the resource requirement defined by Equation (4.1). The service should also be placed only on one fog node in the fog computing environment.

4.3 Service Placement Strategies in the Fog Computing Environment

In this section, we propose three novel strategies for IIoT service placement in the fog computing environment. The first strategy is on a novel Elitism-based Genetic Algorithm (EGA) to place the IIoT service requests. Later, we proposed two novel strategies for IIoT service placement by using hybrid meta-heuristic algorithms, namely: MGAPSO and EGAPSO by combining the genetic algorithm with Particle swarm optimization (MGAPSO) and EGA with PSO, respectively.

4.3.1 Strategy 1: EGA based Service Placement

The service placement problem in the fog computing environment is solved using the Elitism-based Genetic Algorithm (EGA). Elitism-based GA passes the first best or few best chromosomes from the current generation to the next generation. Thus it avoids random destruction of the best chromosome after crossover and mutation operations. Using elite chromosomes will preserve the degeneration of the population and premature convergence. Hence, Elitism-based GA is preferred to the traditional GA for service placement in the fog computing environment. The operations involved in the Elitism-based Genetic algorithm are: Fitness evaluation, Selection of Elite chromosome, Crossover, and Mutation. The chromosome length is equal to the number of services, and the values (genes) represent the fog node number.

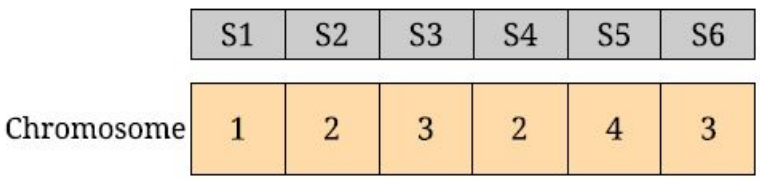


Figure 4.3: Chromosome

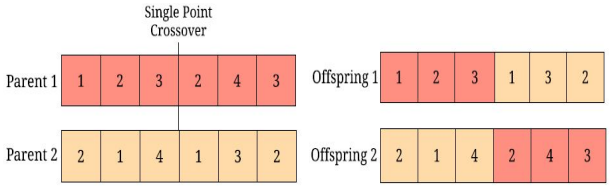
For example, consider six service requests, as shown in Figure 4.3, and the chromosome gene value represents the possible service placement on the fog nodes. Further, the number of fog nodes considered in the fog infrastructure is equal to 4.

Selection Operation

The elitism concept is used to select the best individual for the next generation directly without performing the crossover and mutation operations. Using elitism avoids the loss of the best individual after crossover and mutation operations and speeds up the Genetic Algorithm's performance. The elitism rate for selecting the best individual for the next generation should be less. Thus it avoids the degeneration of the population. Hence, the population is sorted based on the fitness value and selects 8% of the population as the elite chromosomes and passes them to the next generation.

Crossover Operation

The crossover operation is performed on the remaining population to obtain the offspring for the next generation. The single-point crossover operation is performed on the two-parent chromosomes to produce the new offspring from the remaining population, thus finding the best feasible chromosome for the next generation. The single point is selected on both the parent chromosomes and values (genes) after the point is swapped between the two parents to obtain the new offspring. The Figures 4.4 (a) and 4.4 (b) show the chromosome before and after the single point crossover operation, respectively.



(a) Before Crossover (b) After Crossover

Figure 4.4: Crossover operation between the two parent chromosome (a) Before Crossover (b) After Crossover

Mutation Operation

Mutation operation is performed on the new offspring chromosome to mutate one or more genes in the original chromosome to obtain the new chromosome. Thus, the mutation operation on the chromosome preserves the diversity within the population and the premature convergence. The mutation operation is performed on the chromosome by replacing the gene values with the random value within the range of the number of fog nodes [1, N].

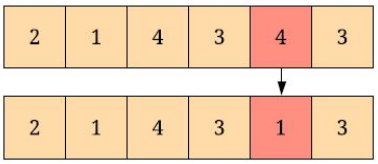


Figure 4.5: Mutation Operation

The example for the mutation operation on the chromosome is shown in Figure 4.5. The above procedure continues until the maximum number of generations is considered and returns the best individual chromosome. The best individual chromosome represents the possible optimal solution for the placement of service requests on the fog nodes to optimize the energy consumption, service cost, and service time, thus ensuring the QoS of IIoT applications.

$$Fitness = \frac{3}{T_t(s_i, f_j) + C_{total}(s_i, f_j) + E_{eng}^{Total}(s_i, f_j)} \tag{4.19}$$

Algorithm 4.1: EGA: Elitism-based Genetic Algorithm for Service Placement in the Fog Computing Environment

Input: Set of all Services S_M=$\{s_1,s_2,.....s_M\}$
Set of all Fog Nodes F_N=$\{f_1,f_2,.....f_N\}$
Result: service_allocation_list ← [ ]
Initialize: crosssover_probability (P_c) ← 0.5, mutation_rate (P_m)←0.3
num_generation ←250, elitism_rate ← 0.08, population←100
population ← Generate Population Randomly()
fitness ← Calculate Fitness(population) using Equation (4.19)
elite_list ← Sorted(fitness,population,elitism rate)
service_allocation_list ← best_individual
rem_population ← (population - elite_list)
while *num_generation* **do**
    parent1, parent2 ← selection operation(rem_population)
    `// Perform Crossover operation`
    child1, child2 ← crossover operation(parent1, parent2, P_c)
    `// Apply Mutation operation on the new child chromosomes`
    newchild ← mutation operation(child1, child2, P_m)
    new_population ← elite_list $\bigcup$ newchild
    fitness ← Calculate Fitness(new population) using Equation (4.19)
    elite_list ← Sorted(Fitness, new_population,elitism rate)
    rem_population← (new_population - elite list)
    service_allocation_list← best_individual
end
return service_allocation_list

Algorithm 4.1 shows the complete steps involved for service placement in the fog environment. The initial population is considered randomly, where each chromosome represents the possible solution of service placement in the fog computing environment. After generating the initial population, the fitness value is calculated for each of the chromosomes by using Equation (4.19). The fitness function is defined as a fraction of the sum of the service time, cost, and total energy consumed. These are multiplied with constant weights of $\frac{1}{3}$, which depends on the number of objective parameters in the optimization model.

4.3.2 Strategy 2: MGAPSO based Service Placement

Further, two different meta-heuristics based hybrid service placement strategies are developed to deploy the IIoT service requests on the fog computing architecture. The Genetic Algorithm (GA) and proposed EGA algorithm are combined with PSO to develop the hybrid MGAPSO and EGAPSO service placement strategies, respectively.

Algorithm 4.2: MGAPSO based Service Placement

Input: Number of fog nodes *N*
Number of service requests *M*
Result: best_chromosome
Intialize: Population← 100, Num_generation ← 300, Mutation_rate (P_m) ← 0.3
Crossover_rate (P_c) ← 0.5
Population← Generate Initial Population Randomly
while *(Num_generation!=0)* **do**
    // Find the Fitness value of chromosome using Equation (4.19)
    fitness ← Calculate_Fitness(Population)
    // Call Selection Operation()
    parent ← Selection operation(Population, fitness)
    // Call PSO using **Procedure 1**
    newparent1, newparent2 ← PSO_Procedure(parent1,parent2, fitness)
    // Apply single point cross over operation on the best particles returned by Procedure 1
    child1, child2 ← Call CrossOver_Operation (newparent1, newparent2, P_c)
    // Apply Mutation operation on the new child chromosomes
    newchild ←Call Mutation_Operation (child1, child2, P_m)
    fitness ← Calculate Fitness(new population) using Equation (4.19)
    best_ chromosome ← Update Service Allocation List
end
return best_ chromosome

Both GA and PSO are meta-heuristic algorithms that work on the initial random population and provide near-optimal solutions for the NP-hard problems. Initially, the

Procedure 1: PSO_Procedure(Particle, Fitness)

Input: Particles, fitness values of particle
Result: best_particle
Initialize: $c_1, r_1, c_2, r_2, P_{best}, G_{best}$
while *Num_generation!=0* **do**
  for *each particle* **do**
    // Update the Velocity and Position of the Particle using the below Equations
    $V_{t+1} = V_t + c_1 * r_1 * (P_{best} - X_t) + c_2 * r_2 * (G_{best} - X_t)$
    $X_{t+1} = X_t + V_{t+1}$
  end
  Find the Fitness of the particle using fitness function Equation (4.19)
  Update the P_{best} and G_{best}
  update best_particle
end
return best_particle

fitness of the chromosome is calculated by using Equation (4.19). Then the new chromosome is selected by using the selection operation of the GA. After the Selection operation, the PSO_Procedure (Procedure 1) is called to find the best feasible particle using the PSO algorithm. Each chromosome is mapped as the particle and then update the particle position and velocity. After the maximum number of iterations, the PSO procedure will return the best particle. The single-point crossover operation is performed on the best particle returned by the PSO to find the new offspring for the next generation. Then mutation operation is applied to find the new chromosome. For mutation operation, the gene (value) is replaced randomly with the value of the number of fog nodes [1, N]. The above procedure continues until the maximum number of generations and returns the best chromosome. The chromosome returned by the MGAPSO will represent the best feasible service allocation order on the fog nodes. Algorithm 4.2 shows the complete procedure of MGAPSO based service placement strategy.

4.3.3 Strategy 3: EGAPSO based Service Placement

The convergence rate for the MGAPSO is slower as it passes all the chromosomes to the next generation till it gets the best chromosome. To address the issue of the MGAPSO algorithm, another hybrid EGAPSO algorithm is developed by combining the Elitism-based GA with the PSO algorithm for IoT/IIoT service placement in the fog computing environment.

Algorithm 4.3: EGAPSO based Service Placement

Input: Number of fog nodes N
Number of service requests S
Result: service_allocation_list
Intialize: Population← 100, Num_generation ← 300
Mutation_rate (P_m) ← 0.3, Crossover_rate (P_c) ← 0.5, service_allocation_list ← best individual, elitism_rate ← 0.08
Population← Generate Initial Population Randomly
`// Initial chromosome represents the possible allocation of services on the fog nodes`
fitness ← Calculate Fitness(population) using Equation (4.19)
elite_list ← Sorted(fitness,population,elitism_rate)
rem_population ← (population - elite_list)
while *(Num_generation!=0)* **do**
`// Call Selection Operation()`
parent1, parent2 ← selection operation(rem_population)
`// Call PSO using` **`Procedure 1`**
newparent1, newparent2 ← PSO_Procedure(parent1,parent2, fitness)
`// Apply single point cross over operation on the best particles returned by Procedure 1`
child1, child2 ← Call CrossOver_Operation (newparent1, newparent2, P_c)
`// Apply Mutation operation on the new child chromosomes`
newchild ←Call Mutation_Operation (child1, child2, P_m)
new_population ← elite_list $\bigcup$ newchild
fitness ← Calculate Fitness(new population) using Equation (4.19)
elite_list ← Sorted(Fitness, new_population,elitism_rate)
rem_population← (new_population - elite list)
service_allocation_list← best individual
end
return service_allocation_list

The convergence rate of the Elitism-based GA is faster than the simple GA since it saves the best chromosome for the next generation without performing the crossover and mutation operation. Hence, the few elite chromosomes are passed to the next generation directly, and then the PSO operation is applied on the remaining chromosomes using Procedure 1. Then, the crossover and mutation operations are performed on the particles returned by the PSO to get the best chromosome. The above procedure is continued until the maximum number of generations and returns the chromosome. The best individual chromosome returned by the EGAPSO will represent the best feasible service allocation order on the fog nodes. Algorithm 4.3 shows the complete procedure of EGAPSO based service placement strategy.

4.4 Performance Evaluation

4.4.1 Experimental Testbed Setup

The Raspberry Pi Model 3B+ devices with Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz quad-core processor, with 1GB LPDDR2 SDRAM and the 5V/2.5A DC power input are used to develop the fog-testbed infrastructure and run the proposed service placement strategies. All these fog nodes run Hypriot [1] a lightweight operating system for low computing power devices that supports the built-in docker[2]. Figure 4.6 shows the fog testbed infrastructure developed for performance evaluation with twenty fog nodes.

Figure 4.6: Fog Computing Testbed for performance evaluation

The different technologies or software components used to develop the fog computing framework to provision the computational resources to host and run the IoT applications is shown in Figure 4.7.

Redis Database: Redis [3] is open source, scalable database, fast and easily deployable on the devices. Both the FMN and FCs run the Redis database instances. In the fog nodes, two types of Redis database instances are used: local and shared. The local Redis instance is used in FCs. Both local and shared Redis database instances are used in the FMN. The local instance in both FCs and the FMN is used to store the fog node/cell configuration information, IP address, and the fog node's total resource utilization statistics. The deployable IoT service images and services output data are stored in the shared Redis database instance of the FMN.

[1] https://blog.hypriot.com/
[2] https://www.docker.com/
[3] https://redis.io/

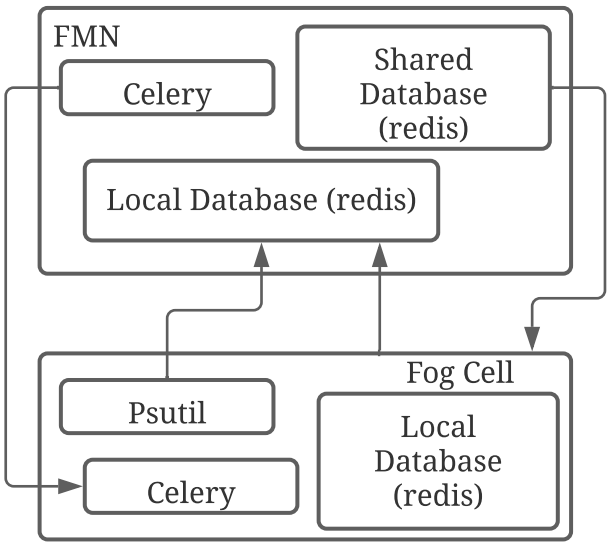


Figure 4.7: The Software Components used in the Testbed

Celery: Celery[4] is used to monitor and get the heartbeats of the fog nodes in the topology to check whether the status of the node is alive or not. The resources' usage is continuously monitored for both the fog master and cell nodes. In the fog framework, resource provisioning is decided based on the availability of the computational resources. FMN avoids service placement if the available resources are insufficient to satisfy the service resource requests. The Celery worker instances are present in both the FMN and fog cell nodes. The Celery worker in FMN triggers fog cells to download data and service images from the shared database, and the Celery present in the fog cell will execute the deployed service requests on the fog cells.

REST is the main factor that enables the communication in micro-services architecture due to various light software component footprints present in the docker. The Flask python framework is used in a developed testbed for communication purposes due to its minimalistic features. The power meter is used for measuring the energy consumption of the fog nodes in the testbed with an input capacity of 240V, 50Hz, 20A.

4.4.2 Application Types for Testbed

The IoT applications such as the smart building, smart manufacturing industry (Industry 4.0), and smart healthcare (Kumari et al., 2018) deploy more number IoT devices

[4]http://www.celeryproject.org/

to monitor and control the environment; which continuously sense and generate massive data. Processing this vast data in the fog environment reduces the service time and provides service in real-time. The performance of the proposed service placement algorithm in the fog computing environment is evaluated for the master-worker IoT application model, which consists of a set of independent modules. Using fog nodes in the industrial environment to process heterogeneous data reduces the service time and avoids significant failures. Also, reducing the total energy consumption increases the battery-operated fog nodes' lifetime, thus increasing the reliability and availability of those devices in the fog environment. An IoT application includes separate modules/services which perform some operation and then send back the response to the actuators. Based on the application scenarios, each IoT application is represented in the form of a directed graph (Mahmud and Buyya, 2019). The sensor data from Melbourne city [5] which consists of sensor readings such as the temperature, humidity, and light are continuously read and accordingly find the average, minimum, and maximum values of each data type and then decide to send the control signal information to the actuators.

4.4.3 Results and Discussion

The proposed service placement strategies are evaluated on the fog infrastructure setup, which dynamically provisions the resources to host the service requests. A different set of services ranging from 200 to 1000 services are considered to carry out the experiments. The number of fog nodes in the testbed is varied from five to twenty in terms of five (i.e., 5,10,15, and 20) nodes. The IIoT service requests are placed on the fog nodes in the testbed using developed service placement strategies and then measured the service time, service cost, and the total energy consumption in the fog environment.

For the proposed algorithms (EGA, MGAPSO, and EGAPSO), the number of generations varies from 50 to 350 in terms of 50, and the mutation rate ranges from 0.1 to 0.3 in steps of 0.05. It is found that the fitness value of the chromosome remains constant after 300 generations at a 0.3 mutation rate as shown in Figure 4.8. Hence, the mutation rate of 0.3 and the number of generations to 300 are used for the proposed hybrid algorithms. The experimental parameters considered for different service placement strategies are given in Table 4.2.

[5] https://data.melbourne.vic.gov.au/Environment/Sensor-readings-with-temperature-light-humidity-ev/ez6b-syvw/data

Table 4.2: Experimental Parameters

Parameters	Values
Number of Nodes	5-20
Number of Services	200-1000
Number of Generation G	300
Population Size N	100
elitism_ rate	8%
Mutation_rate (P_m)	0.3
Crossover_rate (P_c)	0.5
c1, c2	2
r1, r2	0.5

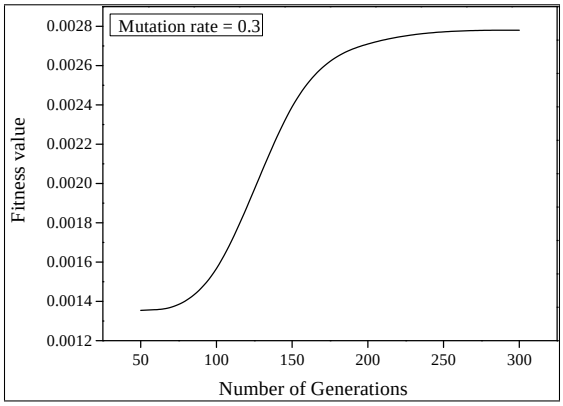


Figure 4.8: Fitness value vs Number of Generations

The experiments are carried out five times for the different service placement strategies and presented an average of the experimental results. The performance of the proposed service placement algorithms are compared with the existing state-of-the-art Genetic Algorithm (GA) (Canali and Lancellotti, 2019), Delay Energy based Task Scheduling (DEBTS) (Yang et al., 2018), Double Matching Strategy (DMS) (Jia et al., 2018), Simulated Annealing (SA) (Rezazadeh et al., 2018), Particle Swarm Optimization (PSO) (Mishra et al., 2018), GAPSO (Yadav et al., 2019) and the two baseline algorithms, namely: First-Fit (FF) and Branch-and-Bound (BB) algorithms (Rakshith et al., 2018) for service placement in the fog computing environment. The experiments are carried out on the developed testbed to evaluate the performance of the proposed EGA, MGAPSO, EGAPSO, and state-of-the-art service placement strategies.

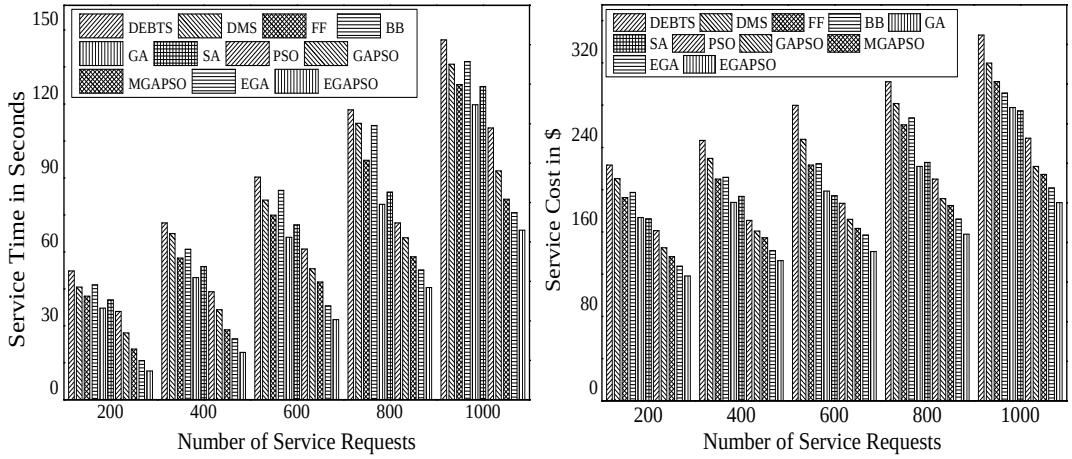
The First-Fit algorithm considers the availability of the computational resources and allocates the services on fog nodes. But, First-Fit does not consider the optimal allocation, and hence it reduces the efficiency by underutilizing the available limited computational resources. Thus it increases the service time, service cost, and energy

consumption in the fog computing environment.

On the other hand, the BB algorithm allocates the services on the fog nodes. As the number of services and the fog nodes in the topology increases, the service time increases exponentially, thus maximizing the service cost and energy consumption in the fog computing environment for servicing the IIoT applications. The DMS approach is based on the deferred acceptance approach for resource allocation in the fog computing environment. The DEBTS approach optimizes the energy consumption and the service delay in the homogeneous fog network. The Lyapunov optimization technique is used to minimize the total energy consumption of fog devices. The GAPSO is the hybrid algorithm for optimizing the energy consumption and the service delay in the fog computing environment. This hybrid GAPSO approach considers the GA and PSO approach for deciding the service allocation in the fog computing environment. The service placement strategies considered for performance evaluations are not optimal as it takes more time to provide the service and thus not suitable for the delay-sensitive IIoT applications. Hence to overcome these problems, EGA and hybrid algorithms are proposed to allocate the services optimally and therefore minimize the service time, service cost, energy consumption in the fog computing environment.

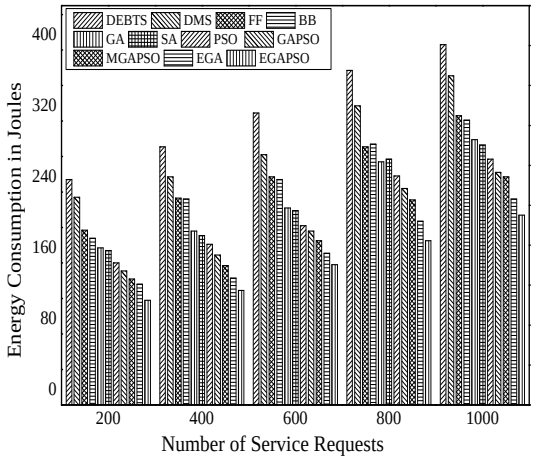
The number of service requests is varied from 200 to 1000. These service requests are generated using uniform distribution and then deployed on the developed two-level fog computing framework using different service placement strategies. The experimental results show that the proposed EGAPSO approach minimizes the service time, cost, and energy consumption for servicing the IIoT applications in the fog computing environment. The experimental results for the various service placement algorithms are shown in Figures 4.9 and 4.10.

The fog nodes in the framework will provide the resources for hosting the IIoT applications and reduce the service time by performing the data processing operations on the available computing resources. The service placement algorithms in the fog computing environment are developed and evaluated for finding the total service time for a set of services by placing the requests on the fog framework. Figure 4.9 shows the total service time, service cost, and energy consumption for the different number of service requests deployed using different service placement strategies on fog testbed



(a) Service Time

(b) Service Cost



(c) Energy Consumption

Figure 4.9: Performance comparison of service placement strategies in terms of (a) Service Time (b) Service Cost and (c) Energy Consumption in the two-level fog computing framework

with 20 fog nodes. Figure 4.9 (a) shows the service time for various service placement algorithms. It is observed that the service time for the EGAPSO algorithm is less when compared to the other service placement algorithms.

The total service cost of the IIoT applications for the different service placement strategies is calculated in the fog computing environment. Figure 4.9 (b) shows the total service cost for servicing the IIoT application service requests in the fog computing framework. Figure 4.9 (b), it is observed that the EGAPSO algorithm uses the fog resources efficiently and further reduces the service cost as compared to the other service placement strategies. The total energy consumption of the fog computing testbed for servicing the deployed IIoT application requests is measured by using an external

power meter connected to the framework with an input capacity of 240V, 50Hz, 20A. The total energy consumed for running a set of services with different service placement algorithms is given in Figure 4.9 (c). It is observed from Figure 4.9 (c) that the total energy consumption of the fog nodes is less for EGAPSO as compared to the other service placement algorithms.

The number of fog nodes in the testbed is varied from five to twenty in terms of five (i.e., 5,10,15, and 20) nodes. The total service time for servicing the 1000 IIoT service requests is calculated. As the number of fog nodes increases, the service time for the IIoT applications minimizes and thus provides the service in real-time for the delay-sensitive IIoT applications. From Figure 4.10, it is observed that using more fog nodes can reduce the total service time for the IIoT service requests in the fog computing environment.

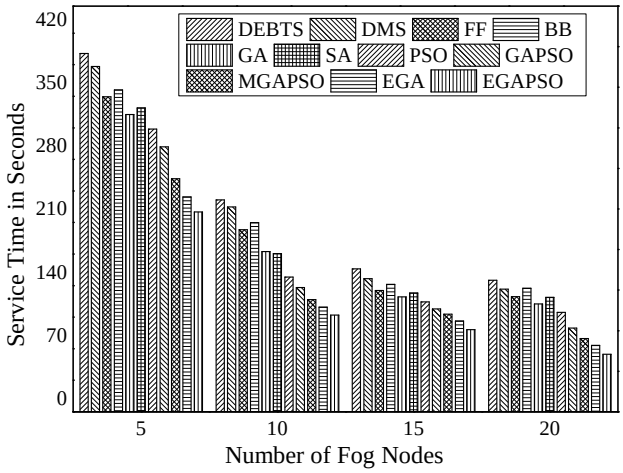


Figure 4.10: Service Time of various service placement strategies for different number of fog nodes in the testbed

4.4.4 Statistical Hypothesis Analysis

The statistical hypothesis testing is carried out for the proposed IIoT service placement algorithms. The t-test analysis is considered for evaluating the service placement strategies in the fog computing environment. The t-test for the proposed hybrid service placement strategies: EGA, MGAPSO and EGAPSO for service placement in the fog computing environment and also compared with First-Fit, Branch-and-Bound, DEBTS, DMS, GA, SA, PSO and GAPSO. The results of t-test analysis are shown in Table 4.3. For the null hypothesis, considered the threshold value of p or the significance level

(confidence level) denoted by α. The widely adopted threshold value of $\alpha = 0.05$ is considered. From this hypothesis analysis, it is found that the value of p for the EGAPSO method is less than the threshold value of α, as shown in Table 4.3 and thus rejects the null-hypothesis. Hence, among the proposed service placement algorithms, EGAPSO is the better approach for IIoT service placement in the fog environment.

Table 4.3: p-values for t-test analysis

Metric	Algorithms	p-Values
Service Time	First-Fit - EGAPSO	0.0271
	Branch-and-Bound - EGAPSO	0.0232
	DEBTS - EGAPSO	0.0371
	DMS - EGAPSO	0.0314
	GA - EGAPSO	0.0218
	SA - EGAPSO	0.0328
	PSO - EGAPSO	0.0291
	GAPSO - EGAPSO	0.0247
	MGAPSO - EGAPSO	0.0222
	EGA - EGAPSO	0.0256
Service Cost	First-Fit - EGAPSO	0.0292
	Branch-and-Bound - EGAPSO	0.0336
	DEBTS - EGAPSO	0.0347
	DMS - EGAPSO	0.0311
	GA - EGAPSO	0.0384
	SA - EGAPSO	0.0418
	PSO - EGAPSO	0.0235
	GAPSO - EGAPSO	0.0261
	MGAPSO - EGAPSO	0.0248
	EGA - EGAPSO	0.0212
Energy Consumption	First-Fit - EGAPSO	0.0260
	Branch-and-Bound - EGAPSO	0.0245
	DEBTS - EGAPSO	0.0221
	DMS - EGAPSO	0.0321
	GA - EGAPSO	0.0393
	SA - EGAPSO	0.0401
	PSO - EGAPSO	0.0214
	GAPSO - EGAPSO	0.0228
	MGAPSO - EGAPSO	0.0318
	EGA - EGAPSO	0.0232

4.4.5 Time Complexity Analysis

The time complexity for the First-Fit algorithm depends on the number of fog nodes *(N)* and the number of IoT/IIoT service requests *(M)* is given by $O(NM)$. The time complexity for Branch-and-Bound algorithm is $O(NM)^2$. The time complexity for the DEBTS and DMS algorithms is $O(NM)^2$, respectively. The time complexity for the GA, SA and PSO is *O(GNM)* and $O((NM)^2 + NM)$, $O(NM)$ respectively.

The time complexity of the Elitism based Genetic algorithm (EGA) depends on the number of generations, population size, and the operations: Fitness evaluation, elitism operation, Selection of an elite chromosome, Crossover, and Mutation. The time complexity is calculated as *O(Generations*(O(Fitness Evaluation) +O(sort operation to select elite list)+ O(selection operation) + O(P_c*Crossover Operation) + O(P_m* Mutation Operation))*, where P_c and P_m are the probabilities of crossover and mutation rates considered for the evaluation, respectively. For updating the elite chromosome, the chromosomes are sorted based on the fitness value and the time complexity for the sorting algorithm used is $O(NlogN)$. The time complexity for the Elitism-based Genetic Algorithm is represented as *O(G*(O(N) +* $O(NlogN)$*+ O(NM) + O(Nm) +O(P_c*Nm) +O(P_m*m)))*, where, *G* is the number of generations, *N* is the size of the population, *M* is size of the chromosome and *m* is the subset of population chromosome selected after the selection operation. If the P_c and P_m values are considered as constant values, then the overall time complexity of the Elitism-based Genetic Algorithm is $O(G(N * M + NlogN))$. The time complexity of the EGAPSO algorithm depends on the size of the generation, fitness evaluation, selection operation, update velocity, the position of a particle, crossover, and mutation operations. The time complexity for the PSO procedure is $O(NM)$.

The time complexity for GAPSO and MGAPSO is given by *O((G*NM)+NM)*. The time complexity of the EGAPSO algorithm is simplified as *O((G*NM) +(N log N) +NM)*, where *N* is the population size, *O(N log N)* is the time complexity for the sorting algorithm used for selecting the elite chromosome in the EGAPSO algorithm.

4.4.6 Limitations of the work

The proposed work considers the container-based fog framework to provide the resources for deploying the IoT/IIoT service requests. Further, various service placement strategies are developed to place the service. The limitation of the work is the inter-dependent IoT applications are not considered for service placement. Further did not consider handling device failures, load balancing, service migrations in the developed fog framework, if any accidental fog node failures in the developed fog framework.

4.5 Summary

This chapter describes the development of docker and containers-based two-level fog framework to provide the resources. Then the fog service placement problem was formulated as a multi-objective optimization problem to ensure the QoS of IoT/IIoT applications. Further, the EGA and the hybrid MGAPSO and EGAPSO based service placement strategies are proposed to place the IoT/IIoT service requests in the fog computing environment. The experiments are carried out for the proposed EGA and the hybrid MGAPSO and EGAPSO and the state-of-the-art service placement algorithms on the fog infrastructure testbed developed using the docker and containers on the cluster of 1.4 GHz 64-bit quad-core processor devices with twenty nodes. The different service placement strategies are assessed in terms of service time, service cost, and the energy consumption of the fog nodes in the two-level fog computing environment. The experimental results show that the proposed hybrid EGAPSO outperforms the other proposed and the existing state-of-the-art service placement strategies considered for the performance evaluation. The Industry 4.0/Smart Industrial applications are delay-sensitive, hence there is a need for the cost-efficient computing architecture to push the intelligence and data analysis close to the source device thus enabling the automation and real-time monitoring. Using the resource constrained fog nodes/server to deploy the machine learning models and analyze the various sensor data from the smart environments reduces the service time and thus enable the real-time monitoring in the smart environments. The fog server based framework for real-time data analytics in the smart industrial environment is discussed in the next chapter.

Chapter 5

Fog Server-based Framework for Real-time Data Analytics in Smart Application Environment

With the rapid growth in the use of IoT and IIoT devices in monitoring and surveillance environments, the amount of data generated by these devices has increased exponentially. There is a need for cost-efficient computing architecture to push the intelligence and data processing close to the data source nodes. Using fog nodes to process and analyze the sensor data at the network edge reduces the service latency, network congestion, thus avoiding significant failures in the smart industry/Industry 4.0 environment. This motivated to develop an intelligent and cost-efficient real-time monitoring system for the smart industrial environment using the available computing resources such as Industrial controller units (ICU), Gateways as fog server. The resource constrained network devices in the industrial environment are used as the fog server to deploy the machine learning model to analyze the sensor data and automate the real-time monitoring. With this motivation, a fog server-based framework is developed as a prototype for intelligent machine malfunction monitoring based on the machines' operating sounds. The developed prototype consists of a fog server at the network edge and the end node for analyzing the machines' operating sounds. Further, the supervised machine learning models are developed and deployed on the fog server to identify and classify the machines as normal and abnormal. Thus using a fog server in the industrial environment to analyze the data will enable real-time monitoring in the industry and avoid significant machine failures. The proposed fog server framework for real-time data analytics in Industry 4.0 environment is based on our publication (Natesha and Guddeti, 2021b).

5.1 Intelligent Machine Malfunction Monitoring System for Industry 4.0

The intelligent machine malfunction monitoring system is developed using the fog computing architecture for Industry 4.0 or smart industrial applications to detect the malfunctioning machines based on the machines operating sound. The Linear Prediction Coefficients (LPC) and Mel Frequency Cepstral Coefficients (MFCC) from the machine sounds are used to build and deploy the supervised machine learning models on the industrial controller units are considered as fog servers. The developed Machine Learning

(ML) models detect and classify the malfunctioning machine as normal and abnormal, thus enabling automation in the industrial environment.

5.1.1 Fog Computing Model for IIoT

The fog server architecture is used for processing the sensor data using the devices present in the manufacturing industry. The industrial controller unit (ICU)/Micro Data Center (MDC) (Aazam and Huh, 2015b) in the manufacturing industry is used as a fog server that can host and run the ML models to process and analyze the machine operating sounds. The fog architecture and the components of the fog server are shown in Figure 5.1.

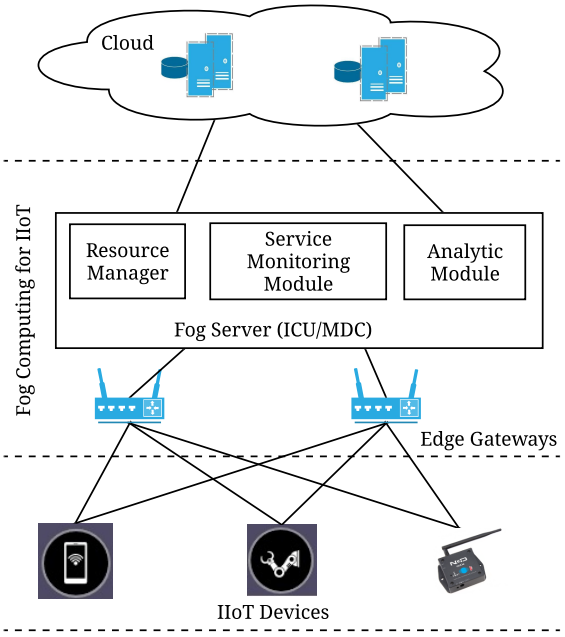


Figure 5.1: Fog Architecture for Smart Industry (Industry 4.0)

The fog server consists of different modules to process and handle the IIoT data in an industrial environment, such as resource manager, service monitoring module, and analytic module. The resource manager manages the dynamic resource provisioning, and the analytical module runs the ML models to analyze the IIoT data. The service monitoring module is responsible for controlling and monitoring the machines if any unusual events are detected.

In the architecture, the IIoT sensors are interfaced with the edge devices, which collect the data and transfer it to the centralized cloud server and fog server for processing

and analyzing the data to check for the fault/malfunction of the machines. Using a fog server in the industrial environment for machine monitoring reduces the machine down-time and thus increases the reliability and availability of the machines towards a higher production rate. The different ML models are developed and deployed on the fog server for analyzing the machine sounds to detect and identify the malfunctioning machines based on their operating sound.

The procedure for classifying the machine sounds as normal and abnormal by deploying the machine learning models on the fog server is shown in Figure 5.2.

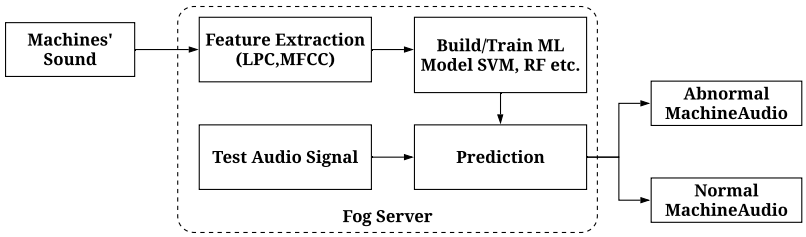


Figure 5.2: Machines' Sound Classification using Fog Server

5.1.2 Feature Extraction

The first step in the machines' sound classification is to extract the essential features of the audio signal. The Linear Prediction Coefficients (LPC) (Sanjaya et al., 2018; Alim and Rashid, 2018) and the Mel-Frequency Cepstral Coefficients (MFCC) (Ittichaichareon et al., 2012; Sahidullah and Saha, 2012) are considered as the audio features for classifying the machine sounds. LPC and MFCC are the widely used features in the sound analysis systems (Koolagudi et al., 2017). These LPC and MFCC features are robust and represent the steady and consistent source behaviors, and further, these features give very accurate estimates of sound and have low computational complexity.

5.1.2.1 LPC Features

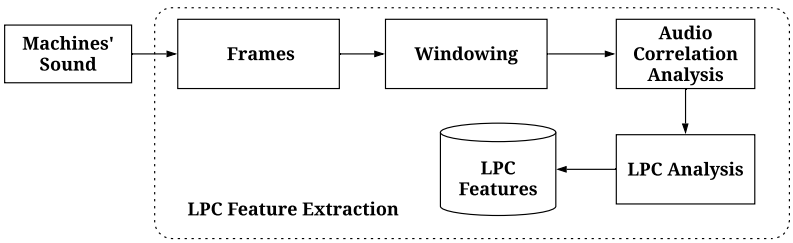


Figure 5.3: Procedure for LPC feature extraction

The LPC feature extraction procedure is shown in Figure 5.3. The first step is to frame the machine sound signal into overlapped frames so that the overlapped frames ensure that there is no signal loss. The audio signals are sampled at 16KHz and thus generate the small frames of length 25ms and considered frame shift length of 10ms. In the next step, the hamming window operation is performed to minimize the discontinuity of the signal from start to end of the audio frames. After windowing each frame, the autocorrelation analysis is done on all the frames of the audio samples. The final step is the LPC analysis to obtain the LPC features that are steady and consistent.

5.1.2.2 MFCC Features

The MFCC features are robust to noise, and the Mel Cepstrum represents the frequencies captured by the human ear. The extracted MFCC features are used for building the ML model. Figure 5.4 shows the procedure for MFCC feature extraction.

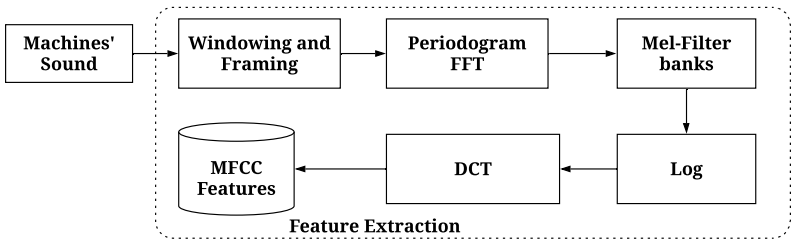


Figure 5.4: Procedure for MFCC feature extraction

The first step is framing and windowing the given audio signal to reduce the frame size so that the sampled audio signals do not vary significantly. The machine sound signal is segmented into the overlapped frames so that there is no signal loss from the original audio signal. The audio signals are sampled at 16KHz and thus generate the small frames of length 25ms with a frameshift of 10ms. Then the hamming window operation is performed on the audio frames to minimize the discontinuity of the signal. After generating the frames and windowing operation, the next step is to calculate the power spectrum of each frame using the Fast Fourier Transform (FFT). The 512-point FFT is applied to extract the information in the frequency domain. Then in the next step, Mel-filter wrapping is applied to model it as triangular filters using 26 Mel-filter banks, and the obtained output is considered as the power spectrum. The Mel-scale mimics the human ear perception of sound. The conversion of frequency (f) in Hz to Mel (m) and Mel (m) to frequency (f) is done by using Equations (5.1) and (5.2) respectively.

$$m = 2595 \log_{10}(1 + \frac{f}{700}) \tag{5.1}$$

$$f = 700(10^{(m/2595)} - 1) \tag{5.2}$$

The log operation is done on the output of the Mel-filter bank, and then Discrete Cosine Transform (DCT) is applied for logarithmically compressed Mel filter banks to obtain the MFCC features. These MFCC features are used for developing supervised machine learning models.

5.1.3 Classification Models

The different Machine Learning (ML) models are considered for solving the machine sound classification problem in the fog computing environment. These models are developed and deployed on the fog server (ICU/MDC) for malfunction machine detection based on their operating sounds in the industrial environment. Thus, it helps for fault detection and reduces severe machines' failure in the industry.

5.1.3.1 Random Forest (RF)

RF is a supervised ML model used for both classification and regression problems. RF creates the number of decision trees called forests on the randomly selected data points. Each tree returns the predictions of the considered classes in the problem. Then, RF selects the class through majority voting, i.e., the class with the maximum number of tree predictions will be regarded as the final prediction of the RF algorithm.

5.1.3.2 Support Vector Machine (SVM)

SVM is a popular supervised ML model for solving classification and regression problems. The main objective of the SVM algorithm is to create a hyperplane for segregating the given data in the best possible way. It selects the hyperplane with the maximum margin between the support vectors (data points). The multidimensional hyperplane represents the separation of the different classes considered in the problem.

5.1.3.3 Logistic Regression (LR)

The most common ML classification model for solving binary classification problems. It is the statistical method used to predict the binary class, and it is fast, uncomplicated, and convenient for solving the classification problem.

5.1.3.4 AdaBoost Classifier (AdaB)

It is the Adaptive boost classifier, combines the multiple classifiers to increase the accuracy of the developed classifier model. The general concept of the AdaB model is to set the weights of the classifier to train the data sample in each iteration such that it ensures the accurate predictions of the data samples. ML classification algorithms are combined with this if it accepts the weights on the training set to increase the accuracy rate; by default, it uses the decision tree classifier.

5.1.3.5 Multi-Layer Perceptron (MLP)

The MLP is a feed-forward Artificial Neural Network (ANN) model which is considered for solving the classification problems. MLP model consists of the input layer, an output layer, which predicts the output, and in-between these two, there are hidden layers with multiple perceptrons. MLP uses back-propagation for learning the model such that it reduces the cost function by changing the bias and weights.

5.2 Performance Evaluation

5.2.1 Experimental Setup

The cloud and fog computing configuration is considered to evaluate the developed supervised ML models for the machine's sound classification. In the cloud computing configuration, the recorded machine sound is transferred to the cloud server. It extracts features and then feeds them to the classification model to predict the machine sound as Abnormal or Normal. Then, send back the control signal to end devices to control the actuators based on the output of ML classifiers or store results (output of ML classifiers) in the cloud server if required for further analysis. But using a centralized cloud to process and analyze the machine sound might increase the service time. In fog computing configuration, devices such as ICU/MDC are considered as fog servers in the industrial environment. Using a fog server to process and analyze the machines' sound

will reduce the service time.

Table 5.1: Dataset details used for Experiment

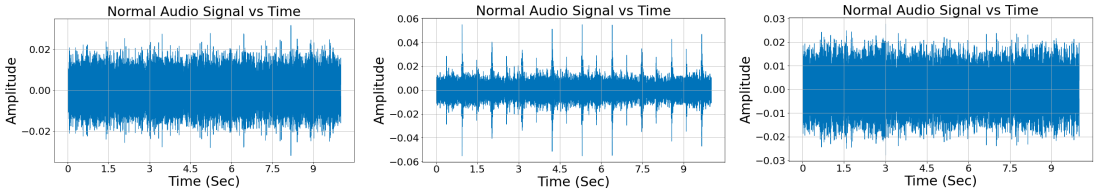
Machine Type	Operations	Anomalous Operations	Model ID	Number of samples	
				Normal	Anomalous
Pump	Discharge to water pool	Leakage, contanimation	id 00	922	162
Valve	Open/ Close repeat	More than two type of contamination	id 00	850	180
Fan	Normal Operation	Voltage change, clogging	id 00	647	235

The Raspberry Pi 3B+ model with CPU 1.4 GHz 64-bit quad-core, ARM Cortex-A53 CPU, 1 GB LPDDR2 SDRAM is used as the end node to collect the machines' sound and activate the actuator devices after receiving the control signal. The device with a 1.8 GHz Dual-Core Intel Core i5 processor with 8 GB, 1600 MHz DDR3 RAM is considered as the fog server in the industrial environment. The remote cloud server (in AWS Cloud) instance type is t2.medium with 2 vCPUS, 4 GB RAM, Intel 2.5 GHz processor is considered as the cloud server which runs the ubuntu 16.04 LTS OS. The recorded machine sound is stored in the end node and then transferred to the fog and cloud servers for further processing. The experiments are carried out using the developed machine learning models on the cloud and the fog server for the recorded audio samples under different Signal-to-Noise Ratio (SNR) levels.

5.2.2 Dataset

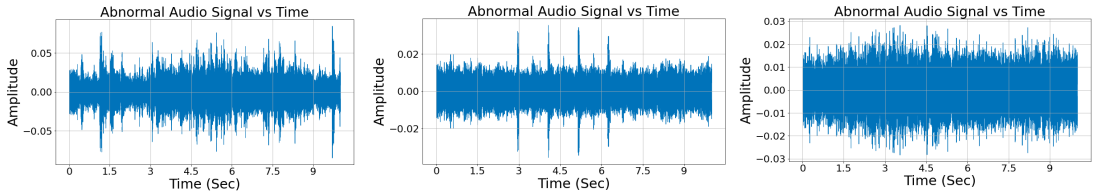
The machine sound dataset for malfunctioning industrial machine investigation and inspection (MIMII) (Purohit et al., 2019a,b) is used for monitoring and controlling the malfunctioning machines in the smart industrial environment. The dataset contains Normal and Anomalous sounds for Pump, Valve, Fan, and Slide Rails machines. The dataset includes 5000-10000 seconds of normal sound audio files and about 1000 seconds of anomalous/abnormal sound audio samples for each machine type. The unusual sounds such as leakage, contamination, rotating unbalance, and rail damage are considered abnormal sounds. The MIMII dataset assists for anomalous detection and ensures the control of these machine types based on the sound. These audio samples are

recorded at -6dB, 0dB, and 6dB SNR levels. The given dataset is collected for seven different types of machines represented by id 00 to id 06 and contains 26092 Normal sound files and 6065 Anomalous sound files. The Valve, Pump, and Fan machine sound are considered to conduct the experiment and classify the machine sound as normal and abnormal. This MIMII dataset is used in the experiment to detect and classify the Normal and Abnormal sounds using supervised machine learning techniques, and the complete details of the dataset used like machine type, model type, and the number of sound files, are given in Table 5.1.



(a) Pump (b) Valve (c) Fan

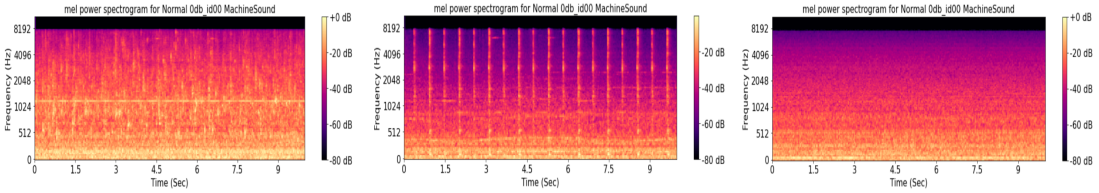
Figure 5.5: Normal Audio Signal Representation for different Machines under SNR= 0dB



(a) Pump (b) Valve (c) Fan

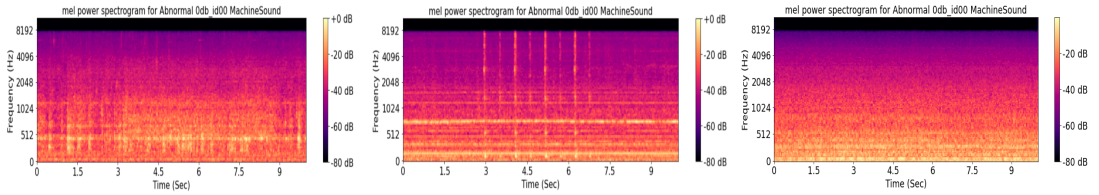
Figure 5.6: Abnormal Audio Signal Representation for different Machines under SNR=0dB

The waveform for both normal and abnormal sounds of different machines under SNR=0dB level is shown in the Figures 5.5 and 5.6, respectively. It shows the change in the amplitude over a period of time. The X-axis is the time duration of the audio signal recorded, and Y-axis represents the amplitude of the machine's sound for both normal and abnormal sounds recorded under the SNR=0dB level in the industrial environment. The Mel-power spectrogram for the different machine sounds at SNR= 0dB level is shown in the Figures 5.7 and 5.8. The X-axis in the spectrogram represents the time, and Y-axis represents the frequency of the sound in Hz. The Mel-power spectrogram for both normal and abnormal machine sounds at SNR=0dB level is shown in Figures 5.7 and 5.8, respectively.



(a) Pump (b) Valve (c) Fan

Figure 5.7: Normal Sound Power Spectrogram for different Machine types at SNR=0dB



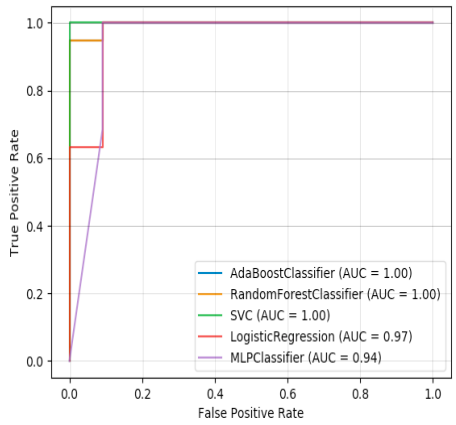
(a) Pump (b) Valve (c) Fan

Figure 5.8: Abnormal Sound Power Spectrogram for different Machine types at SNR=0dB

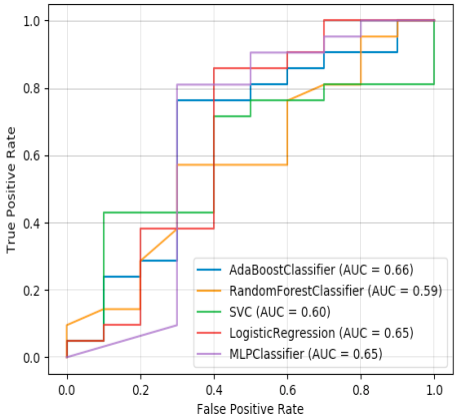
5.2.3 Results and Discussion

The different supervised ML classification models such as Logistic Regression (LR), Random Forest (RF), Support Vector Machine (SVM), AdaBoost (AdaB), and ANN-based Multi-Layer Perceptron (MLP) are deployed on fog server (ICU/MDC) to classify the machine sound in the industrial environment. For experiments, the data split of 80% and 20% is considered as the training and testing data, respectively, for the developed classification algorithms. The metrics for evaluating the performance of these classification models are: Accuracy, Precision, Recall, and F_measure score (F1_Score). Also, plotted the Receiver Operating Characteristic (ROC) curve for each classification model for different machine sounds recorded under different SNR levels.

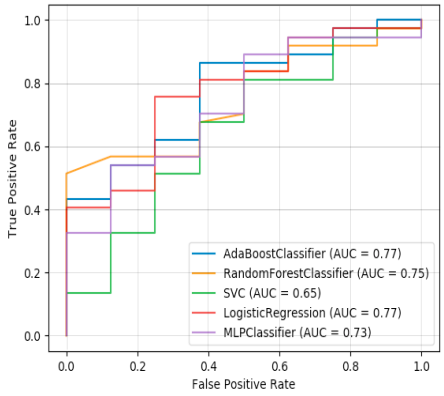
As the considered MIMII dataset is imbalanced, one of the ways to deal with the imbalanced dataset is by selecting the ROC curve and F1_Score as the performance evaluation metrics instead of using the Accuracy, as it can mislead the classifier model performance in the imbalanced learning scenarios (Boyle, 2019; He and Garcia, 2009). Hence, evaluated the developed classifier model's performance by plotting the ROC curve, F1_score, which is the weighted average of precision and recall for the developed ML classifier models (Davis and Goadrich, 2006). The ROC Curves for the deployed classifier models for LPC and MFCC features are given in Figures 5.9-5.14. The ROC curve shows the ability of these developed models to classify the machine sounds



(a) Pump



(b) Valve

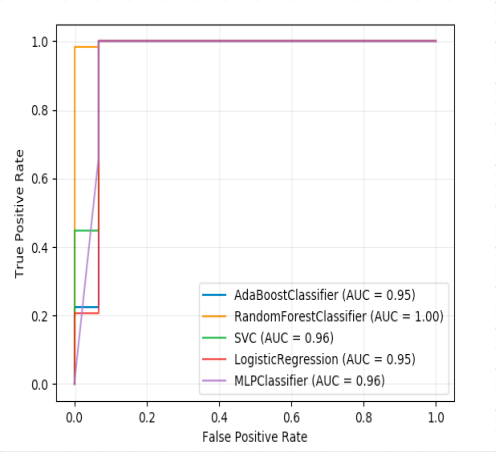


(c) Fan

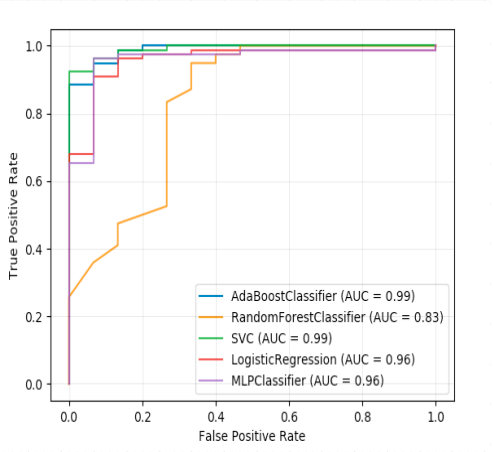
Figure 5.9: ROC curve for the ML models using LPC features for different Machines sound under SNR=-6dB

as normal and abnormal. The ROC curve is represented by using True Positive Rate (TPR) and False Positive Rate (FPR) as defined by using Equations (5.3) and (5.4). Figures 5.9-5.11 show the ROC curve for the deployed ML models using LPC features for classifying the machine sounds recorded under -6dB, 0dB, and 6dB SNR levels, respectively. It shows that the deployed model did not perform well for the Valve and Fan machine sound at -6dB level due to the noise in the audio samples.

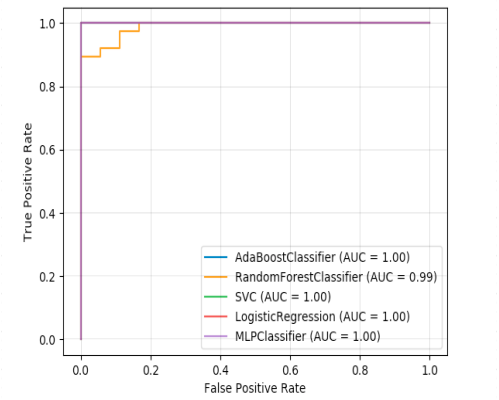
Figures 5.12-5.14 show the ROC curves for deployed ML models using MFCC features for the different machine sounds recorded at -6dB, 0dB, and 6dB SNR levels, respectively. From these figures, it is observed that the ML models are not able to classify the Valve correctly and Fan machine sound at SNR=-6dB level (Figure 5.12) due to the increase in the noise level compared to the other two SNR levels (Figures 5.13



(a) Pump



(b) Valve



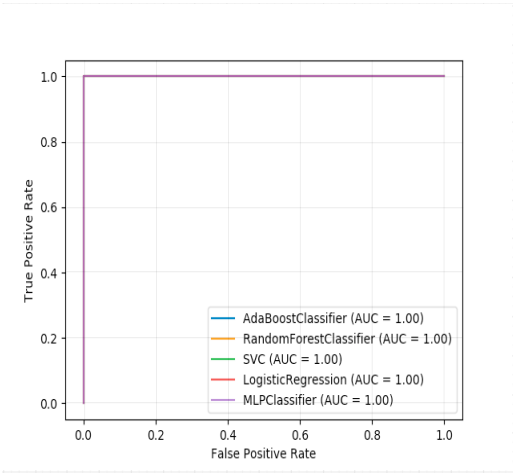
(c) Fan

Figure 5.10: ROC curve for the ML models using LPC features for different Machines sound under SNR=0dB

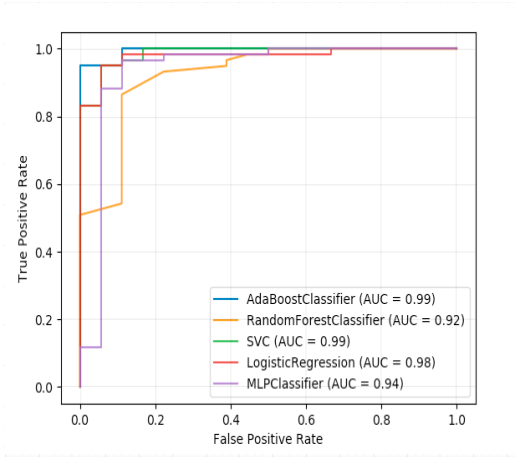
and 5.14). Further, the developed classifiers work better for the MFCC features when compared to the LPC features of the machine sounds. The Area Under the ROC curve (AUC) value indicates how best the developed classification model is and measures the developed model's prediction quality. The AUC value ranges from 0 to 1. The higher AUC value indicates the developed model works better, and the lower AUC value indicates the developed model is not good enough for the classification.

$$TPR = \frac{TruePositive}{TruePositive + FalseNegative} \tag{5.3}$$

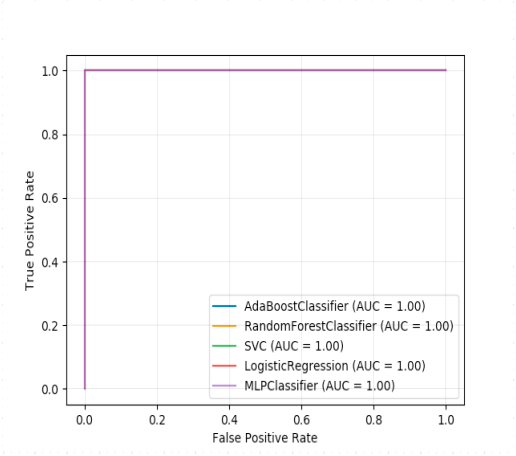
$$FPR = \frac{FalsePositive}{TrueNegative + FalsePositive} \tag{5.4}$$



(a) Pump



(b) Valve



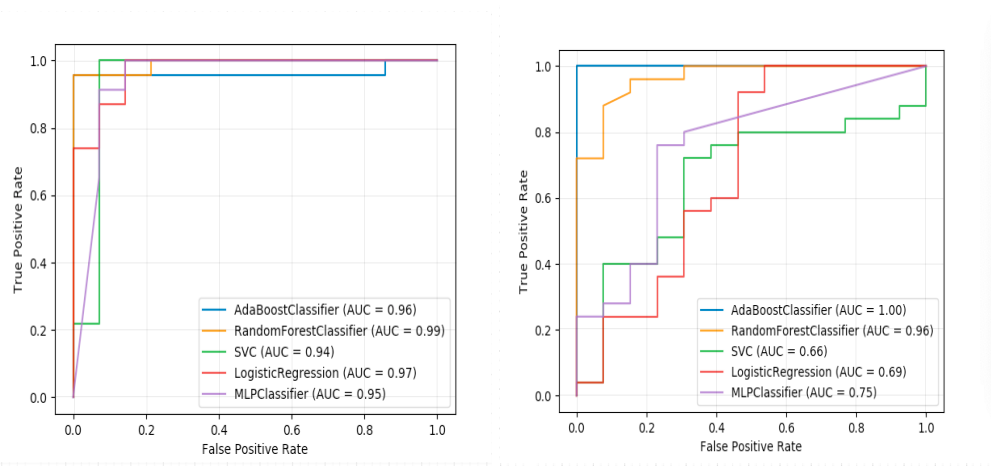
(c) Fan

Figure 5.11: ROC curve for the ML models using LPC features for different Machines sound under SNR=6dB

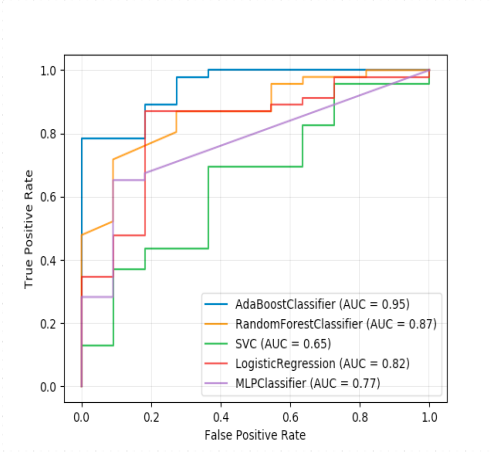
Figure 5.12 shows the AdaB classifier model works better for Valve and Fan machine sounds recorded at SNR=-6dB as compared to other models. Figure 5.13 explains AdaB classifier model works better for all the types of machine sounds as compared to the other models at the SNR=0dB level. Figure 5.14 shows ROC for all the models for SNR=6dB audio samples, where all of the classifier models work better as the noise level is less in the recorded audio samples to classify the machine sounds as normal and abnormal based on their operating sounds.

The other evaluation metrics considered are: Accuracy, Precision, Recall, and F1_Score as defined by using the Equations (5.5)-(5.8).

$$Accuracy = \frac{Number of Correct Predictions}{Total Number of Predictions} \tag{5.5}$$



(a) Pump (b) Valve



(c) Fan

Figure 5.12: ROC curve for the ML models using MFCC features for different Machines sound under SNR=-6dB

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{5.6}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{5.7}$$

$$F1\_Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{5.8}$$

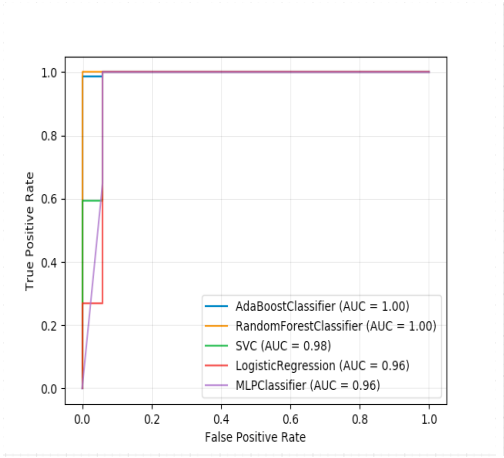
The Accuracy, Precision, Recall, and F1_Score values for the deployed ML models are calculated, and the complete details of the values obtained for each parameter for different ML models using LPC and MFCC features are given in Tables 5.2 and

Table 5.2: Performance of the Classifier Models using LPC Features for different types of Industrial Machines

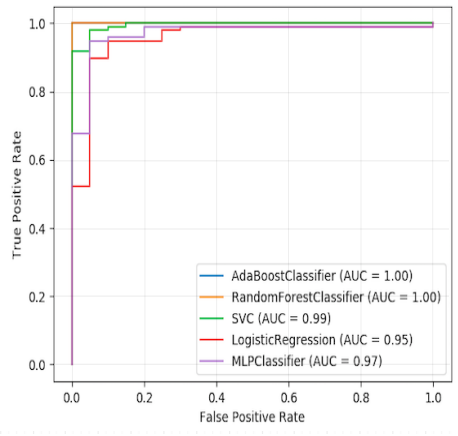
Machine Type	SNR level	Classifiers	Accuracy (%)	Precision (%)	Recall (%)	F1_Score (%)
Pump	-6dB	SVM	83.47	91.66	52.5	66.80
		RF	96.66	97.5	95.45	96.33
		LR	84.41	91.89	60.0	72.25
		AdaB	96.66	97.5	95.45	**96.33**
		MLP	96.66	97.5	95.45	**96.33**
	0dB	SVM	85.71	92.46	63.33	74.97
		RF	98.63	99.15	96.66	97.84
		LR	93.50	90.38	91.90	91.10
		AdaB	98.63	99.15	96.66	**97.84**
		MLP	98.63	99.15	96.66	**97.84**
	6dB	SVM	96.52	93.94	93.94	93.94
		RF	97.39	96.32	94.47	95.36
		LR	96.10	92.93	95.05	93.94
		AdaB	97.39	96.32	94.47	**95.36**
		MLP	97.45	96.32	94.47	**95.36**
Valve	-6dB	SVM	82.60	41.30	50.0	45.23
		RF	61.29	50.66	50.47	49.45
		LR	80.51	40.25	50.0	44.60
		AdaB	67.74	61.0	57.85	**57.88**
		MLP	74.19	74.53	62.61	**63.09**
	0dB	SVM	83.47	91.66	52.5	66.80
		RF	91.39	95.34	73.33	79.37
		LR	93.50	90.38	91.90	**91.10**
		AdaB	92.47	88.58	82.05	84.83
		MLP	95.69	90.51	94.74	**92.45**
	6dB	SVM	91.30	95.23	75.0	80.83
		RF	84.41	91.54	66.66	70.38
		LR	93.50	90.38	91.90	**91.10**
		AdaB	84.41	85.77	68.59	72.23
		MLP	92.20	92.46	85.26	**88.16**
Fan	-6dB	SVM	82.60	41.30	50.0	45.23
		RF	82.22	68.58	64.69	**66.16**
		LR	91.30	95.23	75.0	**80.83**
		AdaB	75.55	59.72	60.64	60.11
		MLP	73.33	60.60	64.18	61.42
	0dB	SVM	96.52	93.94	93.94	**93.94**
		RF	94.64	94.49	93.12	93.76
		LR	86.95	93.18	62.5	66.34
		AdaB	94.64	94.49	93.12	**93.76**
		MLP	94.0	96.05	90.0	92.38
	6dB	SVM	97.39	96.32	94.47	95.36
		RF	96.52	93.94	93.94	93.94
		LR	86.95	93.18	62.5	74.34
		AdaB	97.40	95.86	95.86	**95.86**
		MLP	96.42	95.0	97.36	**96.01**

Table 5.3: Performance of the Classifier Models using MFCC Features for different types of Industrial Machines

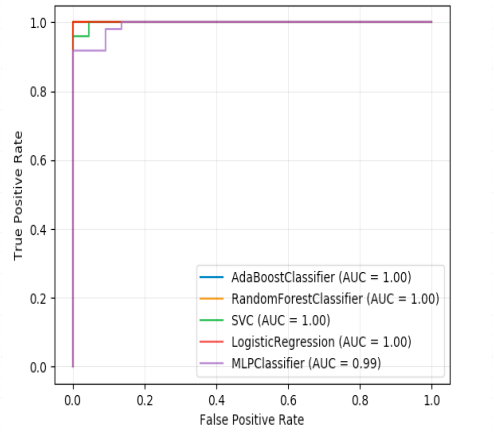
Machine Type	SNR level	Classifiers	Accuracy (%)	Precision (%)	Recall (%)	F1_Score (%)
Pump	-6dB	SVM	91.11	93.93	87.5	89.63
		RF	93.33	92.82	92.82	92.82
		LR	96.66	97.5	95.45	96.33
		AdaB	96.66	95.83	97.36	**96.47**
		MLP	96.66	97.5	95.45	**96.33**
	0dB	SVM	96.33	97.89	88.88	92.67
		RF	98.63	99.15	96.66	97.84
		LR	98.63	99.15	96.66	97.84
		AdaB	98.63	99.15	96.66	**97.84**
		MLP	98.90	99.33	97.05	**98.14**
	6dB	SVM	95.08	97.0	89.28	92.45
		RF	96.72	97.95	92.85	**95.11**
		LR	87.91	93.20	73.80	78.61
		AdaB	96.70	97.94	92.85	95.10
		MLP	98.16	98.92	94.44	**96.51**
Valve	-6dB	SVM	67.74	33.87	50.0	40.38
		RF	61.29	50.66	50.47	49.45
		LR	74.19	71.33	65.23	66.30
		AdaB	87.09	92.0	80.0	**83.15**
		MLP	92.10	94.64	88.46	**90.64**
	0dB	SVM	93.54	86.16	93.46	89.20
		RF	92.47	95.88	76.66	82.63
		LR	94.62	89.32	91.41	90.32
		AdaB	95.69	90.51	94.74	**92.45**
		MLP	97.84	94.11	98.71	**96.22**
	6dB	SVM	96.10	93.87	95.52	94.66
		RF	97.40	96.37	96.37	96.37
		LR	93.50	90.38	91.90	91.10
		AdaB	98.70	99.16	97.22	**98.15**
		MLP	97.91	96.95	96.95	**96.95**
Fan	-6dB	SVM	73.52	36.76	50.0	42.37
		RF	86.66	66.25	93.02	74.45
		LR	82.22	71.96	79.39	74.28
		AdaB	91.11	84.79	84.79	**84.79**
		MLP	91.22	86.80	84.18	**85.40**
	0dB	SVM	93.54	86.16	93.46	89.20
		RF	94.62	89.32	91.41	90.32
		LR	94.62	89.32	91.41	90.32
		AdaB	95.69	90.51	94.74	**92.45**
		MLP	98.80	97.91	99.18	**98.52**
	6dB	SVM	96.10	93.87	95.52	94.66
		RF	96.72	97.95	92.85	95.11
		LR	93.50	90.38	91.90	91.10
		AdaB	97.40	96.37	96.37	**96.37**
		MLP	98.80	97.91	99.18	**98.52**



(a) Pump



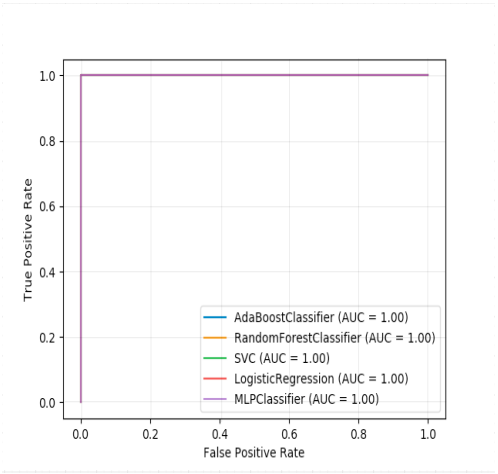
(b) Valve



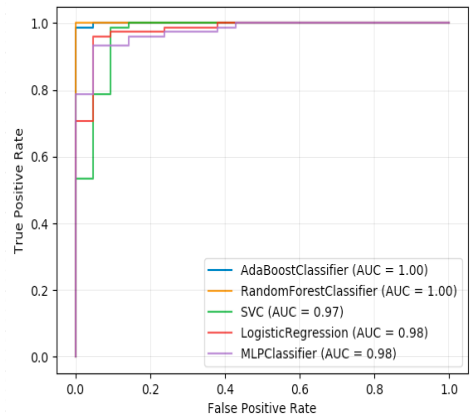
(c) Fan

Figure 5.13: ROC curve for the ML models using MFCC features for different Machines sound under SNR=0dB

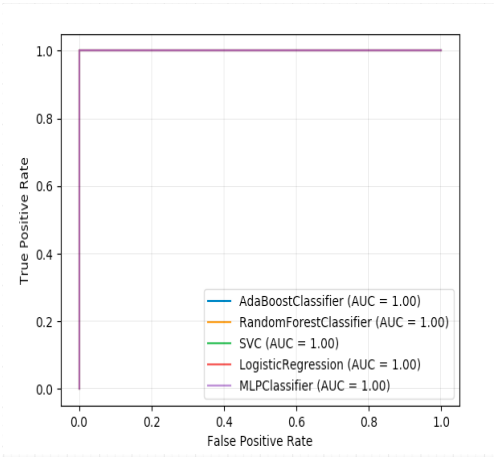
5.3, respectively. The F1_Score is considered to measure the performance of developed models as the considered dataset is imbalanced (Brownlee, 2015, 2014). The experimental results show that AdaB and MLP work better using MFCC features of -6dB and 0dB SNR levels of Pump machine sounds, and for Valve, Fan machines sound recorded under -6dB, 0dB, and 6dB SNR levels in terms of F1_Score. The two best performing ML models at different SNR levels are highlighted in the Tables 5.2 and 5.3. Classifying the machine sounds like the normal and abnormal sounds in the industrial environment helps for fault detection and machine malfunction monitoring. The obtained experimental results can be considered as the benchmark results. Since fog-based machine malfunction monitoring using machine learning models is essential to classify the machine sounds as normal and abnormal in the Industry 4.0/smart industrial



(a) Pump



(b) Valve



(c) Fan

Figure 5.14: ROC curve for the ML models using MFCC features for Different Machines sound under SNR=6dB

environment.

The cloud and fog computing architectures are used independently for comparing the classification time (including both communication and analyzing) to process and classify the machine sound samples as abnormal and normal. The ML models are deployed in the cloud server to analyze and classify the machine's sound as normal and abnormal and record the total classification time. In the cloud computing configuration, the recorded machine sound is transferred to the cloud, where it is processed and analyzed and then sends back the control signal to end devices. In fog computing configuration, the machine sound is transferred to the fog server. It processes and analyzes the machine sound to classify machines' sounds as normal or abnormal. The time to classify the audio samples as normal and abnormal is calculated for 5, 10, 15, and 20

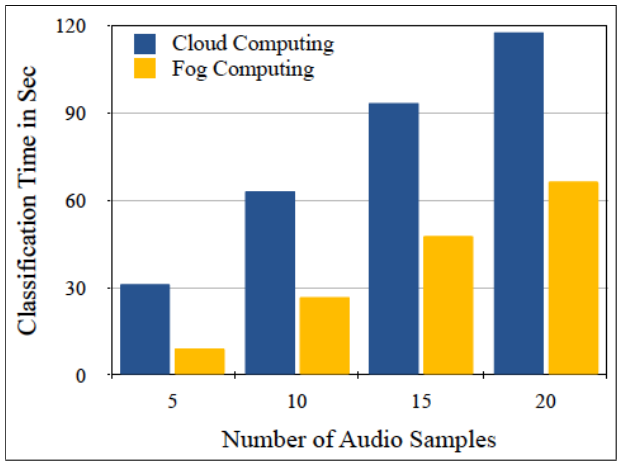


Figure 5.15: Comparison of Classification Time in Fog and Cloud

audio samples using the cloud and fog computing architectures. The single file size is varied from 2.6 MB to 3 MB and the bandwidth speed is approaximately 2.8-5 Mbps for the experimental evaluation. Figure 5.15 illustrates the comparison of the classification time using the fog and cloud computing architectures. The experimental results found that using a remote cloud server to process the machine sound takes more time than the fog computing configuration. Thus, using fog computing architecture to process machine sounds in the Industry 4.0 environment will reduce the classification time ($\approx$50%) and thus minimizes the significant failures of industrial machines.

5.2.4 Statistical Hypothesis Analysis

The statistical hypothesis analysis is carried out for the developed classification models using a t-test. The hypothesis testing compares the best performing classification models with the other developed classification models for LPC, and MFCC audio features considering the F1_Score value. For the t-test, the threshold value considered for the p-Value is 0.05. From the t-test analysis, it is found that the p-values for the models are less than the threshold value of 0.05, as shown in Table 5.4. From Table 5.4 it is observed that p-values are less than 0.05 for MLP, and it thus rejects the null hypothesis. Hence, among the developed classification model, MLP performs better than other classification models in the fog server-based framework.

5.2.5 Time Complexity

The time complexity of the above classification models depends on, number of training samples (m), the number of features (k), the number of trees (P_{tree}), number of support

Table 5.4: p-values for Classification Models

Feature Type	Classification Models	p-values
LPC	SVM-MLP	0.000013
	RF-MLP	0.000010
	LR-MLP	0.003694
	AdaB-MLP	0.000346
MFCC	SVM-MLP	0.000102
	RF-MLP	0.000010
	LR-MLP	0.000013
	AdaB-MLP	0.000010

vectors (n_{sv}), the number of iterations (M). In AdaB classifier M is number of times that the additional model is called. The time complexity for MLP algorithm depends on number of hidden layers (l), number of perceptrons (h) in each layer. The training and testing time complexity for all the above ML classifier models is given in Table 5.5.

Table 5.5: Time Complexity of Classification Models

Classification Model	Training Time Complexity	Testing Time Complexity
SVM	m^2k	kn_{sv}
RF	m^2kP_{tree}	kP_{tree}
LR	kmM	k
AdaB	m^2kM	kP_{tree}M
MLP	mkh^lM	kh^lM

5.2.6 Limitations of the Work

The industrial environment's network and industrial controller units can be used as the fog server to deploy the machine learning models to analyze the IoT/IIoT data. The developed fog server prototype can be deployed in the industrial environment to enable the real-time monitoring of the malfunctioning machines based on the machines' operating sounds. This prototype can be extended and deployed on resource constrained devices using the container-based two-level (Fog Master and Cell node) architecture to analyze data. Further, the service placement strategies will be evaluated with the various workloads in the developed fog frameworks. The above limitations can be addressed as part of the future work on extending and deploy the developed prototype in the industrial environment to monitor malfunctioning machines in real-time.

5.3 Summary

This chapter discusses the use of fog computing architecture for Industry 4.0/ Smart Industry environment. The fog server framework is deveoped as a prototype for the intelligent machine malfunction monitoring system based on the machine operating sounds to identify and classify the machines as normal and abnormal. The LPC and MFCC features are extracted from the machine sound samples in the fog computing environment. The different supervised ML classification algorithms are developed and deployed on the fog server in the manufacturing industrial environment. The deployed ML models can classify the different machine sounds as normal and abnormal, recorded under -6dB, 0dB, and 6dB SNR levels. Also, using fog computing for industrial machine monitoring minimizes the total service time and thus enables real-time monitoring, which avoids the significant machines failures in the Smart Industry/Industry 4.0 environment. The conclusions and the future directions of this research work are discussed in the next chapter.

Chapter 6

Conclusions and Future Directions

6.1 Conclusions

Nowadays, the use of IoT devices is increased exponentially and thus generates an enormous amount of data. Using a centralized cloud for all types of IoT applications is not feasible. It might increase the service time and resource cost due to increased communication time and more network resources usage. Fog Computing is the driving force for addressing these issues and service delay-sensitive IoT applications in real-time, reducing the service cost, and avoiding dangerous failures in the different smart environments. The primary challenge is realizing the fog computing environment using the cloud computing features and the virtualization technique on the resource constrained devices at the network edge. The resource constrained fog nodes are used for IoT service placement and data analytics to minimize the service time, network resource usage and service cost for various smart environments.

The Fog-Cloud computing environment is used to host the IoT applications requests on the Fog-Cloud computing environment such that the QoS of the IoT applications are satisfied. The existing FFD approach is applied to place the services based on the resource available in the fog nodes. Further, the service placement problem in the fog-cloud computing environment is formulated as a multi-objective optimization problem and a novel cost-efficient Deadline-Aware Service Placement (DASP) strategy is proposed to place the IoT/IIoT application services on the Fog-Cloud computing architecture with various network configurations in the fog layer. The experimental results show that the DASP service placement strategy performs better than the FFD, state-of-the-art service placement strategies, and CloudOnly approach considered for the performance evaluation in the Fog-Cloud computing environment. However, the dynamic characteristics of the network topology configurations, the mobility of the devices, and the device failures are not considered for deploying the IoT/IIoT application services in the Fog-Cloud computing environment.

Realizing fog architecture on the resource-constrained devices to provide the computational resources to host and process the service requests is regarded as one of the

main challenges in the fog computing environment. Using simulations and VMs based resource provisioning fog framework on resource constrained devices may not be efficient since it takes more time for booting and consumes more physical machine resources. Hence, a two-level fog computing framework is developed using the docker and containers on 1.4 GHz 64-bit quad-core processor devices. Further, the service placement problem is formulated as the multi-objective optimization problem to minimize the service time, cost, and energy consumption in the fog computing environment. The various meta-heuristic-based service placement strategies are developed and evaluated on the fog framework consisting of twenty fog nodes. The experimental results show that the proposed hybrid EGAPSO based service placement algorithm performs better than the proposed (EGA and MGAPSO) and state-of-the-art service placement strategies in the two-level fog computing environment.

The key limitations of this research contribution are as follows:

- The inter-dependent IoT/IIoT application services are not considered to check the performance of the proposed service placement strategies on the developed multi-level fog testbed.
- Handling the device failures in the developed multi-level fog architecture is not considered.
- Load balancing in the fog computing environment is not considered.

The fog nodes are resource constrained and deploying the machine learning models on a resource constrained devices is a challenge. Hence, the cost-efficient fog server-based framework is developed as a prototype for intelligent machine malfunction monitoring in the industrial environment. The fog server architecture analyzes the machine sounds to detect and monitor the malfunctioning machines in the Smart Industry/Industry 4.0 environment. The various supervised machine learning models are developed and deployed on the fog server to classify the machines' sounds as normal and abnormal. The different machine sound recorded at different SNR levels are used to detect and classify the machines as normal and abnormal. The experimental results show that the developed models could correctly classify the machine's sound recorded under the SNR=6dB level. Also, it is observed that the ML model's performance is superior for using the MFCC machine sound features compared to the LPC features of machines sound of any SNR levels in the fog computing environment. Thus, using a

fog computing environment to process the IoT/IIoT data at the network edge level will minimize the service time and reduce the service cost, network resource consumption, and avoid critical failures in the smart industrial environment.

The key limitations of this research contribution are as follows:

- The developed service placement algorithms are not considered to evaluate the different workloads (audio and video) in the developed fog node and fog server architecture.
- The noise separation from the audio samples is not considered.
- Using distributed ML techniques on the fog server for analysing the machine operating sound is not considered.

6.2 Future Directions

The crucial future directions to address the key limitations of the thesis contributions are as follows:

1. **Resource Provisioning and Service Placement in Fog-Cloud Environment**
   - Develop the Quality of Experience-based service placement strategies in the multi-level fog computing environment to optimize the fog nodes' service time, cost, energy consumption, and resource usage.
   - Design and develop the inter-dependent IoT/IIoT application model and then place the inter-dependent applications modules/service on the multi-level fog computing architecture. Based on the application requirement, design the inter-dependent applications and deploy the applications in the multi-level fog computing environment using the heuristic, meta-heuristic, or game theory-based approaches to provide the service in real-time.
2. **Handling Fog Nodes' Failure and the Service Migration**
   - Develop the fault tolerance mechanism and thus handle the mobility of the devices in the Fog environment
   - Load balancing and the service migrations between the fog nodes in the multi-level fog computing environment. Developing energy-aware, performance aware, and learning-based techniques for load balancing and service migrations in the multi-level fog computing environment.
   - Develop the learning-based workload prediction and allocation strategies in the fog computing environment to achieve low latency and efficient utilization of the fog computing resources. Using ML-based techniques for workload prediction and allocation based on the history of resource usage and the workload allocation in the fog computing environment.

3. **Data Security in the Fog Computing Environment**

   - Ensure the data security in the multi-level fog computing architecture. Developing the secure authentication mechanisms and the secured protocols to identify the malicious fog node, ensure the data privacy in the multi-level fog computing environment.

4. **Fog based Real-time Data Analytics in Smart Environments**

   - To deploy the developed fog server framework in the industrial environment to identify the malfunctioning machines based on their operating sounds.

   - Extend the developed prototype on the container-based two level fog framework to deploy on the resource constrained devices in the industrial environment to analyse the machine data.

   - Using developed service palcement strategies to evaluate the different workloads (audio and video) in the fog server-based frameworks.

   - Deploying the deep learning models using containers on resource constrained devices for distributed data analytics for the smart environments.
   - Video-based intelligent machine malfunctioning monitoring using the fog computing infrastructure in the Smart Industrial Environment. The deep learning techniques to analyze the videos on the fog nodes of the industrial environment and identify the malfunctioning machines in the industry.

   - Use linear or non-liner filters for noise separation from the audio signals and analyse the machine sounds to classify the sounds as normal and abnormal.

   - Orchestrator development for controlling and managing the Containerized Applications in the Fog Computing Environment.

References

Aazam, M. and Huh, E.-N. (2014). "Fog computing and smart gateway based communication for cloud of things". In *2014 International Conference on Future Internet of Things and Cloud*, 464–470. IEEE.

Aazam, M. and Huh, E.-N. (2015a). "Dynamic resource provisioning through fog micro datacenter". In *2015 IEEE international conference on pervasive computing and communication workshops (PerCom workshops)*, 105–110. IEEE.

Aazam, M. and Huh, E.-N. (2015b). "Fog computing micro datacenter based dynamic resource estimation and pricing model for IoT". In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*, 687–694. IEEE.

Aazam, M., St-Hilaire, M., Lung, C.-H., and Lambadaris, I. (2016). "Cloud-based smart waste management for smart cities". In *2016 IEEE 21st International Workshop on Computer Aided Modelling and Design of Communication Links and Networks (CAMAD)*, 188–193. IEEE.

Aazam, M., Zeadally, S., and Harras, K. A. (2018a). "Deploying fog computing in industrial internet of things and industry 4.0". *IEEE Transactions on Industrial Informatics*, *14*(10), 4674–4682.

Aazam, M., Zeadally, S., and Harras, K. A. (2018b). "Fog computing architecture, evaluation, and future research directions". *IEEE Communications Magazine*, *56*(5), 46–52.

Aazam, M., Zeadally, S., and Harras, K. A. (2018c). "Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities". *Future Generation Computer Systems*, *87*, 278–289.

Abdulkareem, K. H., Mohammed, M. A., Gunasekaran, S. S., Al-Mhiqani, M. N., Mutlag, A. A., Mostafa, S. A., Ali, N. S., and Ibrahim, D. A. (2019). "A review of Fog computing and machine learning: Concepts, applications, challenges, and open issues". *IEEE Access*, *7*, 153123–153140.

Aceto, G., Persico, V., and Pescape, A. (2019). "A survey on information and communication technologies for Industry 4.0: state-of-the-art, taxonomies, perspectives, and challenges". *IEEE Communications Surveys and Tutorials*, *21*(4), 3467–3501.

Al-Turjman, F. and Malekloo, A. (2019). "Smart parking in IoT-enabled cities: A survey". *Sustainable Cities and Society*, *49*, 101608.

Alam, M. S., Natesha, B., Ashwin, T., and Guddeti, R. M. R. (2019). "UAV based cost-effective real-time abnormal event detection using edge computing". *Multimedia Tools and Applications*, *78*(24), 35119–35134.

Alavi, A. H., Jiao, P., Buttlar, W. G., and Lajnef, N. (2018). "Internet of Things-enabled smart cities: State-of-the-art and future trends". *Measurement*, *129*, 589–606.

Alim, S. A. and Rashid, N. K. A. (2018). "Some commonly used speech feature extraction algorithms". *From Natural to Artificial Intelligence-Algorithms and Applications*.

Alwasel, K., Jha, D. N., Habeeb, F., Demirbaga, U., Rana, O., Baker, T., Dustdar, S., Villari, M., James, P., Solaiman, E., et al. (2020). "IoTSim-Osmosis: A framework for modelling and simulating IoT applications over an edge-cloud continuum". *Journal of Systems Architecture*, 101956.

Ananthanarayanan, G., Bahl, P., Bodík, P., Chintalapudi, K., Philipose, M., Ravindranath, L., and Sinha, S. (2017). "Real-time video analytics: The killer app for edge computing". *computer*, *50*(10), 58–67.

Arasteh, H., Hosseinnezhad, V., Loia, V., Tommasetti, A., Troisi, O., Shafie-khah, M., and Siano, P. (2016). "Iot-based smart cities: a survey". In *2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC)*, 1–6. IEEE.

Ashjaei, M. and Bengtsson, M. (2017). "Enhancing smart maintenance management using fog computing technology". In *2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, 1561–1565. IEEE.

Atzori, L., Iera, A., and Morabito, G. (2010). "The internet of things: A survey". *Computer networks*, *54*(15), 2787–2805.

Baker, B. S. (1985). "A new proof for the first-fit decreasing bin-packing algorithm". *Journal of Algorithms*, *6*(1), 49–70.

Baker, S. B., Xiang, W., and Atkinson, I. (2017). "Internet of things for smart healthcare: Technologies, challenges, and opportunities". *IEEE Access*, *5*, 26521–26544.

Baldini, I., Castro, P., Chang, K., Cheng, P., Fink, S., Ishakian, V., Mitchell, N., Muthusamy, V., Rabbah, R., Slominski, A., et al. (2017). "Serverless computing: Current trends and open problems". In *Research advances in cloud computing* 1–20. Springer.

Battula, S. K., O'Reilly, M. M., Garg, S., and Montgomery, J. (2020). "A Generic Stochastic Model for Resource Availability in Fog Computing Environments". *IEEE Transactions on Parallel and Distributed Systems*.

Beloglazov, A., Buyya, R., Lee, Y. C., and Zomaya, A. (2011). "A taxonomy and survey of energy-efficient data centers and cloud computing systems". In *Advances in computers*, volume 82 47–111. Elsevier.

Bonomi, F., Milito, R., Natarajan, P., and Zhu, J. (2014). "Fog computing: A platform for internet of things and analytics". In *Big data and internet of things: A roadmap for smart environments* 169–186. Springer.

Bonomi, F., Milito, R., Zhu, J., and Addepalli, S. (2012). "Fog computing and its role in the internet of things". In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 13–16.

Botta, A., De Donato, W., Persico, V., and Pescapé, A. (2016). "Integration of cloud computing and internet of things: a survey". *Future generation computer systems*, *56*, 684–700.

Boyle, T. (2019), "Dealing with Imbalanced Data". https://towardsdatascience.com/methods-for-dealing-with-\imbalanced-data-5b761be45a18. [Online; last accessed 30-Oct-2020].

Bozorgchenani, A., Disabato, S., Tarchi, D., and Roveri, M. (2020). "An energy harvesting solution for computation offloading in Fog Computing networks". *Computer Communications*, *160*, 577–587.

Brogi, A., Forti, S., Guerrero, C., and Lera, I. (2020). "How to place your apps in the fog: State of the art and open challenges". *Software: Practice and Experience*, *50*(5), 719–740.

Brownlee (2014), "Classification Accuracy is Not Enough: More Performance Measures You Can Use". https://machinelearningmastery.com/classification-accuracy-is-not-enough-more-\performance-measures-you-can-use. [Online; last accessed 30-Oct-2020].

Brownlee, J. (2015), "8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset". https://machinelearningmastery.com/tactics-to-combat-imbalanced\-classes-in-your-machine-learning-dataset. [Online; last accessed 30-Oct-2020].

Buyya, R., Broberg, J., and Goscinski, A. M. (2010). "Cloud computing: Principles and paradigms", volume 87. John Wiley and Sons.

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., and Brandic, I. (2009). "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility". *Future Generation computer systems*, *25*(6), 599–616.

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., and Buyya, R. (2011). "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms". *Software: Practice and experience*, *41*(1), 23–50.

Canali, C. and Lancellotti, R. (2019). "GASP: Genetic algorithms for service placement in fog computing systems". *Algorithms*, *12*(10), 201.

Catarinucci, L., De Donno, D., Mainetti, L., Palano, L., Patrono, L., Stefanizzi, M. L., and Tarricone, L. (2015). "An IoT-aware architecture for smart healthcare systems". *IEEE internet of things journal*, *2*(6), 515–526.

Chamberlain, D. (2018), "Containers vs. Virtual Machines (VMs): What's the Difference?". https://blog.netapp.com/blogs/containers-vs-vms/. [Online; last accessed 17-Dec-2020].

Chang, C., Srirama, S. N., and Buyya, R. (2017). "Indie fog: An efficient fog-computing infrastructure for the internet of things". *Computer*, *50*(9), 92–98.

Chatterjee, S., Kar, A. K., and Gupta, M. (2018). "Success of IoT in smart cities of India: An empirical analysis". *Government Information Quarterly*, *35*(3), 349–361.

Chen, N., Chen, Y., You, Y., Ling, H., Liang, P., and Zimmermann, R. (2016). "Dynamic urban surveillance video stream processing using fog computing". In *2016 IEEE second international conference on multimedia big data (BigMM)*, 105–112. IEEE.

Chiu, T.-C., Pang, A.-C., Chung, W.-H., and Zhang, J. (2018). "Latency-driven fog cooperation approach in fog radio access networks". *IEEE Transactions on Services Computing*, *12*(5), 698–711.

Choudhary, V. (2007). "Software as a service: Implications for investment in software development". In *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*, 209a–209a. IEEE.

Concone, F., Re, G. L., and Morana, M. (2019). "A fog-based application for human activity recognition using personal smart devices". *ACM Transactions on Internet Technology (TOIT)*, *19*(2), 1–20.

Constant, N., Borthakur, D., Abtahi, M., Dubey, H., and Mankodiya, K. (2017). "Fog-assisted wiot: A smart fog gateway for end-to-end analytics in wearable internet of things". *arXiv preprint arXiv:1701.08680*.

Dastjerdi, A. V. and Buyya, R. (2016). "Fog computing: Helping the Internet of Things realize its potential". *Computer*, *49*(8), 112–116.

David (2014). "Containers and cloud: From lxc to docker to kubernetes". *IEEE Cloud Computing*, *1*(3), 81–84.

Davis, J. and Goadrich, M. (2006). "The relationship between Precision-Recall and ROC curves". In *Proceedings of the 23rd international conference on Machine learning*, 233–240.

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.

Desikan, K. S., Srinivasan, M., and Murthy, C. S. R. (2017). "A novel distributed latency-aware data processing in fog computing-enabled iot networks". In *Proceedings of the ACM Workshop on Distributed Information Processing in Wireless Networks*, 1–6.

Dhanvijay, M. M. and Patil, S. C. (2019). "Internet of Things: A survey of enabling technologies in healthcare and its applications". *Computer Networks*, *153*, 113–131.

Diez-Olivan, A., Del Ser, J., Galar, D., and Sierra, B. (2019). "Data fusion and machine learning for industrial prognosis: Trends and perspectives towards Industry 4.0". *Information Fusion*, *50*, 92–111.

Diro, A. A. and Chilamkurti, N. (2018). "Distributed attack detection scheme using deep learning approach for Internet of Things". *Future Generation Computer Systems*, *82*, 761–768.

Djemai, T., Stolf, P., Monteil, T., and Pierson, J.-M. (2019). "A Discrete Particle Swarm Optimization approach for Energy-efficient IoT services placement over Fog infrastructures". In *2019 18th International Symposium on Parallel and Distributed Computing (ISPDC)*, 32–40. IEEE.

Etemadi, M., Ghobaei-Arani, M., and Shahidinejad, A. (2020). "Resource provisioning for IoT services in the fog computing environment: An autonomic approach". *Computer Communications*, *161*, 109–131.

Fayos-Jordan, R., Felici-Castell, S., Segura-Garcia, J., Lopez-Ballester, J., and Cobos, M. (2020). "Performance comparison of container orchestration platforms with low cost devices in the fog, assisting Internet of Things applications". *Journal of Network and Computer Applications*, *169*, 102788.

Felter, W., Ferreira, A., Rajamony, R., and Rubio, J. (2015). "An updated performance comparison of virtual machines and linux containers". In *2015 IEEE international symposium on performance analysis of systems and software (ISPASS)*, 171–172. IEEE.

Forti, S., Pagiaro, A., and Brogi, A. (2020). "Simulating FogDirector Application Management". *Simulation Modelling Practice and Theory*, *101*, 102021.

Foukalas, F. (2020). "Cognitive IoT platform for fog computing industrial applications". *Computers and Electrical Engineering*, *87*, 106770.

Fox, G. C., Ishakian, V., Muthusamy, V., and Slominski, A. (2017). "Status of serverless computing and function-as-a-service (faas) in industry and research". *arXiv preprint arXiv:1708.08028*.

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".

Garg, S. K., Versteeg, S., and Buyya, R. (2013). "A framework for ranking of cloud computing services". *Future Generation Computer Systems*, *29*(4), 1012–1023.

Ghanavati, S., Abawajy, J. H., and Izadi, D. (2020). "An Energy Aware Task Scheduling Model Using Ant-Mating Optimization in Fog Computing Environment". *IEEE Transactions on Services Computing*.

Ghosh, R., Longo, F., Xia, R., Naik, V. K., and Trivedi, K. S. (2013). "Stochastic model driven capacity planning for an infrastructure-as-a-service cloud". *IEEE Transactions on Services Computing*, *7*(4), 667–680.

Goudarzi, M., Wu, H., Palaniswami, M. S., and Buyya, R. (2020). "An application placement technique for concurrent iot applications in edge and fog computing environments". *IEEE Transactions on Mobile Computing*.

Goudarzi, S., Anisi, M. H., Ahmadi, H., and Mousavian, L. (2020). "Dynamic Resource Allocation Model for Distribution Operations using SDN". *IEEE Internet of Things Journal*.

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.

Gupta, L., Jain, R., and Chan, H. A. (2016). "Mobile edge computing–an important ingredient of 5g networks". *IEEE Software Defined Networks Newsletter*.

Hassan, H. O., Azizi, S., and Shojafar, M. (2020). "Priority, network and energy-aware placement of IoT-based application services in fog-cloud environments". *IET Communications*, *14*(13), 2117–2129.

He, H. and Garcia, E. A. (2009). "Learning from imbalanced data". *IEEE Transactions on knowledge and data engineering*, *21*(9), 1263–1284.

He, J., Wei, J., Chen, K., Tang, Z., Zhou, Y., and Zhang, Y. (2017). "Multitier fog computing with large-scale iot data analytics for smart cities". *IEEE Internet of Things Journal*, *5*(2), 677–686.

He, S., Guo, L., Guo, Y., Wu, C., Ghanem, M., and Han, R. (2012). "Elastic application container: A lightweight approach for cloud resource provisioning". In *2012 IEEE 26th International Conference on Advanced Information Networking and Applications*, 15–22. IEEE.

He, T., Khamfroush, H., Wang, S., La Porta, T., and Stein, S. (2018). "It's hard to share: Joint service placement and request scheduling in edge clouds with sharable and non-sharable resources". In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, 365–375. IEEE.

Hoque, S., De Brito, M. S., Willner, A., Keil, O., and Magedanz, T. (2017). "Towards container orchestration in fog computing infrastructures". In *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, 294–299. IEEE.

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.

Iqbal, R., Maniak, T., Doctor, F., and Karyotis, C. (2019). "Fault detection and isolation in industrial processes using deep learning approaches". *IEEE Transactions on Industrial Informatics*, *15*(5), 3077–3084.

Isa, I. S. B. M., El-Gorashi, T. E., Musa, M. O., and Elmirghani, J. M. (2020). "Energy Efficient Fog-Based Healthcare Monitoring Infrastructure". *IEEE Access*, *8*, 197828–197852.

Ittichaichareon, C., Suksri, S., and Yingthawornsuk, T. (2012). "Speech recognition using MFCC". In *International Conference on Computer Graphics, Simulation and Modeling (ICGSM'2012)*, 28–29.

Jalali, F., Hinton, K., Ayre, R., Alpcan, T., and Tucker, R. S. (2016). "Fog computing may help to save energy in cloud computing". *IEEE Journal on Selected Areas in Communications*, *34*(5), 1728–1739.

Jia, B., Hu, H., Zeng, Y., Xu, T., and Yang, Y. (2018). "Double-matching resource allocation strategy in fog computing networks based on cost efficiency". *Journal of Communications and Networks*, *20*(3), 237–246.

Kang, H. S., Lee, J. Y., Choi, S., Kim, H., Park, J. H., Son, J. Y., Kim, B. H., and Do Noh, S. (2016). "Smart manufacturing: Past research, present findings, and future directions". *International journal of precision engineering and manufacturing-green technology*, *3*(1), 111–128.

Kavis, M. J. (2014). "Architecting the cloud: design decisions for cloud computing service models (SaaS, PaaS, and IaaS)". John Wiley and Sons.

Kherraf, N., Alameddine, H. A., Sharafeddine, S., Assi, C., and Ghrayeb, A. (2019). "Optimized Provisioning of Edge Computing Resources with Heterogeneous Workload in IoT Networks". *IEEE Transactions on Network and Service Management*, 1–1.

Khochare, A., Ravindra, P., Reddy, S. P., and Simmhan, Y. (2017). "Distributed Video Analytics Across Edge and Cloud Using ECHO". In *International Conference on Service-Oriented Computing*, 402–407. Springer.

Kim, H. and Ben-Othman, J. (2018). "A collision-free surveillance system using smart UAVs in multi domain IoT". *IEEE communications letters*, *22*(12), 2587–2590.

Kim, H.-S. (2016). "Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are". *International Journal of Cisco*.

Koolagudi, S. G., Vishwanath, B. K., Akshatha, M., and Murthy, Y. V. (2017). "Performance Analysis of LPC and MFCC Features in Voice Conversion Using Artificial Neural Networks". In *Proceedings of the International Conference on Data Engineering and Communication Technology*, 275–280. Springer.

Krishna, K. L., Silver, O., Malende, W. F., and Anuradha, K. (2017). "Internet of Things application for implementation of smart agriculture system". In *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, 54–59. IEEE.

Kumari, A., Tanwar, S., Tyagi, S., and Kumar, N. (2018). "Fog computing for Healthcare 4.0 environment: Opportunities and challenges". *Computers and Electrical Engineering*, *72*, 1–13.

Lee, W., Nam, K., Roh, H.-G., and Kim, S.-H. (2016). "A gateway based fog computing architecture for wireless sensors and actuator networks". In *2016 18th International Conference on Advanced Communication Technology (ICACT)*, 210–213. IEEE.

Li, H., Shou, G., Hu, Y., and Guo, Z. (2016). "Mobile edge computing: Progress and challenges". In *2016 4th IEEE international conference on mobile cloud computing, services, and engineering (MobileCloud)*, 83–84. IEEE.

Li, L., Ota, K., and Dong, M. (2018). "Deep learning for smart industry: Efficient manufacture inspection system with fog computing". *IEEE Transactions on Industrial Informatics*, *14*(10), 4665–4673.

Liang, Y., Li, W., Lu, X., and Wang, S. (2019). "Fog computing and convolutional neural network enabled prognosis for machining process optimization". *Journal of Manufacturing Systems*, *52*, 32–42.

Lin, K., Pankaj, S., and Wang, D. (2018). "Task offloading and resource allocation for edge-of-things computing on smart healthcare systems". *Computers and Electrical Engineering*, *72*, 348–360.

Liu, B., Xu, X., Qi, L., Ni, Q., and Dou, W. (2020). "Task scheduling with precedence and placement constraints for resource utilization improvement in multi-user MEC environment". *Journal of Systems Architecture*, 101970.

Liu, C., Xiang, F., Wang, P., and Sun, Z. (2019). "A Review of Issues and Challenges in Fog Computing Environment". In *2019 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, 232–237. IEEE.

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, P., Qi, B., and Banerjee, S. (2018). "Edgeeye: An edge service framework for real-time intelligent video analytics". In *Proceedings of the 1st International Workshop on Edge Systems, Analytics and Networking*, 1–6.

Liu, W., Kong, C., Niu, Q., Jiang, J., and Zhou, X. (2020). "A method of NC machine tools intelligent monitoring system in smart factories". *Robotics and Computer-Integrated Manufacturing*, *61*, 101842.

Luo, J., Yin, L., Hu, J., Wang, C., Liu, X., Fan, X., and Luo, H. (2019). "Container-based fog computing architecture and energy-balancing scheduling algorithm for energy IoT". *Future Generation Computer Systems*, *97*, 50–60.

Luo, S., Chen, X., Zhou, Z., Chen, X., and Wu, W. (2020). "Incentive-Aware Micro Computing Cluster Formation for Cooperative Fog Computing". *IEEE Transactions on Wireless Communications*, *19*(4), 2643–2657.

Mabrouk, A. B. and Zagrouba, E. (2018). "Abnormal behavior recognition for intelligent video surveillance systems: A review". *Expert Systems with Applications*, *91*, 480–491.

Mahmoud, M. M., Rodrigues, J. J., Saleem, K., Al-Muhtadi, J., Kumar, N., and Korotaev, V. (2018). "Towards energy-aware fog-enabled cloud of things for healthcare". *Computers and Electrical Engineering*, *67*, 58–69.

Mahmud, R. and Buyya, R. (2019). "Modelling and simulation of fog and edge computing environments using iFogSim toolkit". *Fog and edge computing: Principles and paradigms*, 1–35.

Mahmud, R., Kotagiri, R., and Buyya, R. (2018). "Fog computing: A taxonomy, survey and future directions". In *Internet of everything* 103–130. Springer.

Mahmud, R., Ramamohanarao, K., and Buyya, R. (2018). "Latency-aware application module management for fog computing environments". *ACM Transactions on Internet Technology (TOIT)*, *19*(1), 1–21.

Mahmud, R., Toosi, A. N., Rao, K., and Buyya, R. (2019). "Context-aware placement of industry 4.0 applications in fog computing environments". *IEEE Transactions on Industrial Informatics*.

Masip-Bruin, X., Marín-Tordera, E., Juan-Ferrer, A., Queralt, A., Jukan, A., Garcia, J., Lezzi, D., Jensen, J., Cordeiro, C., Leckey, A., et al. (2018). "mF2C: towards a coordinated management of the IoT-fog-cloud continuum". In *Proceedings of the 4th ACM MobiHoc Workshop on Experiences with the Design and Implementation of Smart Objects*, 1–8.

Merlino, G., Bruneo, D., Distefano, S., Longo, F., and Puliafito, A. (2014). "Stack4Things: integrating IoT with OpenStack in a Smart City context". In *2014 International Conference on Smart Computing Workshops*, 21–28. IEEE.

Mishra, S. K., Puthal, D., Rodrigues, J. J., Sahoo, B., and Dutkiewicz, E. (2018). "Sustainable service allocation using a metaheuristic technique in a fog server for industrial applications". *IEEE Transactions on Industrial Informatics*, *14*(10), 4497–4506.

Mital, M., Pani, A. K., Damodaran, S., and Ramesh, R. (2015). "Cloud based management and control system for smart communities: A practical case study". *Computers in Industry*, *74*, 162–172.

Murtaza, F., Akhunzada, A., ul Islam, S., Boudjadar, J., and Buyya, R. (2020). "QoS-aware service provisioning in fog computing". *Journal of Network and Computer Applications*, 102674.

Naas, M. I., Parvedy, P. R., Boukhobza, J., and Lemarchand, L. (2017). "iFogStor: an IoT data placement strategy for fog infrastructure". In *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*, 97–104. IEEE.

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)*, 24–25. IEEE.

Natesha, B. and Guddeti, R. M. R. (2021a). "Adopting elitism-based Genetic Algorithm for minimizing multi-objective problems of IoT service placement in fog computing environment". *Journal of Network and Computer Applications*, *178*, 102972.

Natesha, B. and Guddeti, R. M. R. (2021b). "Fog-based intelligent machine malfunction monitoring system for industry 4.0". *IEEE Transactions on Industrial Informatics*, *17*(12), 7923–7932.

Natesha, B. and Guddeti, R. M. R. (2022). "Meta-heuristic Based Hybrid Service Placement Strategies for Two-Level Fog Computing Architecture". *Journal of Network and Systems Management*, *30*(3), 1–23.

Neto, A. J., Zhao, Z., Rodrigues, J. J., Camboim, H. B., and Braun, T. (2018). "Fog-based crime-assistance in smart iot transportation system". *IEEE access*, *6*, 11101–11111.

Nguyen, N. D., Phan, L.-A., Park, D.-H., Kim, S., and Kim, T. (2020). "ElasticFog: Elastic Resource Provisioning in Container-Based Fog Computing". *IEEE Access*, *8*, 183879–183890.

Okay, F. Y. and Ozdemir, S. (2016). "A fog computing based smart grid model". In *2016 international symposium on networks, computers and communications (ISNCC)*, 1–6. IEEE.

Ouerhani, N., Pazos, N., Aeberli, M., and Muller, M. (2016). "IoT-based dynamic street light control for smart cities use cases". In *2016 International Symposium on Networks, Computers and Communications (ISNCC)*, 1–5. IEEE.

Park, S. J., Subramaniyam, M., Kim, S. E., Hong, S., Lee, J. H., Jo, C. M., and Seo, Y. (2017). "Development of the elderly healthcare monitoring system with IoT". In *Advances in Human Factors and Ergonomics in Healthcare* 309–315. Springer.

Patman, J., Alfarhood, M., Islam, S., Lemus, M., Calyam, P., and Palaniappan, K. (2018). "Predictive analytics for fog computing using machine learning and GENI". In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 790–795. IEEE.

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.

Peralta, G., Garrido, P., Bilbao, J., Agüero, R., and Crespo, P. M. (2020). "Fog to cloud and network coded based architecture: Minimizing data download time for smart mobility". *Simulation Modelling Practice and Theory*, *101*, 102034.

Pham, T. N., Tsai, M.-F., Nguyen, D. B., Dow, C.-R., and Deng, D.-J. (2015). "A cloud-based smart-parking system based on Internet-of-Things technologies". *IEEE Access*, *3*, 1581–1591.

Powell, C., Desiniotis, C., and Dezfouli, B. (2020). "The Fog Development Kit: A Platform for the Development and Management of Fog Systems". *IEEE Internet of Things Journal*, *7*(4), 3198–3213.

Premsankar, G., Di Francesco, M., and Taleb, T. (2018). "Edge computing for the Internet of Things: A case study". *IEEE Internet of Things Journal*, *5*(2), 1275–1284.

Puliafito, C., Mingozzi, E., Longo, F., Puliafito, A., and Rana, O. (2019). "Fog computing for the internet of things: A Survey". *ACM Transactions on Internet Technology (TOIT)*, *19*(2), 1–41.

Purohit, H., Tanabe, R., Ichige, K., Endo, T., Nikaido, Y., Suefusa, K., and Kawaguchi, Y. (2019a). "MIMII Dataset: Sound Dataset for Malfunctioning Industrial Machine Investigation and Inspection". *arXiv preprint arXiv:1909.09347*.

Purohit, H., Tanabe, R., Ichige, K., Endo, T., Nikaido, Y., Suefusa, K., and Kawaguchi, Y. (2019b), "MIMII Dataset: Sound Dataset for Malfunctioning Industrial Machine Investigation and Inspection".

Rahimi, M., Songhorabadi, M., and Kashani, M. H. (2020). "Fog-based smart homes: A systematic review". *Journal of Network and Computer Applications*, *153*, 102531.

Rakshith, G., Rahul, M., Sanjay, G., Natesha, B., and Reddy, G. R. M. (2018). "Resource provisioning framework for IoT applications in fog computing environment". In *2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, 1–6. IEEE.

Ramírez, W., Masip-Bruin, X., Marin-Tordera, E., Souza, V. B. C., Jukan, A., Ren, G.-J., and de Dios, O. G. (2017). "Evaluating the benefits of combined and continuous Fog-to-Cloud architectures". *Computer Communications*, *113*, 43–52.

Rashmi, M., Ashwin, T., and Guddeti, R. M. R. (2020). "Surveillance video analysis for student action recognition and localization inside computer laboratories of a smart campus". *Multimedia Tools and Applications*, 1–23.

Ravindra, P., Khochare, A., Reddy, S. P., Sharma, S., Varshney, P., and Simmhan, Y. (2017). "ECHO: An Adaptive Orchestration Platform for Hybrid Dataflows across Cloud and Edge". In *International Conference on Service-Oriented Computing*, 395–410. Springer.

Ray, P. P. (2017). "Internet of things for smart agriculture: Technologies, practices and future direction". *Journal of Ambient Intelligence and Smart Environments*, *9*(4), 395–420.

Resources, A. C. (2022), "Alibaba Cloud Elastic Compute Service". https://www.alibabacloud.com/product/ecs?spm=a3c0i.7938564.8215766810.1.533f441ev6pNxa/. [Online; last accessed 30-Mar-2022].

Rezazadeh, Z., Rahbari, D., and Nickray, M. (2018). "Optimized module placement in IoT applications based on fog computing". In *Electrical Engineering (ICEE), Iranian Conference on*, 1553–1558. IEEE.

Rimal, B. P., Choi, E., and Lumb, I. (2009). "A taxonomy and survey of cloud computing systems". In *2009 Fifth International Joint Conference on INC, IMS and IDC*, 44–51. Ieee.

Ruan, L., Guo, S., Qiu, X., and Buyya, R. (2020). "Fog Computing for Smart Grids: Challenges and Solutions". *arXiv preprint arXiv:2006.00812*.

Ruan, L., Liu, Z., Qiu, X., Wang, Z., Guo, S., and Qi, F. (2018). "Resource allocation and distributed uplink offloading mechanism in fog environment". *Journal of Communications and Networks*, *20*(3), 247–256.

Sahidullah, M. and Saha, G. (2012). "Design, analysis and experimental evaluation of block based transformation in MFCC computation for speaker recognition". *Speech communication*, *54*(4), 543–565.

Salaht, F. A., Desprez, F., Lebre, A., Prud'Homme, C., and Abderrahim, M. (2019). "Service placement in fog computing using constraint programming". In *2019 IEEE International Conference on Services Computing (SCC)*, 19–27. IEEE.

Salman, O., Elhajj, I., Chehab, A., and Kayssi, A. (2018). "IoT survey: An SDN and fog computing perspective". *Computer Networks*, *143*, 221–246.

Sanjaya, W. M., Anggraeni, D., and Santika, I. P. (2018). "Speech Recognition using Linear Predictive Coding (LPC) and Adaptive Neuro-Fuzzy (ANFIS) to Control 5 DoF Arm Robot". In *Journal of Physics: Conference Series*, volume 1090, 012046. IOP Publishing.

Saraubon, K., Anurugsa, K., and Kongsakpaibul, A. (2018). "A smart system for elderly care using iot and mobile technologies". In *Proceedings of the 2018 2nd International Conference on Software and e-Business*, 59–63.

Satija, U., Ramkumar, B., and Manikandan, M. S. (2017). "Real-time signal quality-aware ECG telemetry system for IoT-based health care monitoring". *IEEE Internet of Things Journal*, *4*(3), 815–823.

Satyanarayanan, M. (2017). "The emergence of edge computing". *Computer*, *50*(1), 30–39.

Saucedo-Dorantes, J. J., Delgado-Prieto, M., Osornio-Rios, R. A., and de Jesus Romero-Troncoso, R. (2020). "Industrial Data-driven Monitoring based on Incremental Learning applied to the Detection of Novel Faults". *IEEE Transactions on Industrial Informatics*.

Sharma, P., Chaufournier, L., Shenoy, P., and Tay, Y. (2016). "Containers and virtual machines at scale: A comparative study". In *Proceedings of the 17th International Middleware Conference*, 1–13.

Shekhar, S., Chhokra, A., Sun, H., Gokhale, A., Dubey, A., and Koutsoukos, X. (2019). "URMILA: A performance and mobility-aware fog/edge resource management middleware". In *2019 IEEE 22nd International Symposium on Real-Time Distributed Computing (ISORC)*, 118–125. IEEE.

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.

Shi, W. and Dustdar, S. (2016). "The promise of edge computing". *Computer*, *49*(5), 78–81.

Simmhan, Y. (2017a). "Big data and fog computing". *arXiv preprint arXiv:1712.09552*.

Simmhan, Y. (2017b). "IoT analytics across edge and cloud platforms". *IEEE IoT Newsletter*.

Simmhan, Y., Ravindra, P., Chaturvedi, S., Hegde, M., and Ballamajalu, R. (2018). "Towards a data-driven IoT software architecture for smart city utilities". *Software: Practice and Experience*, *48*(7), 1390–1416.

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.

Solutions, C. F. C. (2015). "Unleash the power of the Internet of Things". *Cisco Systems Inc*.

Souza, V. B., Masip-Bruin, X., Marín-Tordera, E., Sànchez-López, S., Garcia, J., Ren, G.-J., Jukan, A., and Ferrer, A. J. (2018). "Towards a proper service placement in combined Fog-to-Cloud (F2C) architectures". *Future Generation Computer Systems*, *87*, 1–15.

Systems, C. (2015). "Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are".

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

Tang, B., Chen, Z., Hefferman, G., Pei, S., Wei, T., He, H., and Yang, Q. (2017). "Incorporating intelligence in fog computing for big data analysis in smart cities". *IEEE Transactions on Industrial informatics*, *13*(5), 2140–2150.

Tang, Z., Zhou, X., Zhang, F., Jia, W., and Zhao, W. (2018). "Migration modeling and learning algorithms for containers in fog computing". *IEEE Transactions on Services Computing*, *12*(5), 712–725.

Tange, K., De Donno, M., Fafoutis, X., and Dragoni, N. (2020). "A systematic survey of industrial internet of things security: Requirements and fog computing opportunities". *IEEE Communications Surveys and Tutorials*, *22*(4), 2489–2520.

Tao, F. and Qi, Q. (2017). "New IT driven service-oriented smart manufacturing: framework and characteristics". *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *49*(1), 81–91.

Teerapittayanon, S., McDanel, B., and Kung, H.-T. (2017). "Distributed deep neural networks over the cloud, the edge and end devices". In *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 328–339. IEEE.

Tsakanikas, V. and Dagiuklas, T. (2018). "Video surveillance systems-current status and future trends". *Computers and Electrical Engineering*, *70*, 736–753.

Tuli, S., Basumatary, N., and Buyya, R. (2019). "Edgelens: Deep learning based object detection in integrated iot, fog and cloud computing environments". In *2019 4th International Conference on Information Systems and Computer Networks (ISCON)*, 496–502. IEEE.

Tuli, S., Basumatary, N., Gill, S. S., Kahani, M., Arya, R. C., Wander, G. S., and Buyya, R. (2020). "Healthfog: An ensemble deep learning based smart healthcare system for automatic diagnosis of heart diseases in integrated iot and fog computing environments". *Future Generation Computer Systems*, *104*, 187–200.

Tuli, S., Mahmud, R., Tuli, S., and Buyya, R. (2019). "Fogbus: A blockchain-based lightweight framework for edge and fog computing". *Journal of Systems and Software*, *154*, 22–36.

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

Varshney, P. and Simmhan, Y. (2020). "Characterizing application scheduling on edge, fog, and cloud computing resources". *Software: Practice and Experience*, *50*(5), 558–595.

Vilalta, R., Vía, S., Mira, F., Casellas, R., Muñoz, R., Alonso-Zarate, J., Kousaridas, A., and Dillinger, M. (2018). "Control and management of a connected car using sdn/nfv, fog computing and yang data models". In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, 378–383. IEEE.

Voorsluys, W., Broberg, J., Buyya, R., et al. (2011). "Introduction to cloud computing". *Cloud computing: Principles and paradigms*, 1–44.

Wang, C., Dong, S., Zhao, X., Papanastasiou, G., Zhang, H., and Yang, G. (2019). "Saliencygan: Deep learning semisupervised salient object detection in the fog of iot". *IEEE Transactions on Industrial Informatics*, *16*(4), 2667–2676.

Wang, J., Zheng, P., Lv, Y., Bao, J., and Zhang, J. (2019). "Fog-IBDIS: Industrial big data integration and sharing with fog computing for manufacturing systems". *Engineering*, *5*(4), 662–670.

Wang, Y., Uehara, T., and Sasaki, R. (2015). "Fog computing: Issues and challenges in security and forensics". In *2015 IEEE 39th Annual Computer Software and Applications Conference*, volume 3, 53–59. IEEE.

Xavier, T. C., Santos, I., Delicato, F. C., Pires, P. F., Alves, M. P., Calmon, T. S., Oliveira, A. C., and Amorim, C. L. (2020). "Collaborative resource allocation for Cloud of Things systems". *Journal of Network and Computer Applications*, 102592.

Xing, Y. and Zhan, Y. (2012). "Virtualization and cloud computing". In *Future Wireless Networks and Information Systems* 305–312. Springer.

Yadav, V., Natesha, B., and Guddeti, R. M. R. (2019). "GA-PSO: Service Allocation in Fog Computing Environment Using Hybrid Bio-Inspired Algorithm". In *TENCON 2019-2019 IEEE Region 10 Conference (TENCON)*, 1280–1285. IEEE.

Yang, H., Kumara, S., Bukkapatnam, S. T., and Tsung, F. (2019). "The internet of things for smart manufacturing: A review". *IISE Transactions*, *51*(11), 1190–1216.

Yang, J., Wen, J., Wang, Y., Jiang, B., Wang, H., and Song, H. (2019). "Fog-Based Marine Environmental Information Monitoring Toward Ocean of Things". *IEEE Internet of Things Journal*, *7*(5), 4238–4247.

Yang, Y., Wang, K., Zhang, G., Chen, X., Luo, X., and Zhou, M.-T. (2018). "MEETS: Maximal energy efficient task scheduling in homogeneous fog networks". *IEEE Internet of Things Journal*, *5*(5), 4076–4087.

Yang, Y., Zhao, S., Zhang, W., Chen, Y., Luo, X., and Wang, J. (2018). "DEBTS: Delay energy balanced task scheduling in homogeneous fog networks". *IEEE Internet of Things Journal*, *5*(3), 2094–2106.

Yassine, A., Singh, S., Hossain, M. S., and Muhammad, G. (2019). "IoT big data analytics for smart homes with fog and cloud computing". *Future Generation Computer Systems*, *91*, 563–573.

Yi, S., Hao, Z., Qin, Z., and Li, Q. (2015). "Fog computing: Platform and applications". In *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, 73–78. IEEE.

Yi, S., Li, C., and Li, Q. (2015). "A survey of fog computing: concepts, applications and issues". In *Proceedings of the 2015 workshop on mobile big data*, 37–42.

Yu, W., Najafi, A., Huang, Y., and Garcia-Ortiz, A. (2020). "Combination of Task Allocation and Approximate Computing for Fog Architecture based IoT". *IEEE Internet of Things Journal*.

Zahoor, S., Javaid, N., Khan, A., Ruqia, B., Muhammad, F. J., and Zahid, M. (2018). "A cloud-fog-based smart grid model for efficient resource utilization". In *2018 14th International Wireless Communications and Mobile Computing Conference (IWCMC)*, 1154–1160. IEEE.

Zanella, A., Bui, N., Castellani, A., Vangelista, L., and Zorzi, M. (2014). "Internet of things for smart cities". *IEEE Internet of Things journal*, *1*(1), 22–32.

Zeng, X., Fang, B., Shen, H., and Zhang, M. (2020). "Distream: scaling live video analytics with workload-adaptive distributed edge intelligence". In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, 409–421.

Zhang, B., Wang, X., and Huang, M. (2018). "Multi-objective optimization controller placement problem in internet-oriented software defined network". *Computer Communications*, *123*, 24–35.

Zhang, W., Zhang, Z., and Chao, H.-C. (2017). "Cooperative fog computing for dealing with big data in the internet of vehicles: Architecture and hierarchical resource management". *IEEE Communications Magazine*, *55*(12), 60–67.

Publications

Journal Papers

1. **Natesha B V** and Ram Mohana Reddy Guddeti, "Adopting Elitism-based Genetic Algorithm for Minimizing Multi-objective Problems of IoT Service Placement in Fog Computing Environment", **Journal of Network and Computer Applications**, Volume = "178", DOI= "https://doi.org/10.1016/j.jnca.2020.102972", (SCIE and Scopus indexed), Publication Date: 2021/3/15.

2. **Natesha B V** and Ram Mohana Reddy Guddeti, "Fog-based Intelligent Machine Malfunction Monitoring System for Industry 4.0", **IEEE Transactions on Industrial Informatics**, Volume=17, Issue=12, Pages=7923-7932, DOI= 10.1109/TII.2021.3056076, (SCIE and Scopus indexed), Publication Date: 2021/2/2.

3. **Natesha B V** and Ram Mohana Reddy Guddeti, "Meta-Heuristic based Hybrid Service Placement Strategies for Two-level Fog Computing Architecture", **"Journal of Network and Systems Management"**, DOI:10.1007/s10922-022-09660-w, Volume=30, Number=3, Pages=47, (SCIE and Scopus Indexed), Publication Date: 2022/4/22.

4. **Natesha B V** and Ram Mohana Reddy Guddeti, "Cost Efficient Deadline-Aware IoT Service Placement in Fog-Cloud Computing Environment", **"Journal of Supercomputing"**, (Under Review).

Conference Papers

1. **Natesha B V** and Ram Mohana Reddy Guddeti, (2018) "Heuristic-based IoT Application Modules Placement in the Fog-Cloud Computing Environment", In Proc. of the 11^{th} IEEE/ACM International Conference on **Utility and Cloud Computing (UCC 2018)**, Pages 24-25,
DOI: 10.1109/UCC-Companion.2018.00027, Publication Date: 2018/12/17.

2. Rakshith G, Rahul M V, Sanjay G S, **Natesha B V** and Ram Mohana Reddy G (2018), "Resource Provisioning Framework for IoT Applications in Fog Computing Environment", In Proc. of the 12^{th} IEEE International Conference on **Advanced Networks and Telecommunication Systems (ANTS 2018)**, pages 1-6, DOI: 10.1109/ANTS.2018.8710172, Publication Date: 2018/12/16

3. **Natesha B V** and Ram Mohana Reddy Guddeti (2020), "Fog-based Video Surveillance System for Smart City Applications", In Proc. of the 8^{th} International Conference on **Frontiers of Intelligent Computing: Theory and Applications. (FICTA 2020)**, Pages 747-754, DOI: 10.1007/978-981-15-5788-0_70, Publication Year: 2021.

Other Publications

1. Md. Shahzad Alam, **Natesha B V**, Ashwin T.S, and G. Ram Mohana Reddy (2019), "UAV based Cost-Effective Real-Time Abnormal Event Detection using Edge Computing", **Springer Multimedia Tools and Applications,** DOI: 10.1007/s11042-019-08067-1, Publication Date: 2019/12.

2. **Natesha B V**, Neeraj Sharma, Shridhar Domanal, and Ram Mohana Reddy Guddeti (2018), "GWOTS: Grey Wolf Optimization based Task Scheduling at the Green Cloud Data Center", In Proc. of the 14^{th} International Conference on **Semantics, Knowledge and Grids (SKG 2018)**, pp. 181-187,
DOI: 10.1109/SKG.2018.00034, Publication Date: 2018/9/12.

3. Vinita Yadav, **Natesha B V** and Ram Mohana Reddy G (2019), "GA-PSO: Service Allocation in Fog Computing Environment Using Hybrid Bio-Inspired Algorithm" (2019), In Proc. of the **IEEE Region 10 Conference TENCON 2019**, pages 1280-1285, DOI: 10.1109/TENCON.2019.8929234, Publication Date:2019/10/17.

Curriculum Vitae

Mr. Natesha B V
Full-Time Research Scholar
Department of Information Technology
National Institute of Technology Karnataka
P.O. Srinivasanagar, Surathkal
Mangalore-575 025

Permanent Address

Natesha B V
Banavase, Shantigrama (Hobli)
Mosale Hosahalli (Post), Hassan (District)
Karnataka (State), PIN-573212
Email: nateshbv18@gmail.com
Mobile: +91-9901427324.

Academic Records

1. M.Tech. in Computer Network Engineering, The National Institute of Engineering (NIE) Mysore, 2014 (CGPA 9.62).

2. B.E. in Computer Science and Engineering, HMS Institute of Technology (HMSIT), Tumkur, 2011 (First Class 68.12%).

Research Interests

Cloud Computing
Fog Computing
Internet of Things
Machine Learning

Programming Languages

C, Java, Python.