Edge Hill University

Role of Middleware in Improving Reliability of Fog Applications

by

Hezekiah Yeng Samwini

A thesis submitted in partial fulfilment for the degree of Doctor of Philosophy

in the

Department of Computer Science

Supervisory Team:

Prof Ella Pereira

Dr. Mohsin Raza

Dr. Umar Khan

June 2025

Declaration

The work presented in this thesis was carried out at the Department of Computer Science, Edge Hill University. Unless otherwise stated, it is the original work of the author.

While registered as a candidate for the degree of Doctor of Philosophy, for which submission is now made, the author has not been registered as a candidate for any other award. This thesis has not been submitted in whole, or in part, for any other degree.

HEZEKIAH YENG SAMWINI

Department of Computer Science
Edge Hill University
St Helens Road
Ormskirk
L39 4QP
UK

June 2025

Declaration of Authorship

I, HEZEKIAH YENG SAMWINI, declare that this thesis titled 'THE ROLE OF MIDDLEWARE IN IMPROVING RELIABILITY OF FOG APPLICATIONS' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. Except for such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ____

Date: ____

"A distributed system is one in which the failure of a computer you didnt even know existed can render your own computer unusable."

Leslie Lamport

Edge Hill University

Abstract

Department of Computer Science

Doctor of Philosophy

by Hezekiah Yeng Samwini

Fog computing extends cloud services to the network edge to address latency and reliability challenges in modern applications. However, resource management in fog environments remains a challenge due to heterogeneity and the dynamic nature of fog nodes. While middleware has traditionally addressed similar challenges in distributed computing, its application to fog computing resource management has not been thoroughly explored. Therefore, this thesis proposes a middleware approach to resource management in fog computing.

A novel adaptive middleware architecture employing the MAPE-K(Monitor, Analyse, Plan, Execute over shared Knowledge) self-adaptation framework to dynamically switch between clustered and peer-to-peer configurations based on conditions in the environment. The middleware introduces: (1) a request handling algorithm that implements multi-tier resource discovery across fog nodes, clusters, and cloud layers with $O(1)$ to $O(S)$ complexity depending on service lookup implementation; (2) an Autonomous System-inspired clustering mechanism with dedicated inter-cluster communication interfaces for cross-domain resource sharing; and (3) a decentralised P2P bootstrapping algorithm for optimal peer selection based on propagation delay and computational capacity.

The proposed solutions are evaluated through simulations using iFogSim and Omnet++. Simulation experiments include processing over 3,500 IoT requests following an exponential distribution with a mean inter-arrival time, $\lambda = 100ms$ and 3-20 fog nodes to evaluate the performance of centralised and decentralised architectures. The simulation results demonstrate improvements in response time, network utilisation, and energy efficiency compared to traditional cloud-centric and static/non-collaborative fog architectures. The results highlight the potential of middleware in improving adaptability and reliability of fog computing.

This work's key contributions are: (1) First middleware framework enabling both clustered and P2P fog architectures with adaptive switching; (2) Novel inter-cluster resource sharing mechanisms addressing the gap in fog clustering research; (3) Comprehensive resource management framework integrating task offloading, load balancing, and service discovery.

Acknowledgements

This work would not have been possible without the support of many individuals whose help I deeply appreciate. I extend my sincere gratitude to everyone who has assisted me in various ways throughout this research.

It has been a great pleasure working with the staff and students at Edge Hill University during my time as a doctoral student. Producing this thesis would not have been possible without the mentoring and support of Ella Pereira, my director of studies. Her patience, kindness, and guidance have been invaluable throughout this journey. I owe an outstanding debt of gratitude to Mohsin Raza, my supervisor, whose invaluable feedback and insights taught me the importance of taking the time to produce meaningful work and shaped this work. I also thank Muhammad Awais for his support and mentorship during his time at Edge Hill. Thanks to Umar Khan for his guidance especially in the final stages of this work. This work would not have been possible without their feedback and encouragement.

I am also grateful to all the staff (teaching and non-teaching) at the Department of Computer Science for the assistance they provided throughout this research. Thank you to Amr, Sally, Kaja, Mark and all the admin staff for all the work you did for me related to conferences and other matters. I want to extend my sincere thanks to Edge Hill University for granting me the opportunity to undertake this studentship as a PhD student and Graduate Teaching Assistant.

To all those I shared *Room-F09* with at different stages, especially Babatunde, Paul, Aditya, and Reena, who have been with me the longest—thank you. I benefitted greatly from our interactions. Thanks also to all the friends I made during my time in Ormskirk, Mike Somervell and the wonderful people at Cottage Lane Mission Church.

I am profoundly grateful to my close friends in the Brummie Navigators—Kim and Insook, Joseph and Johanna, Dave, Sterling, and many others. Thank you so much for your prayers, support, and encouraging words.

I am deeply grateful to my family and friends for their continuous support and encouragement. I could not have achieved this without their patience, kindness, and encouragement. I am especially thankful to my parents, Nathan and Gloria, to whom I owe so much. Thanks also to my siblings—Esther, Cephas and Sarah, Nehemiah, and Abigail—for their encouragement. My heartfelt gratitude goes to Edwin, Emmanuel, and Phil Dordoe.

To all who have helped me on this journey, I don't have the space to list all your names, but I am truly grateful. Thank you.

Contents

List of Figures

List of Tables

Abbreviations

AMQP	**A**dvanced **M**essage **Q**ueuing **P**rotocol
AWS	**A**mazon **W**eb **S**ervices
CBOR	**C**oncise **B**inary **O**bject **R**epresentation
CC	**C**loud **C**omputing
CoAP	**C**onstrained **A**ccess **P**rotocol
CPS	**C**yber-**P**hysical **S**ystems
CSP	**C**loud **S**ervice **P**rovider
DTLS	**D**atagram **T**ransport **L**ayer **S**ecurity
EEG	**E**lectro**e**ncephalo**g**raphy
HTTPS	**H**yper**t**ext **T**ransfer **P**rotocol **S**ecure
IaaS	**I**nfrastructure **a**s **a** **S**ervice
IEEE	**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers
IoT	**I**nternet **o**f **T**hings
IoV	**I**nternet **o**f **V**ehicles
ITS	**I**ntelligent **T**ransport **S**ystem
JSON	**J**ava**S**cript **O**bject **N**otation
LTE-M	**L**ong **T**erm **E**volution for **M**achines
LwM2M	**L**ightweight **M**achine to **M**achine
MAC	**M**edium **A**ccess **C**ontrol
MEC	**M**obile **E**dge **C**omputing / **M**ulti-**A**ccess **E**dge **C**omputing
MQTT	**M**essage **Q**ueuing **T**elemetry **T**ransport
NB-IoT	**N**arrow**b**and **I**nternet of **T**hings

NIST	**N**ational **I**nstitute of **S**tandards and **T**echnology
OFC	**O**pen **F**og **C**onsortium
OSI	**O**pen **S**ystems **I**nterconnection
PaaS	**P**latform **as a** **S**ervice
QoE	**Q**uality **of** **E**xperience
QoS	**Q**uality **of** **S**ervice
RFID	**R**adio **F**requency **I**dentification
SaaS	**S**oftware **as a** **S**ervice
SoA	**S**ervice-oriented **A**rchitecture
TCP	**T**ransmission **C**ontrol **P**rotocol
TLS/SSL	**T**ransport **L**ayer **S**ecurity / **S**ecure **S**ockets **L**ayer
UDP	**U**ser **D**atagram **P**rotocol
URLLC	**U**ltra-**R**eliable **L**ow-**L**atency **C**ommunications
UUID	**U**niversal **U**nique **I**dentifier
VANET	**V**ehicular **A**d-hoc **N**etworks
Wi-Fi	**W**ireless **F**idelity
WSN	**W**ireless **S**ensor **N**etworks
XML	**E**xtensible **M**arkup **L**anguage
XMPP	**E**xtensible **M**essaging and **P**resence **P**rotocol

List of Publications

The following publications have arisen from the research presented in this thesis:

1. **H. Samwini**, M. Awais, and E. Pereira, "Middleware for Resource Sharing in Fog Computing with IoT Applications," in *IEEE EUROCON 2023 - 20th International Conference on Smart Technologies*, May 2023, pp. 508–513. **(Chapter 5)**
2. **H. Samwini**, M. Raza, E. Pereira, U. Khan, and M. Awais, "Critical analysis of resource sharing and optimization in fog clustering," in *2024 International Conference on Emerging Trends in Smart Technologies (ICETST)*, Karachi, Pakistan: IEEE, October 2024, pp. 1–6. **(Chapter 6)**

To the glory of God . . .

CHAPTER 1

Introduction

In the first decades of the internet, computers connected people to access information stored on servers or for communication. That has changed in the last two decades. More “things” are connected to the internet now than computers. It has been estimated that the Internet of Things (IoT) would be worth between $5.5 trillion and $12.6 trillion in value globally by 2030 [1]. The growing number of connected devices means an exponential growth in data, which requires processing. In 2024, the total amount of data created, captured, copied, and consumed worldwide is projected to reach 149 zettabytes, up from just 2 zettabytes in 2010. [2]. The current centralised cloud architecture cannot deal with large volumes of data as more devices are connecting to the network everyday. Fog computing has the potential to address this challenge. However, fog computing has not yet fully developed as a computing paradigm. One of the key aspects which must be addressed for fog computing is the management of computing and other resources. This thesis presents an investigation into the role of middleware in addressing this challenge.

1.1 Overview

Fog Computing has been given significant attention by several research communities in computing since it was first proposed in 2012 by Cisco [3]. It represents a paradigm that addresses several challenges identified in cloud computing, especially as IoT expands the volume of data expected to be processed by the cloud. However, fog architecture presents several challenges and raises questions that must be addressed within multiple fields for the paradigm to become a reality. The exponential increase in the number of devices connected to the internet has necessitated a reconsideration of Cloud Computing. As the Internet of Things enables more devices which require cloud services to connect to the Internet, it is becoming infeasible to send all data to the cloud for processing. This section presents an overview of the challenges the current cloud architecture faces–partly a result of its success. It shows how the Internet of Things has necessitated a modification of Cloud Computing and briefly discusses the need for Fog Computing.

Cloud Computing made the long-held vision of computing as a utility a reality by making computing resources readily available to users as and when required. Cloud computing saves developers the capital cost of deploying servers and network infrastructure to host their applications. Developers focus on developing applications, while Cloud Service providers offer the resources and expertise to deploy large-scale data centres focused on making computing resources available on a pay-per-use basis [4]. This arrangement makes upward and downward scaling of an application's resource provision more flexible, as resources to a user can be increased and decreased automatically without the direct involvement of human actors or much effort. The cost savings and flexibility the cloud affords have contributed significantly to the growth of the Internet. Also, because of its flexibility, new services are deployed in a relatively short time, and developers and researchers can explore new ideas at minimal cost [5]. Despite this success, cloud computing faces challenges that have engaged the attention of various research communities, most of which are related to the growing number of users.

The rapid growth of the Internet of Things (IoT) has led to an exponential increase in cloud services [6]. IoT connects physical objects to the Internet. Increasing smart devices at the network edge challenges cloud computing architecture. In the current cloud architecture, end users use services by sending data from the edge through the network core to distant data centres owned and managed by cloud service providers (Amazon, Google, Microsoft, IBM, etc.). While this model has worked well, the increasing number of smart or IoT devices at the edge presents new challenges. First, smart devices produce high volumes of data (Big Data) [7], which requires processing to be valuable. The timely processing of data and actuation in response to its output makes IoT devices smart [8]. This notwithstanding, the characteristics of smart devices (small form factors, little or no processing or storage resources and low power) mean they cannot process the data they produce [9, 10]. Unlike traditional cloud applications used by computing devices through users, most IoT applications have end devices with little or no resources to process their data. They are, therefore, more dependent on the cloud than traditional cloud applications. This has implications for the cloud and application performance.

In certain applications, running services on distant data centres impacts the Quality of Experience (QoE) of users due to increased round-trip delays. Data Centres are usually located a long distance away from end users and sometimes across continents. The farther away a data centre is, the longer the wait a user would have while using an application. Figure 1.1 illustrates the relationship between the user location and the data centre location. The figure shows the round-trip times for pinging Amazon Web Services data centres from their London Data Centre. As the chart shows, as data moves farther away from Europe, the delay increases considerably. Such delays mean that for latency-sensitive applications, it is impractical to run services in distant data centres. According to Schulz et al. [9], latency-critical IoT applications such as autonomous driving require a delay of 50ms, while smart grids require up to 20ms delay. Smart Factories have the lowest delay requirement of 250us to 10ms. For such applications, the mean value of 117ms recorded in figure 1.1 would result in undesirable application performances.

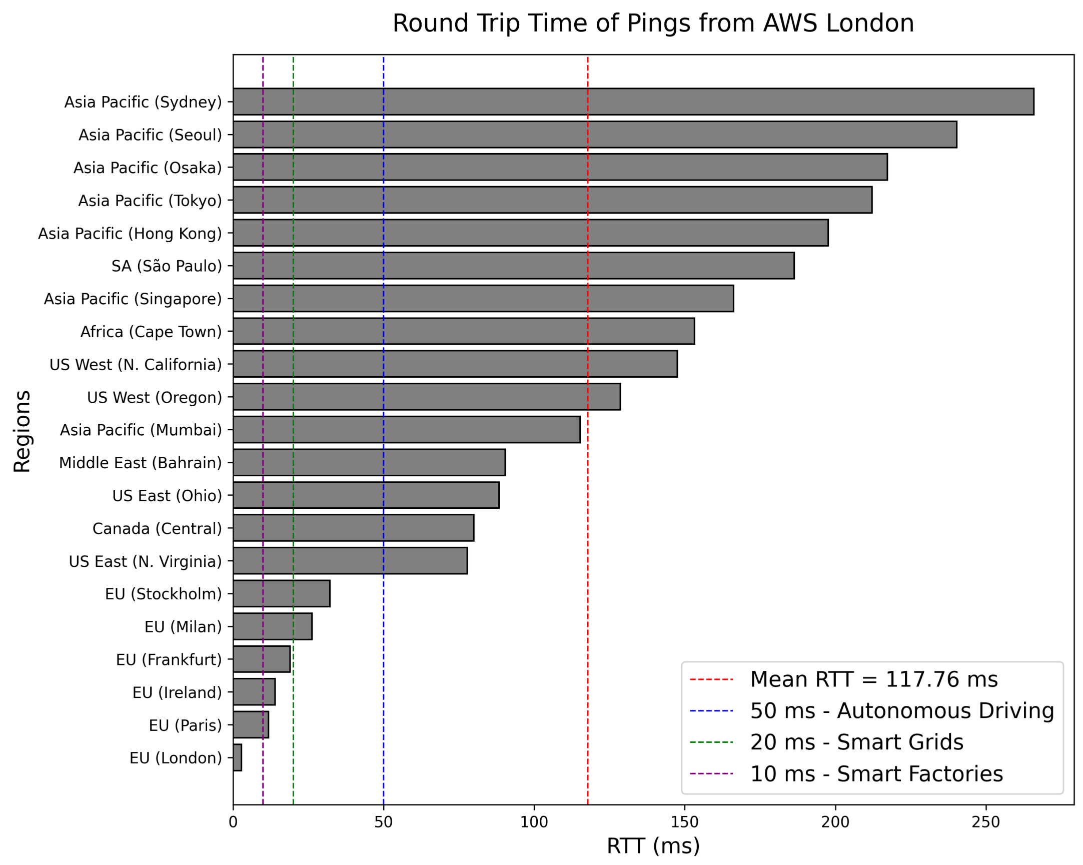


Figure 1.1: Round Trip Time to Amazon Data Centres from London Data Centre and delay requirements of some smart applications.

In the case of delay-critical applications such as autonomous driving, the results could be fatal [11].

Another effect of the exponential increase in IoT devices at the network edge is the sheer increase in network traffic. The growth of IoT devices at the network edge and their requirement for computation resources mean that much more data must traverse the network towards the cloud. The frequency and latency of communication of these devices make it impracticable to send all requests from users and IoT devices to the cloud [12]. In most IoT applications, sensors collect states of physical objects, and actuators send instructions to the objects based on the recorded states. In applications such as Intelligent Transportation Systems, the sensor-processing-actuation loop must remain within the context of the objects or devices involved [13]. Smart vehicles are estimated to generate over 1 terabytes of data per hour[14]. Processing such data at a distant data centre removes it from its context. It is counterproductive for data for an Intelligent Transport System in Liverpool to send all the data to a data centre in Seatle for processing, for example. It would also load the network devices at the network edge if all objects (vehicles, roadside units, traffic lights, radar, etc) sent their data to the same data centre for processing. Furthermore, for specific applications, other factors restrict the use of the cloud. Legal requirements that restrict the storage and processing of personal data outside a jurisdiction mean that processing must be done locally for such applications. For example, in healthcare IoT applications, a lot of sensitive user data is collected, which may restricted from being processed outside the jurisdiction where the data is collected [15].

To solve the above challenges of the IoT-cloud continuum, Cisco proposed fog computing, as an extension of cloud computing, to complement the cloud by processing data closer to its source (at the network edge or core) to improve the user experience for delay-sensitive applications [3]. Fog computing makes use of network devices, servers, and other devices that are close to the user to process user requests. However, the paradigm is yet to become a reality as there are still several challenges for researchers to address. Fog Computing adds a layer to the IoT-cloud architecture. This introduces complexities such as managing the resources at the fog layer and ensuring that users receive the same service guarantees that the predictable cloud environment affords.

1.2 Motivation

Fog computing has been widely acknowledged as a suitable solution to the challenges of cloud computing. As highlighted above, fog computing as a paradigm is not mature. Challenges must be addressed to make it a reality [16, 17]. Given that fog computing adds a layer to the current cloud architecture, it makes the system more complex. Furthermore, the devices at the fog layer are dynamic and heterogeneous in several ways. These aspects of fog computing make resource management and reliability crucial for the development of the paradigm.

Resource management in fog computing requires a holistic approach. Unlike other proposed solutions to providing computation closer to where data is produced, fog computing is envisioned as an extension of the cloud [3]. It, therefore, brings together different resources, including the user and cloud layers. Research into fog resource management has investigated task offloading, application placement, load balancing and resource allocation [18, 19]. Most existing works have focused on specific aspects of resource management and proposed techniques to address them. Factors affecting the reliability of fog systems, such as the dynamic nature of the fog environment, have not been thoroughly investigated [17, 20]. Moreover, reliability aspects, such as availability, have not received much attention. The heterogeneous nature of devices at the fog layer, their effect on resource management and reliability of the fog systems also need to be addressed [21].

Furthermore, although middleware has been utilised in distributed systems to address several challenges associated with resource management in fog computing, there has been limited research on middleware specifically designed for fog computing. In fog computing, devices may be mobile or serve as volunteer nodes made available for a brief, unpredictable period [22]. Moreover, fog nodes and IoT devices are heterogeneous in various ways, such as size, availability, and resources. The need for middleware in fog computing has been identified as an important aspect that has to be addressed [10]. The authors in [23] identified over 500 middleware proposals for IoT. Middleware has also been an important part of cloud computing [24].

Based on the analysis of existing studies, the following aspects of Fog Resource Management remain to be addressed:

- A unified framework that integrates scheduling, resource allocation, load balancing and task offloading at the fog layer.
- Empirical studies quantifying the trade-offs between resource management considerations in fog computing, such as utilisation, reliability and availability.

- Middleware solutions that effectively abstract resource heterogeneity while maintaining performance optimisation.

To address these gaps, the work presented in this thesis explores the use of middleware for resource management in fog computing. A middleware framework is proposed to manage the interaction among fog nodes and between fog nodes and end devices or the cloud. The objective is to ensure reliability while maintaining heterogeneity. An analysis of the fog resource management literature is used to develop the middleware features. Middleware manages the interaction between fog nodes and between the end devices or the cloud to ensure reliability while maintaining heterogeneity.

1.3 Aims and Objectives

The principal aim of the work presented in this thesis is to devise and evaluate middleware framework and its associated services for efficient resource management in fog computing. The proposed middleware enables resource pooling across the fog layer, enhancing application reliability and performance.

To achieve the above aim, it is divided into the following research objectives:

1. Critical Analysis of Resource Management in Fog Computing

   Resource management in fog computing has received a lot of attention from researchers. This objective aims to establish the state-of-the-art to inform the direction of the study. A review of resource management approaches in fog computing identifies key challenges in Fog Computing and requirements for the proposed middleware framework.

2. Definition of Middleware Features

Drawing from the findings of the first objective, this objective seeks to define the properties and features of the proposed middleware based on identified requirements. The design draws from distributed system principles to present a middleware architecture and framework for fog computing.

3. Design and Development of Middleware Components

   Develop middleware components to manage resources in fog computing. Develop and/or adopt techniques considering the trade-offs between various resources and application requirements at the fog layer.

4. Performance Evaluation of Middleware

   The final objective evaluates the middleware using relevant performance metrics. The assessment examines its impact on fog computing services and applications, comparing it against cloud architectures and existing fog computing frameworks. Additionally, the middleware's performance is tested with latency-critical applications.

1.4 Contributions

This thesis makes the following contributions to knowledge:

1. Middleware for Fog Computing

   The middleware includes features derived from current research on resource management in fog computing. Specifically, this work contributes to developing a middleware platform in the context of an intermediary layer in multi-layered distributed system architectures. The middleware is designed to address challenges of heterogeneity, service discovery, resource sharing, and dynamic environment. Furthermore, the middleware supports different architectural environments, which are used at the fog layer, namely: hierarchical, clustered and peer-to-peer architectures.

2. Middleware-enabled Clustered Fog Architecture

This work presents a clustered fog architecture that follows the Internet model of Autonomous Systems. Each cluster is an Autonomous System with all nodes under common administrative control. As part of this model, one feature of the middleware is to enable interoperability across Autonomous clusters. Consequently, load balancing and scheduling within and across clusters is evaluated. The concepts of inter and intracluster resource management using middleware is also developed. The scalability of an application relative to load balancing across different inter-cluster distances for fog clusters is analysed. The results demonstrate that factors such as the distance between fog devices, which is hitherto ignored in the literature, are crucial factors for load balancing at the fog layer.

3. Frameworks and Algorithms for Resource Management among Fog Clusters

   As part of the middleware framework, algorithms for various resource management tasks have been developed and presented in this work. Frameworks have been developed and presented for resource discovery, load balancing and request handling. Specifically, the developed techniques are distinct from existing approaches in two ways. First, they are deployed within clustering and peer-to-peer environments. Second, they are developed as part of an adaptive middleware framework. This is an important contribution to fog research as it shows the benefits of combining different aspects of resource management required for fog computing instead of developing and evaluating specific techniques.

4. Middleware-enabled Peer-to-peer Fog Architecture

   This work also presents a peer-to-peer model for fog computing managed by middleware. Following the results of the work on clustering, further work is done to take advantage of the benefits of peer-to-peer architectures. The peer-to-peer model showed high scalability and utilisation compared to clustering. As there is a dearth of work that explores the use of peer-to-peer architectures for fog computing, this work constitutes an important contribution towards understanding the use of peer-to-peer architectures in the fog environment. Moreover, it also presents a middleware framework for peer-to-peer fog layer collaboration.

5. Simulation Tool for Clustered and Peer-to-peer Fog Architectures

   Extension of Omnet++ with simulation of peer-to-peer and clustered environment in fog computing. This will help other researchers explore further directions of fog/edge computing paradigms under different architectures.

1.5 Thesis Overview

The structure of the rest of the thesis is presented below:

Chapter 2 presents a background and review of the related literature. It also provides an overview of Cloud Computing and the Internet of Things, leading to a focus on fog computing, its features, architecture, and challenges. Core literature on fog resource management and infrastructure (specifically middleware) are discussed to show the gaps the work in this thesis addresses. This chapter situates the work contained in the rest of the thesis within the existing research in fog computing.

Chapter 3 follows the discussion of existing work and the identification of gaps to present the approach and methods adopted in this work to address the gaps. The chapter presents the stages involved in the research and discusses the key decisions made, the tools used and the rationale for these choices. The discussion is also in the existing literature to demonstrate how the adopted approach aligns with existing work. The chapter discusses simulation as a method of system evaluation for distributed systems and discusses its benefits and limitations. The discussion concludes with a presentation of the simulation tool, which is written for parts of this work.

Chapter 4 is the first of the thesis's four core chapters. It presents the details of the middleware model proposed in this work. The chapter starts by analysing resource management in fog computing based on the relevant literature. The analysis leads to presentation and discussion of the middleware's principles and features. Following that, the middleware architecture is presented with a discussion of its components. This chapter represents the focal point for the rest of the thesis. The three chapters that follow it may be read independently as standalone contributions based on the components of the middleware presented in this chapter.

Chapter 5 details and evaluates the middlewares transparency and abstraction principles. It also presents results for evaluating how middleware processes request from IoT applications and presents load balancing among neighbouring fog devices. The chapter also presents the results for evaluating the request handling framework using an EEG application. The contents of this chapter are published in [25].

Chapter 6 extends the work in the previous chapter by introducing clusters under centralised controllers. The focus is on the load balancing and task scheduling functions of the middleware. The chapter also presents a significant contribution of this workthe clustering architecture for fog computing organised as autonomous systems with middleware providing interoperability between clusters. The work in this chapter in published in [26]

Chapter 7 presents the middleware's adaptive feature and shows how it works in different fog architectures. It also presents the peer-to-peer fog architecture. The chapter ends with an evaluation of peer-to-peer architecture, including a comparison of its performance with that of cluster architecture.

Chapter 8 presents a discussion of the key findings of the thesis and their implications in relation to existing work in the field. The chapter ends with a discussion of some future directions for the work.

Chapter 9 concludes the thesis by restating the contributions and implications of the thesis. It ends with some final remarks on the research and its contribution.

CHAPTER 2

Background and Related Work

The primary aim of the thesis is to devise and evaluate middleware framework and its associated services for efficient resource management in fog computing. To achieve this goal, it is important to establish a state-of-the-art basis for the investigation, which is the purpose of this chapter.

This chapter provides a background overview of the relevant related technologies and traces the origins of fog computing, focusing on understanding how characteristics of fog computing necessitate resource management. Following that, the chapter highlights the existing research in the key thematic areas of the thesis, namely resource management, middleware, and reliability, and highlights the gaps that the work presented in the rest of the thesis will address. This chapter thus underpins the thesis by contextualising it in the existing literature.

Structurally, there are six thematic sections with relevant subsections. The first three sections present the background. They consist of sections on the Internet of Things, Cloud Computing and Fog Computing. The final two sections follow the third section and present reviews of related literature on resource management, reliability and middleware in fog computing. Based on the reviewed literature and the state-of-the-art presented, the final section summarises the findings, describes the gaps the work in the thesis will address and links these to the next chapter.

2.1 Internet of Things (IoT)

Kevin Ashton of the MIT Auto-ID Centre introduced the idea of the Internet of Things for the supply chain industry in 1999 [27]. Ashton argued that the potential of computers was limited because they were restricted to only data input from human users. The Internet of Things was thus to allow computers to sense the objects in their surroundings and gather data from objects without relying on human beings who could make errors while inputting the data. The Internet of Things then was to use sensors and Radio Frequency Identification (RFID) chips embedded in everyday objects as sources of data for computers. Six years after the term was first used, the United Nations International Telecommunications Union (ITU) expanded the vision in its 2005 Internet Report for connectivity between anything with anyone at anytime from any place [28]. The concept of IoT has continued to evolve since then.

IoT connects everyday objects to the internet to allow them access or share data or instructions. Typical phases of IoT systems are sensing data, processing to make sense of the data and taking action on the information [29]. RFID tags are used to identify objects as an RFID sensor senses them. With RFID technology, the movement of objects across a supply chain, goods within a supermarket or books in a book store can be tracked with no human intervention [30]. Additionally, sensors and actuators allow objects to interact with people and other objects over the network. Moreover, some objects become smart by processing some of the data they sense [31]. IoT applications have been extended to healthcare [32], power grids [33], transportation [34], agriculture [35], and industry [36, 37]. In 2008, the number of things connected to the internet exceeded the number of humans on the planet [38].

The following are the common protocols used for IoT across the OSI layer stack:

1. **Message Queuing Telemetry Transport (MQTT)** is an International Standards Organisation (ISO) standard publish-subscribe-style application

Table 2.1: Mapping of OSI Model Layers to IoT Protocols and Technologies

OSI Model Layer	**IoT Protocols and Technologies**
Application	MQTT, CoAP, HTTP/HTTPS, AMQP, XMPP, DDS, LwM2M
Presentation	TLS/SSL, CBOR, JSON, XML, Protocol Buffers (protobuf)
Session	MQTT, CoAP, DTLS
Transport	TCP, UDP, QUIC
Network	IPv6, 6LoWPAN, RPL, Thread
Data Link	IEEE 802.15.4, Ethernet, Wi-Fi MAC, Bluetooth LE, LoRaWAN MAC
Physical	IEEE 802.15.4, Wi-Fi (802.11), Bluetooth, LoRa, NB-IoT, Sigfox, RFID, Z-Wave, LTE-M

protocol that works with TCP/IP protocol. It is designed for limited-bandwidth applications.

2. **Constrained Access Protocol (CoAP)** is designed for Machine-to-Machine communications and follows the request-response model.

3. **Advanced Message Queuing Protocol (AMQP)** is an application protocol for transferring messages between organisations. It provides security, interoperability, reliability, queuing and routing.

4. **Wireless Fidelity (Wi-Fi)** IEEE 802.11 is a radio wave communication protocol that serves as a wireless alternative to wired ethernet technology for Local Area Networks (LAN). Wi-Fi operates at 2.4GHz or 5GHz frequencies.

5. **Wi-Fi HaLow**, based on the IEEE 802.11ah specification, is a new low-power, long-range Wi-Fi technology for IoT connections. It operates in the 900 MHz band and consequently has a longer range (up to approximately 1 km) and better propagation (due to its lower frequency).

6. **Bluetooth (IEEE 802.15.1)** Is a low power wireless technology operating at the 2.4 GHz ISM Band. Bluetooth is designed to create Personal Area Networks (PAN). It is designed for low-cost, low-power communications, supporting data rates of 1 Mb/s. It also uses the 2.4GHz frequency.

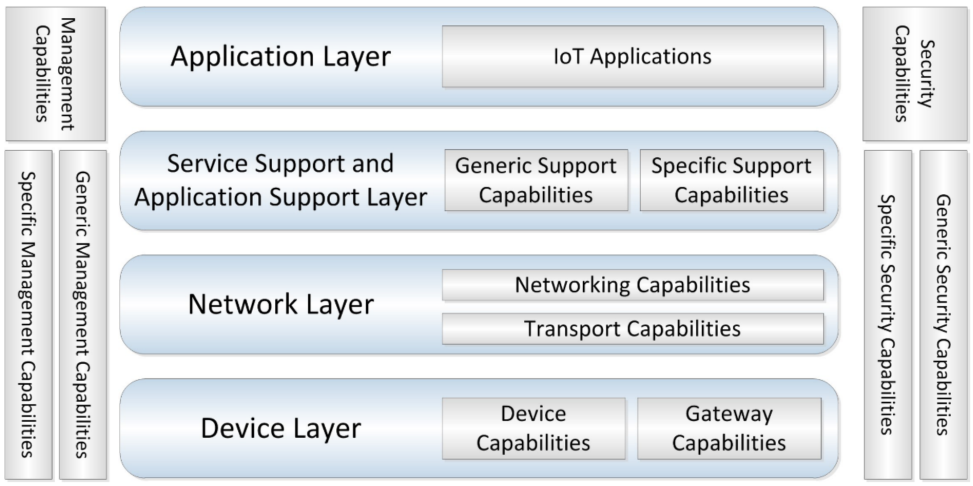

Management Capabilities
Application Layer
IoT Applications
Security Capabilities
Specific Management Capabilities
Generic Management Capabilities
Service Support and Application Support Layer
Generic Support Capabilities
Specific Support Capabilities
Network Layer
Networking Capabilities
Transport Capabilities
Device Layer
Device Capabilities
Gateway Capabilities
Specific Security Capabilities
Generic Security Capabilities

Figure 2.1: ITU IoT Reference Model [42]

7. **RFID:** First invented in 1948 [39]. It is used to identify objects or record metadata. The RFID system consists of a reader and several RFID tags. The tag is a microchip connected to an antenna. The microchip stores data, and the antenna transmits the data stored on the chip to a reader through radio waves at frequencies ranging from 100kHz to 10 GHz.

8. **Zigbee:** A low-cost network with three components: a coordinator which connects the ZigBee network to an existing or larger network, routers which pass on data from other devices in the network towards the coordinator, and end devices, which are devices with limited functionality.

Table 2.1 maps the layers of the OSI model to IoT protocols.

Researchers take different views of describing IoT systems. For instance, Atzori *et al.* [40] identify three aspects of IoT, namely things, the internet and semantics. Semantics refers to how we make sense of the data collected by things. The internet or network is the medium of sending the data to the point of storage or processing. There are also several layered architecture models proposed for IoT. Two common models are the three layer model [41] and the ITUs five layered model [42] shown in figure 2.1.

Xu *et al.* take a functional approach and propose a four-layer Service-Oriented Architecture comprising Sensing, Networking, Service and Interface Layers [43]:

1. **Sensing Layer:** sensor nodes including Bluetooth devices, scalar sensors, analog sensors, digital sensors and RFID tags. Senses or collects data directly interfaces with the environment. The characteristics of this layer are the low-power requirements, and they are characterised by Wireless Sensor Networks (WSN). Similar to the other network devices, IoT devices have a Universal Unique Identifier (UUID)

2. **Networking Layer:** This layer is for communicating with other devices. This is for sending the sensed data to the point of processing to other parts of the system. Networks may be local networks, social networks, databases, WSNs, or the internet.

3. **Service Layer:** This layer offers services to user. It processes data collected. The services layer consists of several services that make use of diverse sources of data, including data from other IoT devices, the internet, etc., to make sense of the data collected. It consists of the business logic that transforms the data into information that produces value.

4. **Interface Layer:** provides a platform for managing the interaction between various components, including users, other devices and services.

IoT connects many things to the network and, consequently, produces huge volumes of data that require computation. The increasing heterogeneity of the devices compounds this challenge. Also, the heterogeneity extends to the protocols they use and the data they produce [21]. Managing the mobility of devices in IoT is another challenge from a data processing perspective [44]. Other areas to be addressed in IoT include interoperability, scalability, and the processing of high volumes of data generated to meet application requirements [36].

As discussed above, IoT devices often have limited capability to process the data they generate. Therefore, they rely on Computation Offloading, which is transferring complex or intensive computational tasks to a remote serveroften, this is to the cloud.

2.2 Cloud Computing

Cloud computing is a computing model that provides computing resources, including hardware and software, to clients on a pay-per-use basis over the Internet. Resources are held in data centres operated and managed by CSPs who make them available to clients with little or no human interaction between the parties. The objective is to make computing resources utilities like gas or electricity. The term *computing as a utility* was first used by Corbato *et al.* in 1965 [45] but was only fully realised at the start of the 21st century when Amazon launched Amazon Web Services (AWS) to make resizable compute resources available over the internet [46].

Computing as a utility reduces capital expenditure for deploying new applications [47]. Developers and researchers can focus on providing technological solutions without worrying about the setup and operation of servers and other infrastructure to keep their systems running. Cloud service providers such as Amazon, Google and Microsoft focus on setting up, maintaining and provisioning hardware and software resources while their clients—corporations, start-ups, researchers, and individuals—focus on deploying their products or solutions [4]. Cloud Service Providers (CSP) make processing, storage, network, and other computing resources readily available to users as and when needed without direct human interaction with the service provider.

The United States National Institute of Standards and Technology defined Cloud Computing. They defined five characteristics, three service models, and four deployment models of Cloud Computing. The essential characteristics are *on-demand self-service, broad network access, rapid elasticity, measured service and rapid elasticity* These are achieved in four deployment models [48]:

- A data centre owned, managed, and used by an organisation providing services within the organisation is a *private cloud.* Large corporations and governments usually deploy private clouds. The resources are not available to the public.

- In *public clouds* resources are made available to the public on a pay-per-use basis. Resources can be requested and accessed by anyone with internet access.
- A *Community cloud* is owned by a group of organisations with a common interest, like education or health institutions. They are setup for the collective interest of the ownership. The resources are either managed by the owners or by a third party on their behalf.
- *Hybrid clouds* are a combination of two or more of the other models. Hybrid Clouds are often used when private cloud owners require additional resources from public clouds due to increased demand for resources or failure [49].

Cloud Computing providers use three main service models to provide services to clients — (i) Infrastructure-as-a-Service; (ii) Platform-as-a-Service; and (iii) Software-as-a-Service [48].

Under Infrastructure-as-a-Service or IaaS, a service provider provisions hardware resources to the client. The client sets up and runs the System Software and other software required to make the system functional. With Platform-as-a-Service, the service provider provides tools for clients to develop and run their own software on the providers platform. The client is not concerned with the system software behind the system. Applications deployed by the client must be compatible with the platform of the provider. In Software-as-a-Service, the service provider makes specific applications or resources available to clients. The client uses the application and resources without direct access to the system the application is running on.

An essential feature of Cloud Computing Systems is elasticity. Elasticity enables more resources to be provisioned when they are needed and released or given back when they are not [50]. In 2008, the Washington Post made 17,481 pages of documents from the US National Archives publicly available 9 hours after they were released. The Newspaper was able to do this using 200 computers from Amazon EC2 [51]. Although elasticity creates the perception of the availability of an infinite number of resources to the user, there is a limit to the resources service providers can provide. Moreover, energy consumption by data centres impact environmental sustainability [52].

Despite its success, cloud computing faces challenges that may impact its continued patronage and growth. Security and Privacy concerns arise from how cloud resources are shared and managed [53]. Also, applications in fields such as healthcare, emergency services and transportation, which are delay-sensitive, cannot cope with the high latency resulting from running processes on servers in remote data centres [54]. For Cloud Computing to continue its growth and impact, these challenges need to be addressed [24]. Moreover, as discussed in the previous section, the increasing number of networked devices from the Internet of Things (IoT) and other technologies significantly increases the demand for cloud resources [10].

2.3 A Brief Historical Perspective

The origins of the paradigms proposed to address the challenges identified with Cloud Computing can be traced back to the early days of the cloud itself. As with other most new technologies, the convergence of different technologies which target unrelated issues led to their emergence. This section traces the origins of paradigms which attempt to address these challenges. The distinguishing characteristics of each is highlighted leading up to Fog Computing and how it is unique to the others. To restate the problem, cloud computing faces challenges as more IoT devices and new internet applications are deployed. These include dealing with the high volumes of data produced by IoT devices, meeting requirements of delay-sensitive applications, and ensuring quality of service and service level agreement guarantees are met even when the system is overloaded. These challenges may lead to reduced patronage of cloud computing as users do not get the level of service they expect from using the cloud.

The earliest technology to target these challenges predates cloud computing; it arose in response to a challenge identified with another disruptive technology: the mobile computing. Similar to IoT devices, although mobile phones had the potential to disrupt several fields, such as healthcare and entertainment, they were limited by their computational capabilities and battery life [55]. To realise the potential of mobile phones, Mahadev Satyanarayanan proposed *Cyber Foraging* in 2001 [56]. Cyber Foraging offloads computationally intensive tasks from mobile devices to capable servers for processing. When Cyber Foraging was first proposed, there was no technology to provide the capable server required until Cloud Computing emerged a few years later, birthing *Mobile Cloud Computing.*

Through Mobile Cloud Computing, mobile phones became more capable, leading to the emergence of different applications and services such as social media and mobile healthcare. However, the requirement for improved latency for some mobile applications led to the emergence of *Cloudlets* [57], computing resources that are resource-rich and available to nearby mobile devices. This was the start of Mobile Edge Computing (MEC) [58]. Mobile devices made use of nearby cloudlets or used the cloud where there were no cloudlets available. *Edge Computing* was coined as the use of cloudlets extended to other end-devices apart from mobile phones [59]. Also, Mobile Edge Computing has been renamed Multi-Access Edge Computing [60].

It is important to note that MEC is well-suited to cell phones because phones have some computational resources, although they are limited. Consequently, they can interact with nearby cloudlets and resort to the cloud when no cloudlets are available. On the other hand, IoT devices range across every kind of device and include devices which have no computational capabilities. IoT also covers a wide area and often consists of devices collaborating within their context to provide services, while mobile phones often act independently. This distinction led to the emergence of Fog Computing [58].

What all these computation offloading paradigms have in common is the use of computation resources closer to users - where data is produced. Fog Computing is unique as it provides multiple tiers of computing between the cloud and the user, which makes it scale vertically as well as horizontally [61, 62]. Also, unlike its precursors, fog computing is not independent of the cloud. It works with the cloud; it does not exist or work independently of the cloud [63]. The multi-tiered, coordinated approach of the fog layer raises complex management issues which are interesting from a distributed systems perspective and require further investigation.

2.4 Fog Computing

Fog computing was first introduced by Cisco in 2012 as a complementary paradigm to cloud computing [3]. Fog uses compute, network and storage resources across the network on devices such as gateways, switches and routers to perform some processing, management and storage for the cloud [64].

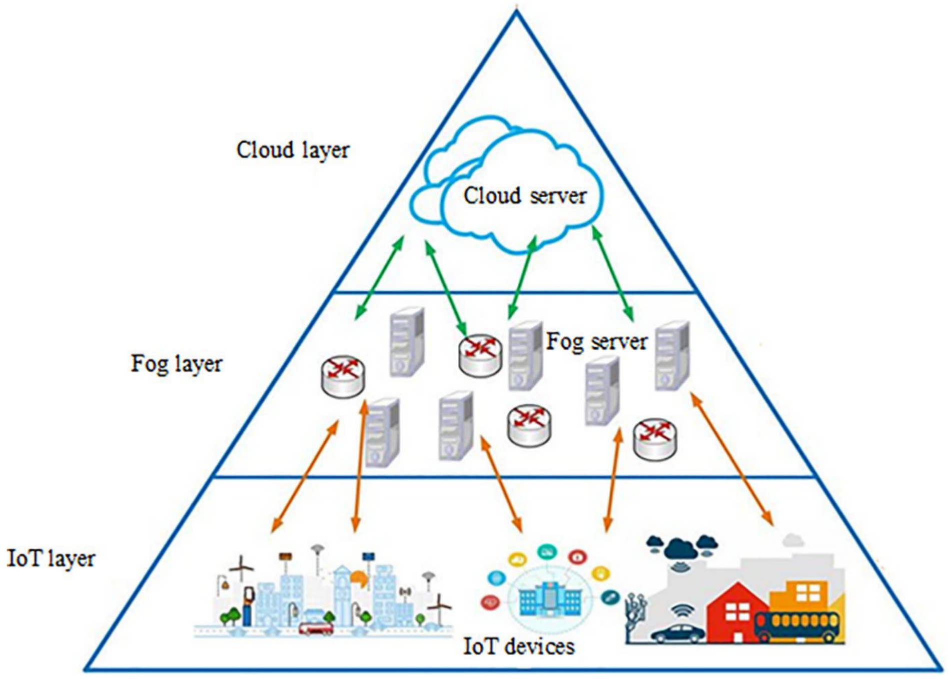


Figure 2.2: Fog Architecture

Figure 2.2 shows a fog architecture with three layers. The lowest layer, the IoT and end-user layer, has IoT and end devices. Certain applications have either IoT devices or end devices only at this layer and not both [19]. The middle layer, or fog layer, has devices with varying computational power, storage and network capabilities. The devices may be grouped into domains. Each domain has several fog nodes. The fog layer may have multiple tiers with increasing capabilities the closer they are to the cloud. Since fog nodes are closer to the user and have varying characteristics, fog computing is inherently heterogeneous and more context-aware than the cloud. The third or top layer of the architecture is the traditional cloud data centre. This layer has powerful servers for heavy computational, storage and network tasks. However, they are far from IoT and end devices. Moreover, the cloud layer maintains a system-wide view of the system.

Fog computing benefits both cloud data centres and end-users by reducing the workload of cloud data centres and reducing users' latency. Devices at the fog layer complement cloud servers by carrying out defined tasks and passing more demanding and less delay-sensitive tasks to cloud servers. This arrangement resolves the challenges identified with cloud computing. First, by processing the data from IoT close to the data source, more contextualised problem-solving is possible, and the cloud can process and store data for longer-term and system-wide support. This reduces the pressure on the network towards the cloud and, consequently, on cloud servers. Secondly, for delay-sensitive applications, processing on nearby devices reduces the Network Propagation Delay and, thus, the latency of applications. By adding a layer closer to the data source in the cloud architecture, fog computing addresses the significant challenges associated with traditional cloud computing.

In line with its definition as an extension of the cloud, NIST proposed three service models for fog computing [65]: *Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS).* They also define deployment models mirroring those of cloud computing: *private fog node, community fog node, public fog node, and hybrid fog node.* Furthermore, the NIST conceptual model identifies autonomy, programmability, manageability, heterogeneity and hierarchical clustering as the key attributes of fog computing.

2.4.1 Fog Computing Features

Features of fog computing include heterogeneity, low latency, location awareness, geographical distribution and mobility. These are discussed briefly here.

Heterogeneity

Fog Computing uses resources on devices at various layers of the network to provide services to end users and the cloud. Devices vary in several ways, including processing power, network capabilities, proximity to end-users and mobility. Additionally, the network infrastructure also varies from high-speed fibre links to wireless access technologies such as Wi-Fi, 4G, and ZigBee. Furthermore, devices in a fog system are run and owned by different providers. In a clustered architecture, for example, each cluster may be operated by a different provider with a different platform from other domains [26].

Low Latency

Fog nodes may be any device across the network: routers, switches, gateways, roadside units, etc. As these devices are closer to the end-user than are cloud servers, processing on fog nodes will reduce the response time users experience. This is especially useful for Ultra-Reliable Low-Latency Communications (URLLC) applications in which the delay in reaching and getting a response from a remote server affects the Quality of Experience of the user [9, 66]. This reduced latency will be crucial for emerging technologies such as Tactile Internet [67].

Geographical Distribution

Fog computing takes advantage of resources across various levels of the network, which are closer to and at the edge. This brings together devices from a widespread geographical area. It also provides a distributed, decentralised architecture with nodes carrying out specified tasks for user devices within their defined area and sending more demanding tasks to a central server [68].

Context-Awareness

Due to the geographical distribution of fog nodes discussed above, fog nodes can provide a more accurate context of the end user(s) they serve compared to centralised cloud servers. Context-specific data such as location, activity of other users and network load are helpful for specific applications such as Intelligent Transport Systems (ITS) [34]. Context information can be exploited to improve services and resource allocations. Each fog node is responsible for a pre-defined area or task, and so the location of the end user is linked to the fog node. In a traditional cloud computing setup, all users connect to a centralised location or set of locations, which makes it difficult to provide location-tailored solutions.

Mobility

Mobility in Fog Computing can be viewed in two ways: supporting mobile end-user devices [69] and mobile fog nodes [70]. Since fog nodes are distributed and coordinated to support the cloud, they are best placed to support situations where users move from location A to B while still having a request on a Fog Node in location A. Several researchers have proposed mechanisms for dealing with mobility support, especially for transport applications [71, 72]. Additionally, mobile fog nodes are expected in fog computing. In specific applications, such as transport, some scenarios involving mobile fog nodes have been proposed [73]. These devices have processing, network and storage resources that are available to end users, but they are also mobile and may change their location.

The above features make Fog Computing best suited for new IoT applications that involve real-time interaction between users and processing points. Several studies have proposed fog-based application areas and case studies. The following section discusses the use of fog for IoT in industry, healthcare and transportation.

2.4.2 Fog Applications

Fog computing improves the quality of applications in domains such as industry, smart cities, transport, and healthcare. This section discusses fog-IoT applications in industry, transportation, and healthcare.

Industry

The fourth industrial revolution, also known as Industry 4.0, is fundamentally transforming manufacturing and industrial operations through the integration of cyber-physical systems, IoT devices, and advanced computing paradigms like fog computing. The first industrial revolution was driven by mechanisation and steam power. Later, the discovery of electricity made the assembly line possible and ushered in the Second Industrial Revolution. The third industrial revolution used computers for automation in the manufacturing process. Now, in the fourth industrial revolution, Industrial IoT (IIoT) has introduced Cyber-Physical Systems (CPS) and is making machines smart.

Smart Manufacturing and Factories: Smart factories leverage extensive sensor networks to monitor and automate processes on the factory floor [74]. Fog computing plays a crucial role in enabling real-time decision-making by processing sensor data locally rather than sending it to distant cloud servers [75]. For example, in automotive manufacturing, fog nodes can process vibration data from assembly line equipment to detect anomalies and prevent costly breakdowns before they occur. This local processing reduces latency from hundreds of milliseconds to single-digit milliseconds, enabling immediate corrective actions [76].

Predictive Maintenance: Industrial fog computing enables sophisticated predictive maintenance strategies by analysing equipment sensor data in real-time. Machine learning models running on fog nodes can detect patterns indicating impending equipment failure, allowing maintenance to be scheduled proactively rather than reactively. This approach reduces unplanned downtime by up to 50% and maintenance costs by 10-40% compared to traditional scheduled maintenance approaches [77]. Recent studies demonstrate that fog-enabled deep learning models in smart factories can predict equipment failures with high accuracy, reducing downtime and maintenance costs by 60% [78].

Quality Control and Inspection: Computer vision systems deployed on fog nodes can perform real-time quality inspection of manufactured products. High-resolution cameras capture images of products as they move through production lines, and fog-based image processing algorithms identify defects within milliseconds. This enables immediate rejection of faulty products and adjustment of manufacturing parameters, significantly improving overall product quality and reducing waste.

Energy Management: Industrial facilities consume significant amounts of energy, and fog computing enables intelligent energy management through real-time monitoring and optimisation, as fog computing aims to reduce energy consumption in industrial sensor networks [75]. Fog nodes can process data from smart metres, environmental sensors, and equipment monitors to optimise energy consumption patterns, integrate renewable energy sources, and participate in demand response programmes.

Supply Chain Optimisation: Fog computing enhances supply chain visibility and responsiveness by processing RFID, GPS, and sensor data from goods in transit. Local processing enables real-time tracking of inventory levels, environmental conditions during transport, and automatic reordering when stock levels reach predetermined thresholds.

Process Optimisation: In continuous manufacturing processes such as chemical production or steel manufacturing, fog computing enables real-time process optimisation by analysing sensor data from temperature, pressure, and flow sensors. This local processing capability allows for immediate adjustments to maintain optimal operating conditions and product quality whilst minimising energy consumption and waste generation.

Transportation Systems

The need to efficiently manage and improve security in transportation systems motivated the concept of Intelligent Transportation Systems (ITS) [79]. Approximately eight million traffic accidents occur annually and are responsible for 1.3 million fatalities [80]. Also, in 2017, UK drivers wasted 31 hours in rush-hour traffic, costing each motorist over one thousand pounds [81]. Likewise, the European Union (EU) spent 4% of its Gross Domestic Product (GDP) on transportation issues in 2011 [82]. Intelligent Transportation Systems use modern sensing and communication technologies (IoT) to improve transportation and transport management.

Vehicular Ad-hoc Networks (VANET) enable communication between vehicles and other vehicles and between vehicles and road infrastructure, cloud servers and users [83]. VANET has evolved into the Internet of Vehicles [84]. To process data in the Internet of Vehicles efficiently and close to the data source, researchers have proposed fog-based vehicular systems. In [85], Fog Vehicular Computing uses packed vehicles with computing resources to process data produced by other vehicles and end devices. The authors in [86] proposed Vehicular Fog Computing for a city-wide transportation management system. Fog computing has also been proposed to manage transportation security. Neto *et al.* [87] proposed a fog-based crime detection system for transportation systems. Furthermore, Darwish *et al.* [80] proposed an architecture for managing and analysing big data in Intelligent Transportation Systems.

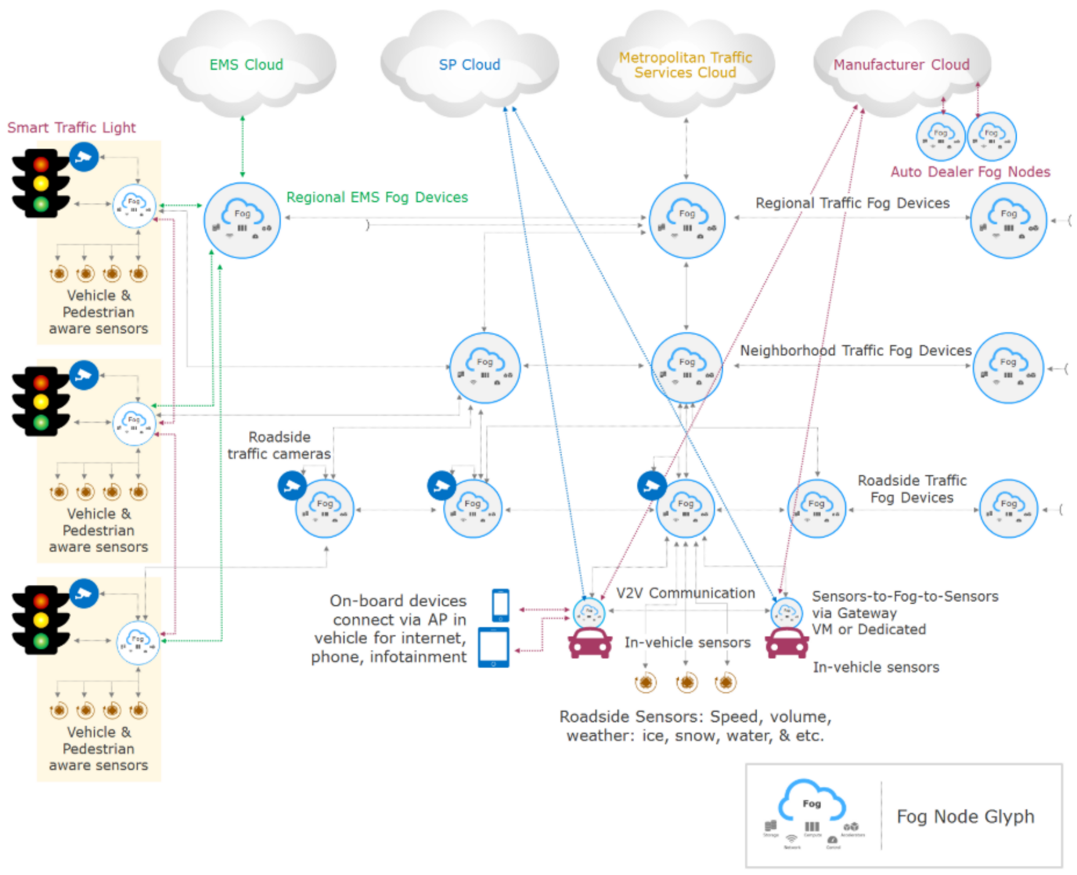

EMS Cloud
SP Cloud
Metropolitan Traffic Services Cloud
Manufacturer Cloud
Smart Traffic Light
Auto Dealer Fog Nodes
Regional EMS Fog Devices
Regional Traffic Fog Devices
Vehicle & Pedestrian aware sensors
Neighborhood Traffic Fog Devices
Roadside traffic cameras
Roadside Traffic Fog Devices
V2V Communication
On-board devices connect via AP in vehicle for internet, phone, infotainment
Sensors-to-Fog-to-Sensors via Gateway VM or Dedicated
In-vehicle sensors
Roadside Sensors: Speed, volume, weather: ice, snow, water, & etc.
Fog
Fog Node Glyph

FIGURE 2.3: OpenFog smart car and traffic control system [63]

Figure 2.3 shows the OpenFog smart car and traffic control system [63]. The figure shows how different aspects of an intelligent highway system may work together. Drivers and passengers access the internet using an access point in the vehicle. Smart vehicles communicate with themselves using VANET's vehicle-to-vehicle communication. They also send relevant sensed data to manufacturers, service providers, and various cloud servers. Smart vehicles cannot rely on the cloud to process real-time data because the delay may lead to slower decision-making by vehicles, resulting in late decision-making and undesired outcomes. Smart traffic lights control traffic intelligently using data from roadside traffic cameras, roadside units, and vehicle and pedestrian sensors. Moreover, data is sent to metropolitan traffic cloud servers to support long-term decision-making for the entire system. The system shows how fog computing will work with cloud computing rather than replace it entirely.

Healthcare

The authors in [88] identified five ways technology is used in healthcare: health promotion and disease prevention, diagnosis, monitoring, treatment and supporting health services. They also found that the dominant m-health technologies in an era was based on the predominant devices and technologies used in that era. For example, in the early 2000s Personal Digital Assistants were popular and thus PDA applications for m-health dominated the research on m-health. Also, in recent years mobile apps have dominated due to the popularity of smartphones. Advances in health IoT devices are expected to drive m-health applications [89]. M-Health applications range from disease management applications such as measuring blood sugar level to control insulin dosage (in the management of diabetes), monitoring vulnerable patients through fall detection and tracking fitness with wellness applications [90].

IoT applications in healthcare present unique challenges that cannot be resolved with traditional cloud computing. One such challenge is the low latency requirement of some health applications. In emergencies, the difference between life and death may be down to a few milliseconds. Also, applications such as ECG monitoring produce constant streams of data, which may not be useful to store in the cloud. Such data is monitored to detect abnormal signals which require medical attention. Sending all that data to the cloud is counterproductive. Furthermore, medical sensors are expected to be lightweight with small form factors to ensure they are not cumbersome for the user to carry around [91]. This requirement makes the option of processing within the sensor infeasible as including processing and storage components will increase the weight and form factor of sensors. Additionally, the risk of failure is higher when data is sent for processing several hops away from the data source. In [92], the authors recorded a 50% frame loss rate when testing their emergency pre-hospital communication system using cloud servers. They proposed edge processing of the data as a possible solution.

It can be seen from the above discussion that there is a wide range of domains in industry, transportation, and healthcare that use or could be improved with IoT

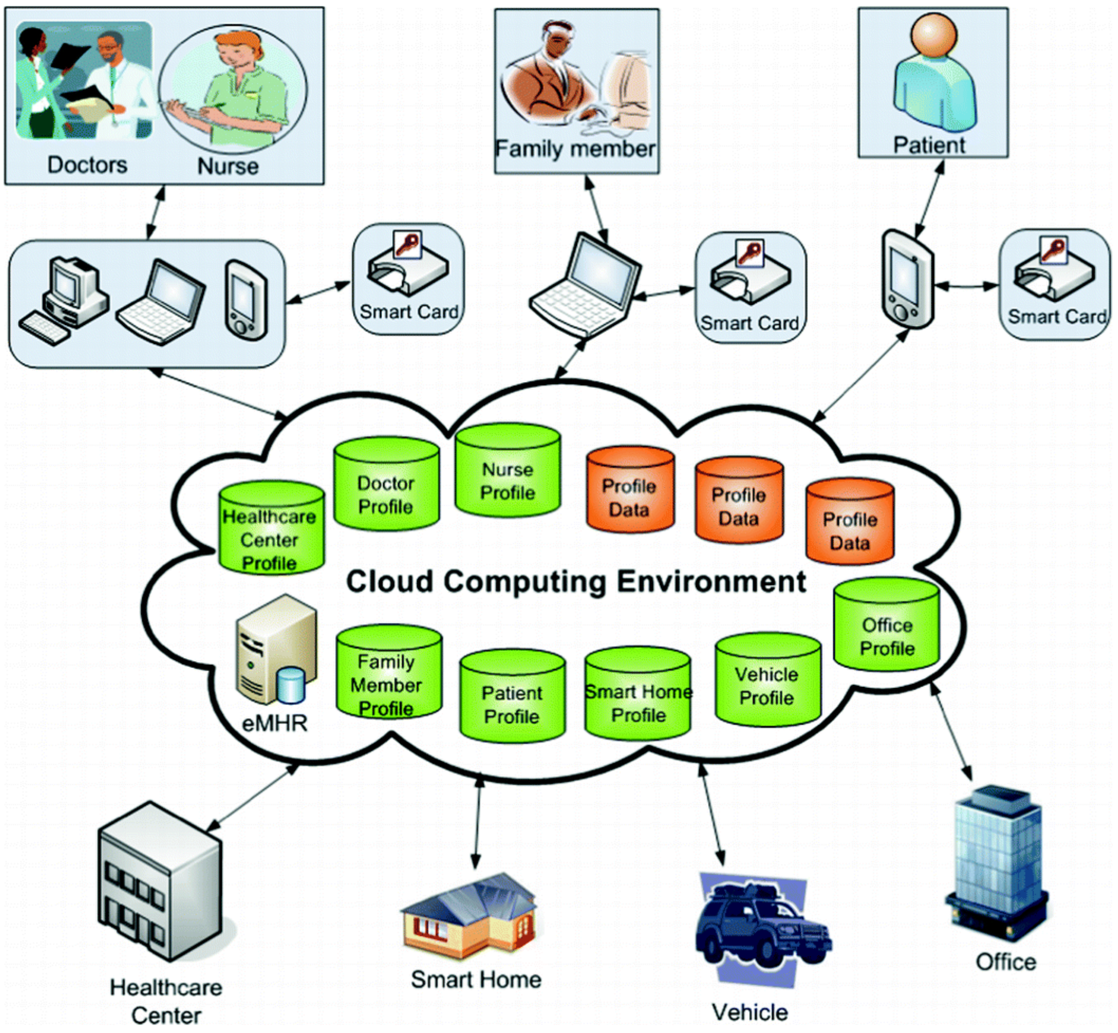


FIGURE 2.4: Ubiquitous Healthcare System with Cloud [93]

and cloud computing. Enhancing cloud computing with fog extends the possible applications even further. The healthcare domain is unique because it encompasses several different varieties of use cases, each with its own set of requirements. Figure 2.4 shows ubiquitous healthcare system. With the increasing numbers of IoT devices, all expected to use the internet and the inherent challenges with some healthcare applications, a thorough investigation of healthcare requirements is required to inform the design of suitable system management schemes such as schedulers for Fog Computing. In chapter 5 a use case of fog computing with Electroencephalography (EEG) monitoring is presented.

2.4.3 Fog Architectures

A review of the literature on fog computing architecture shows a wide range of approaches and designs. The variety stems from two facts about fog computing. First, fog computing as a computing paradigm is still in its infancy and has not yet fully developed. To the best of this researchers knowledge there are no real-world full implementations of fog yet. Architectures for fog are therefore constantly being proposed and refined by research communities. Secondly, fog computing encompasses several fields associated with computing, and each of these fields have their own approaches and styles for developing architectures. The different architectures are thus a reflection of the different notions of architectures in the fields involved [94]. Fog computing involves networking, programming, hardware, services to name a few. Each of these fields have different communities and views of the concept of architecture. Mann [94], presents an analysis of the different views of architecture in fog computing and proposes a three-dimensional framework for engaging with fog computing architectures. The three dimensions of Manns framework are the devices dimension, the system dimension, and the functionality dimension.

The device dimension is often the focus of the networking community and comprises different nodes and the links between them. The device dimension has a widely accepted three-layer architecture shown in figure 2.2. Most proposed device architectures are slight variations of the three-layer architecture. Variations have been made to the bottom layer [95] and the fog layer [96]. The systems dimension describes the hardware, software, and service layers for fog. This dimension includes proposals for virtualisation, middleware, applications, and operating systems. The functionality dimension includes the architecture of specific components that make fog functional. These include resource and service management as well as application-specific functional architectures, which have been proposed for specific applications like healthcare and smart cities. Mann [94] points out that these dimensions are independent from each other, although each of them is important in all fog systems. This section presents a review of some fog architectures related to resource management and middleware. Although resource management is a functional task under the functionality dimension and middleware fall under the system dimension in Manns framework, it is important to present a general discussion of fog architectures to properly situate resource management and middleware in fog systems.

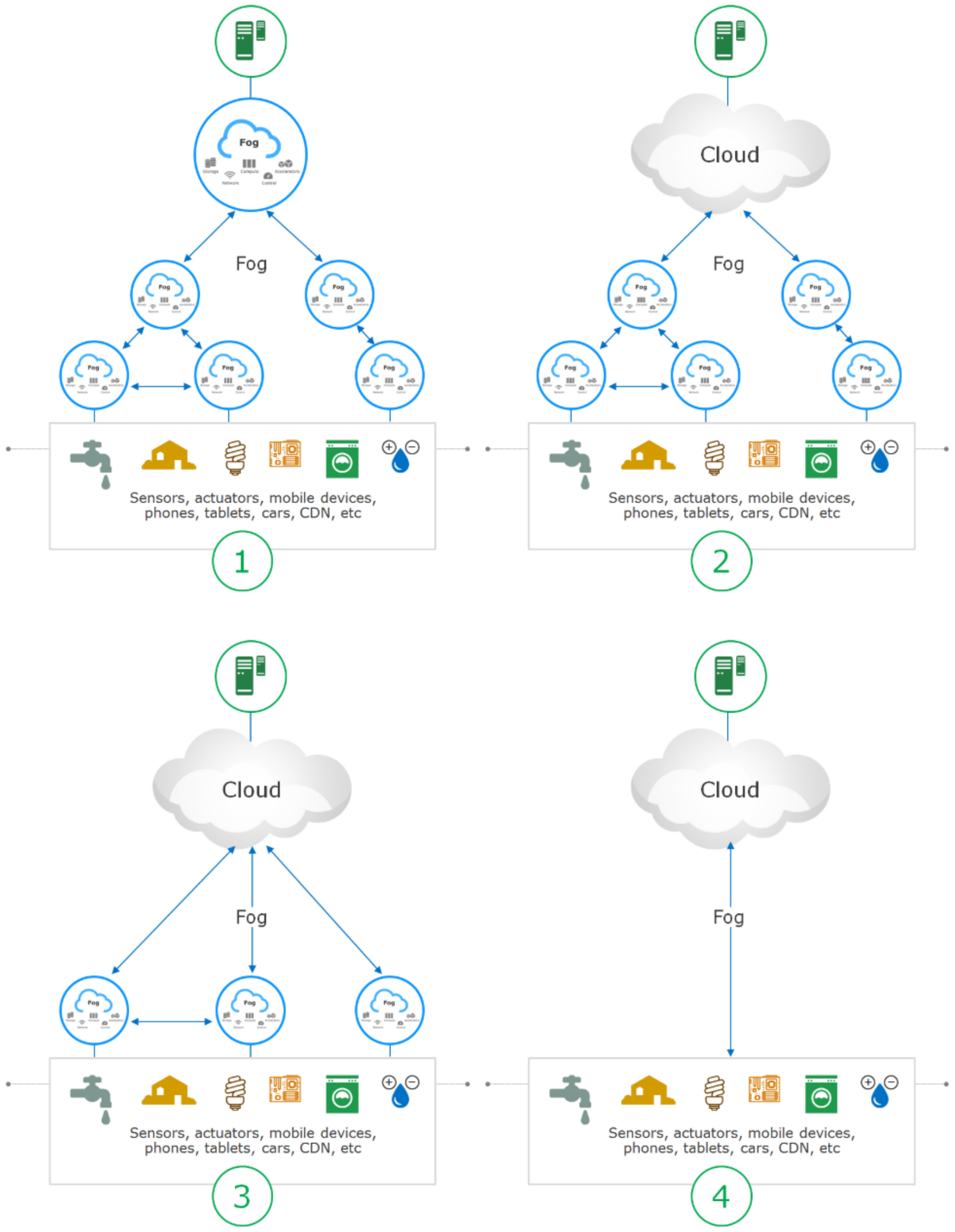

Fog
Fog
Sensors, actuators, mobile devices, phones, tablets, cars, CDN, etc
1
Cloud
Fog
Sensors, actuators, mobile devices, phones, tablets, cars, CDN, etc
2
Cloud
Fog
Sensors, actuators, mobile devices, phones, tablets, cars, CDN, etc
3
Cloud
Fog
Sensors, actuators, mobile devices, phones, tablets, cars, CDN, etc
4

FIGURE 2.5: The Four our Fog Deployment models [63]

The OpenFog Consortiums Fog Reference Architecture has been adopted by the IEEE as IEEE Standard 1934-2018 [63]. It consists of multiple layers of fog nodes between IoT things and the cloud. Sublayers of the fog layer have varying degrees of computation power, storage, and other resources available to the system. Each layer of fog nodes may carry out some level of computation, and higher layers, that is, those closer to the cloud, carry out more demanding tasks. The reference architecture also presents four fog deployment models, as shown in figure 2.5. In the first model, the system is independent of the cloud. This is useful for applications that cannot use the cloud due to regulatory requirements, security, or unavailability, such as military or healthcare. In the second model, the system relies on cloud servers for tasks that are not delay-sensitive, while the fog layer carries out operational functions and some delay-sensitive tasks. This model is envisioned for use cases such as building management and retail. In model three, the fog layer performs delay-sensitive processing tasks, and the cloud performs operational and control functions. Model four is like a traditional cloud deployment; the cloud does most of the processing. The role of fog, if any, is limited to monitoring and safety control.

Other reference architectures include the Clouds Lab Architecture [97] and the Cisco Architecture [3]. What these architectures have in common is a view of Fog as a layer with limited resources providing services to both user devices and the Cloud. The resources provided by fog nodes are limited, and in applications such as healthcare, would be more critically needed by some users than others. Also, the fog nodes at a given layer may be capable of coordinating with each other and with other nodes along the continuum [98]. This presents a corporative system with resources available from different fog nodes available to the end user. Several application-specific architectures have also been proposed for fog computing. Architectures have been proposed for healthcare [99], smart cities [96]. Mouradian *et al.* [19] grouped fog architectures into application specific and application agnostic architectures.

2.5 Reliability

Reliability is the ability of a system to behave as expected and complete tasks promptly over a specified period. It is primarily user-centric. Users judge the reliability of a given system differently because of different requirements and expectations from the system. The reliability of software systems is linked to failure and recovery from failure. Reliability metrics include:

1. Mean Time to Failure (MTTF): the average time for system failure.
2. Mean Time to Repair (MTTR): the average time it takes to repair the system when it fails.
3. Mean Time Between Failures: how long the system is unavailable. It is the sum of MTTF and MTTR
4. Availability: the probability that the system is in operation at a given time.

The regular reliability models for traditional software cannot be applied directly to the cloud computing environment [100]. For cloud service users, service availability is an important consideration for reliability. Cloud Providers pledge near-perfect availability (99.99%) in SLAs [101].

Cloud service users adopt fault tolerance strategies such as redundancy to improve reliability [102]. Dai et al. [103] analysed cloud computing and developed reliability models for cloud services. There is currently no comprehensive model for reliability in fog computing in the literature.

2.6 Middleware

Gateways are an integral part of IoT [44, 104]. They serve as a bridge between IoT sensors and the internet [104]. Gateways are usually the first hop from an IoT device towards the internet. They solve the challenge of communication heterogeneity between IoT sensors and mobile communication networks or the internet by providing interfaces for multiple network protocols, such as Bluetooth, ZigBee, 6LoWPAN, etc., at the IoT end and internet protocol towards the internet. Thus, the Gateway receives data from various sensors of diverse protocols and sends them using LTE, 5G, DSL, Wi-Fi etc. over IP network to fog or cloud servers. Moreover, in fog computing, gateways have assumed additional functions [29]. Aazam *et al.* [105–107] present the smart gateway, a gateway that preprocesses data received from sensors before forwarding to fog nodes or cloud servers. The smart gateway is a fog node because it performs some tasks (data filtering, trimming, reconstruction, etc.) before forwarding data to the cloud. The smart gateway as a fog device has been implemented for healthcare applications [108]. Despite being closest to the data source and thus having the quickest connection time to end devices, gateways usually have much limited resources compared with other devices higher up the architecture, therefore, they are limited in the workload they can take on to support the cloud in a fog system. Smart gateways have the advantage of being closest to IoT devices, however, because they serve as a means for IoT devices to access services, they constitute a single point of failure in an IoT system.

Middleware has also been proposed to provide an abstraction layer to solve architecture mismatch problems associated with connecting IoT Systems with cloud computing [24]. Figure 2.6 shows the role of middleware for the Cloud of Things (CoT) proposed by Farahzadi *et al.* Middleware hides the complexity and heterogeneity of a system and makes it easier to develop applications for it [109]. Nastic *et al.* [110–112] have introduced provisioning middleware for IoT clouds at the edge, similar to fog computing. Their middleware has components in the cloud and on gateway devices close to IoT devices.

2.6.1 Key Middleware Challenges in Fog Computing

The integration of middleware in fog computing environments presents several critical challenges that current approaches have not adequately addressed:

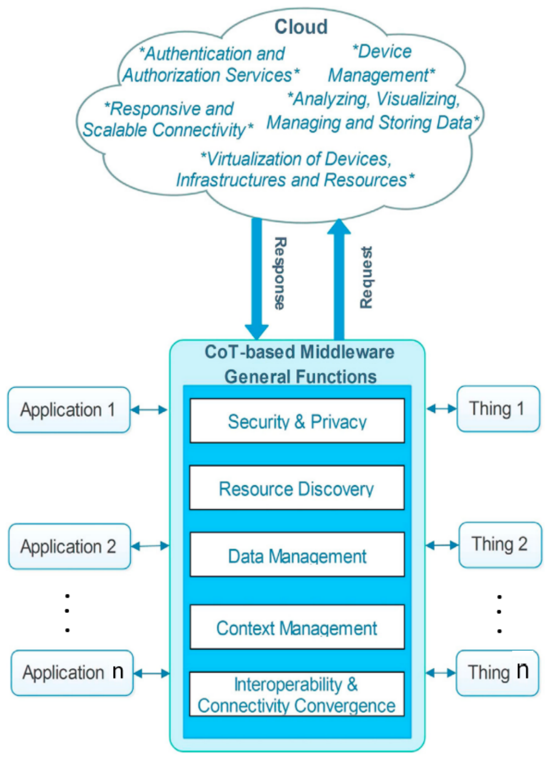

Cloud
Authentication and Authorization Services
Device Management
Responsive and Scalable Connectivity
Analyzing, Visualizing, Managing and Storing Data
Virtualization of Devices, Infrastructures and Resources
Response
Request
CoT-based Middleware General Functions
Security & Privacy
Resource Discovery
Data Management
Context Management
Interoperability & Connectivity Convergence
Application 1
Application 2
Application n
Thing 1
Thing 2
Thing n

FIGURE 2.6: Role of middleware in Cloud of Things [24]

Resource Management Complexity: Existing gateway-based middleware solutions struggle with efficient resource allocation across heterogeneous fog nodes. Smart gateways, whilst providing local processing capabilities, have limited computational resources that constrain their ability to handle complex workloads or coordinate with other fog nodes.

Reliability and Fault Tolerance: The single point of failure problem inherent in gateway-centric approaches creates reliability vulnerabilities. When a smart gateway fails, all IoT devices depending on it lose connectivity and services, highlighting the need for distributed, fault-tolerant middleware architectures.

Heterogeneity Management: Current middleware solutions provide basic protocol translation but lack comprehensive management of resource heterogeneity across fog nodes with varying computational capabilities, storage capacity, and network connectivity.

Cooperative Resource Utilisation: Existing approaches do not fully exploit the potential for cooperation between fog nodes. Most middleware implementations treat gateways as isolated processing units rather than components of a collaborative fog infrastructure.

These challenges directly motivate the middleware approach proposed in this thesis. Unlike existing gateway-centric solutions that focus on individual device functionality, the proposed middleware provides:

Distributed Resource Management: Rather than relying on resource-limited gateways as single processing points, the middleware coordinates resource allocation across multiple fog nodes to optimise system-wide performance.

Fault-Tolerant Architecture: By enabling fog nodes to cooperate and provide backup services for each other, the middleware eliminates single points of failure inherent in current smart gateway approaches.

Comprehensive Abstraction: The middleware hides not only protocol heterogeneity but also resource heterogeneity, enabling applications to utilise fog resources transparently without knowing which specific fog node executes their tasks.

Cooperative Processing: The middleware enables fog nodes to share workloads and resources dynamically, maximising utilisation of available resources across the fog layer rather than being constrained by individual gateway limitations.

2.7 Related Work

The previous sections presented an overview of IoT, cloud, and fog computing. Their goal was to provide a background to the areas of research this thesis will contribute to. This section will build on this background by presenting the current relevant literature and highlighting the gaps in knowledge. The section will end with an analysis of these gaps and restate the aim of this work.

2.7.1 Resource Management in Fog Computing

Resource management at the fog layer is critical for two main reasons. First, unlike the cloud, the fog layer is resource-constrained. Most devices at the fog layer have a different primary function within the network; working as a fog node is a secondary function [62]. They may also have smaller form factors and processors [113]. Secondly, fog nodes are heterogeneous in various ways. They differ/vary in architecture, resource availability, power and speed, to name a few. The processing, network and storage resources at the fog layer must, therefore, be managed to achieve the desired level of performance. The CPU, memory, network, virtual machines, and energy resources at the fog layer must be managed to ensure that processing can be done by fog nodes without impacting the networking or other functions of fog nodes and without violating Service Level Agreements for applications. Furthermore, at the fog layer, several fog nodes in a domain may be able to share resources for efficient execution of tasks [98, 114]. In such a scenario, the resources within the fog domain must be managed for efficient utilisation.

In an early work on resource provisioning in fog computing, Agarwal *et al.* [115] used a fog server manager to allocate resources on fog nodes to service requests from clients. In their architecture, requests which cannot be serviced by a fog node are forwarded to the cloud. There is no cooperation among fog nodes. Kimovski *et al.* [116], proposed an architectural approach modelled after the human brain to achieve adaptive resource management at the fog layer. Task offloading across different layers in the fog architecture has been investigated by researchers as a resource management approach. Papers [117, 118] adopt greedy heuristic approaches to offloading tasks. Mahmud *et al.* [119] tackle the problem from a Quality of Experience (QoE) perspective. They use fuzzy logic to prioritise application placement requests. In [62], ENORM, a framework for dynamically managing resources on edge nodes is presented. Resource Management on ENORM, however, involves the cloud and is therefore not ideal for applications that are delay-sensitive. Also, ENORM focuses on resource management on a single fog node; it does not do so for multiple nodes cooperatively.

In paper [98], Zhang *et al.* adopt a hierarchical approach to resource management in a cooperative fog computing system for intelligent transportation systems. Resources are managed by a coordinator fog node which collects relevant information from other nodes within a domain. Reliance on a coordinator server, however, presents a single point of failure for the system.

2.7.2 Reliability in Fog Computing

In an early work on reliability in fog computing, Madsen *et al.* [120] discussed how reliability is achieved and evaluated in cluster, grid and cloud computing and extended the discussion to reliability in fog computing. The discussion on fog, however, focuses on Wireless Sensor and Actuator Networks. Checkpointing, replication and rescheduling are used to ensure reliability in cluster and grid systems. The authors in [121] also briefly present a mathematical model of network reliability in Multi-state Cloud Networks with fog nodes. They also present an algorithm for evaluating reliability in such systems.

Two works focus mainly on the reliability of fog computing itself. Firstly, Popentiu-Vladicescu and Albeanu [122] reviewed the reliability of fog computing, looking at three aspects: reliability of fog nodes(hardware), reliability of network and reliability of software. They present a brief discussion of the importance of reliability and why it is crucial for fog IoT applications. Furthermore, they look at the reliability of IoT sensors and their impact on the system's dependability. They also discuss the Open Fog Reference Architecture and its RAS pillar (Reliability, Availability and Serviceability).

The other work on reliability in fog computing is reported in paper [123]. The paper identifies features of mobile operating systems that make them unreliable for use as fog nodes. Using the Android operating system, the authors provide solutions to the problems identified. The goal of their solution is to make the Android OS capable of running applications with strict time requirements. The challenges they identified are the lack of prioritisation in the software stack of mobile OS and the presence of services that preempt running applications, resulting in pauses during execution. The changes they make in Android include allowing developers to specify the time requirements of the applications, incorporating priorities in the communication model, implementing pause-less memory management and specifying real-time behaviours in APIs.

2.7.3 Fog Middleware

The importance of middleware/abstraction in fog architectures has been discussed since the early days of fog computing. In [10], Bonomi proposed a fog abstraction layer to hide heterogeneity and manage resources at the fog layer. Nath *et al.* [29] also identifies the need for middleware at the fog layer to control the network and other resources on fog nodes. Moreover, Aazam *et al.* [124] view the entire fog layer as a middleware for Cloud Computing. A few proposals for fog middleware have been presented.

Middleware have been proposed for network-edge paradigms. Table 2.2 presents middleware proposed for fog and other network-edge paradigms. Most of the proposed middleware have been for Mobile(Multi-Access) Edge Computing (MEC). In paper [125], the authors developed middleware for managing computation at the network's edge for mobile users. Rodrigues *et al.* [126] also designed a middleware for mobile edge-clouds with the publish-subscribe approach. In another mobile edge middleware [109], Carrega *et al.* present Mobile Edge Computing as a type of fog computing. Their middleware solution focuses on the deployment of distributed applications by developers. Additionally, Orsini *et al.* in two works present and evaluate a self-adaptive middleware for task offloading in a mobile environment [127, 128]. Middleware for Mobile Edge Computing are popular because the end devices in MEC are mobile phones. Most of the middleware for Mobile Edge Computing have a distributed architecture with mobile phones acting as agents and other components stored on the cloud or edge processing devices. Mobile phones, although resource-constrained, have much more processing and storage resources than IoT sensors. Thus, these middleware are not well-suited for IoT applications.

Table 2.2: Middleware for Fog and Edge Computing

Paper	**Functionalities and Strengths**	**Applications**	**Limitations**
[114]	Distributed task execution on fog nodes	Seismography	No mobility support
[129, 130]	Service-Oriented approach Flexible to add additional services Monitors services with multi-agent	Smart City applications and Cyber-Physical Systems	No direct communication between fog nodes
[56]	Selection of best edge nodes for execution of tasks Supports mobility	Object detection applications	Requires prior knowledge of users movement path for mobility support.
[35]	Data filtering	Greenhouse IoT applications	Limited to a single application (Farm IoT)
[131]	IoT application life-cycle management	Generic	
[132]	Secure end-to-end communication for end devices, edge nodes and cloud	Generic	Focused only on security

[133]	Compliance with Organisation for Economic Cooperation and Development (OECD) privacy policy for Health IoT	Healthcare	Has a very specific function. Focused on privacy in healthcare
[53]	Task scheduling Data acquisition	Generic	Only the architecture is presented
[109]	Distributed application deployment	Generic	Developed for a Mobile Edge Computing (telecom networks)
[125]	Task offloading Management of application modules across devices in cloudlet	Augmented reality application	
[126]	Distributed processing Publish-subscribe service Distributed file storage	Content delivery application Emergency rescue application Distributed face recognition	Designed for mobile edge-clouds
[134]	Distributed processing on IoT devices with no cloud support Distribution of data among IoT devices Realtime data analysis	Generic	Not suitable for large scale deployment
[127, 128]	Task offloading	Generic	For Mobile Edge Computing

	Self-adaptive management of nodes Supports mobility		

The works in [135, 136] present fog middleware architectures. Paper [135] presents Distributed IoT-Fog Architecture for Application Management (DIFAAM) to manage the life cycle of applications as they are run within the fog or cloud. Its goal is to ensure that application requirements are met by processing nodes before assigning tasks to nodes. Pore *et al.* [136] propose another architecture for fog and edge middleware. Their focus is on task scheduling and data collection in mobile fog environments.

Studies by Nader *et al.* [129, 130, 137] present a Service-Oriented Middleware approach for fog computing. In [129], a Service-Oriented Middleware approach is adopted for a smart city. The middleware abstracts system resources as services made accessible to devices across all layers in the system. The authors implement their middleware for a cyber-physical system in [130]. The Service-Oriented approach makes it possible to add new services after deployment. Also, a Service Oriented Middleware provides flexibility for large-scale IoT applications [137].

Paper [114] presents a Fog-based middleware for distributed cooperative data processing at the fog layer. Their proposed middleware fog nodes have two modes: they either work together on a task or work independently. Like the other middleware proposed for fog computing, their system is implemented in a specific application - subsurface imaging and monitoring.

Shekhar *et al.* [138] use middleware for task offloading in a mobile IoT environment. Their middleware manages resources across all layers of the fog architecture, intending to ensure that service-level objectives are met even when edge devices are mobile. Their proposed solution, however, requires prior knowledge of the users movement, which is not practical in real-life scenarios. Other middleware have been proposed for privacy and security in fog systems [132, 133]

2.7.4 Research Gaps

Despite the widespread interest in fog computing, there are not, to the best of our knowledge, works that examine the role of middleware in fog resource management. Given that the challenges that fog computing presents, that is, heterogeneous nodes, unreliable actors and a dynamic environment, are addressed through middleware in other distributed systems [139], the use of middleware for resource management makes for an interesting area for research.

Some research studies propose fog middleware approaches that have not been evaluated; others propose solutions for specific use cases and applications. There is a dearth of studies that study all aspects of resource management in fog computing and propose a middleware to address them. Also, there is a need for studies that look at the middleware problem in fog computing holistically instead of targeting specific applications.

Research on resource management in fog computing has focused on specific tasks such as task scheduling, task offloading and application placement. Additionally, some approaches rely on cloud servers for some resource management tasks. Furthermore, although middleware has been used classically in distributed systems to manage the resource management challenges in fog, such as heterogeneity, the use of middleware in fog computing for resource management has not been fully explored. Middleware proposed for fog computing is either limited to specific applications or does not fully meet the requirements of fog architectures. For instance, although cooperation between fog nodes in the same fog domain has been presented in most architectures, few resource management approaches involve cooperative fog resource management. Moreover, interfaces between different layers of the Fog-IoT architectures have not been proposed. Furthermore, because of the dynamic nature of the fog environment, resource management has to be adaptive—adaptive approaches to system management can be achieved through middleware.

Reliability is also an essential aspect of fog applications. Aspects of reliability, like timeliness and availability, are necessary for delay-sensitive applications.

Notwithstanding, existing approaches to fog application management have not considered reliability. Particularly, it is unclear how the approach to resource management and other techniques employed in fog computing will affect the performance(reliability) of applications.

To maximise the use of available resources at the fog layer for the end-user devices and cloud, this thesis proposes and evaluates a layer of abstraction that hides the details of the fog layer from devices at the end-user. Middleware will integrate communication, node management, task scheduling, task offloading at the fog layer. It would also handle interactions between the fog layer and the cloud and IoT devices. It is posited that, because fog nodes have limited resources and vary in the resources they can provide, using a middleware between the fog and users will enable fog nodes to rely on other fog nodes within the system to provide services requested by users, without the users knowing which fog node completed the task. This approach enables transparent resource sharing among fog nodes, allowing the system to optimise resource utilisation without exposing infrastructure complexity to end users.

2.8 Summary

The goal of this chapter was to situate the work presented in the rest of this thesis in the context of research in fog computing. The chapter overviewed IoT, cloud computing, and fog computing. The chapter proceeded with a presentation of the nature of the fog environment, which led to a discussion on the need for resource management in fog computing. Current literature on fog resource management was then discussed. The second part of the paper reviewed related work on resource management, middleware and reliability in fog computing to highlight the gaps this thesis will investigate.

The growth of IoT and its attendant demands on cloud computing contributed to the emergence of several computing paradigms targeted at making computation resources available to users at the network edge. The evolution of these paradigms has culminated in the emergence of fog computinga paradigm that extends the cloud by using available resources between the cloud and users. The fogs heterogeneous and dynamic environment requires effective resource management. However, although middleware has been used for resource management in distributed systems, there is a dearth of research that addresses holistically the use of middleware in fog computing. This thesis seeks to make contributions in this regard.

The next chapter builds on this one by presenting the approach used to address the gaps identified here. The following chapters will then present the contributions made in these areas.

CHAPTER 3

Research Design

This chapter introduces the research design, discusses each component, and explains the rationale and justification for using the chosen approach over other options. The goal is to show how the aim of this thesis has been achieved. Thus, this chapter is key to understanding the chapters that follow, which implement the framework set out in this chapter.

3.1 Research Approach

This thesis investigates the use of middleware for efficient resource management at the fog layer in fog computing. It is hypothesised that, given the nature of the fog architecture and environment, middleware would play an important role in the reliability of fog computing, particularly for delay-sensitive applications. Much work has been done on the aspects of resource management at the fog layer, which would target improving specific areas. However, as the entire system relies on a combination of various schemes, it is argued that middleware at the fog layer, how it is implemented, and the resource management techniques and algorithms used will affect the reliability of applications.

The research design is broadly in two parts: the middleware design phase and the experimentation and evaluation phase. The first phase aimed to investigate

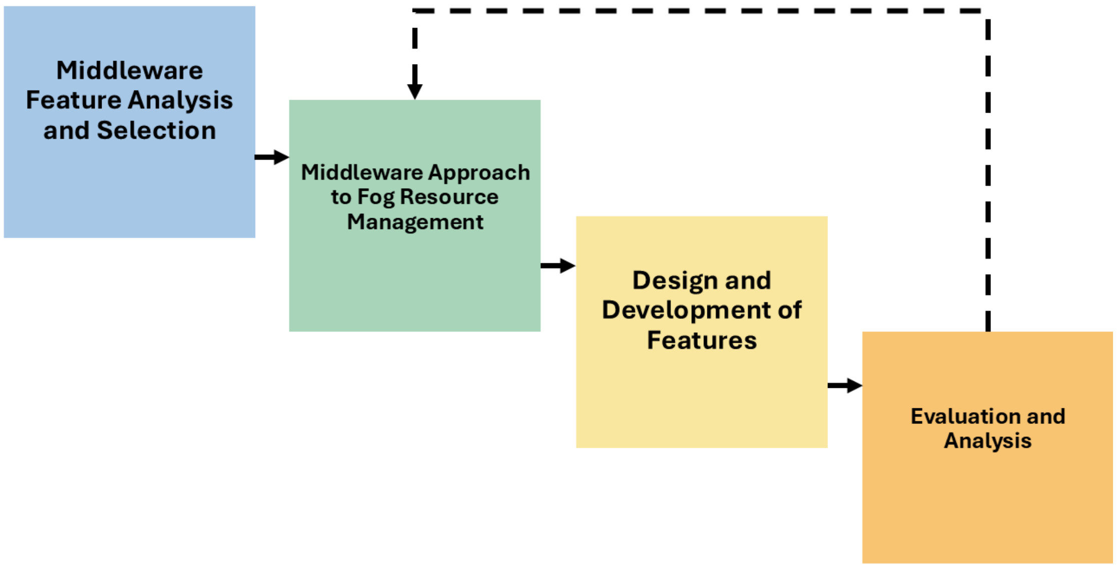

Middleware Feature Analysis and Selection
Middleware Approach to Fog Resource Management
Design and Development of Features
Evaluation and Analysis

Figure 3.1: The Research Design used in the thesis

the need for middleware in fog computing and select the features necessary for the middleware. The second phase involved the evaluation of the design decisions made through simulation and the final middleware design.

The research design encompasses four stages, as illustrated in Figure 3.1. An iterative approach was employed for middleware development, whereby evaluation results from each proposed solution informed subsequent design decisions and feature enhancements. This iterative methodology is evident in the progressive development of core features presented in Chapters 5, 6, and 7.

The simulation results presented in Chapter 5, which examined processing user requests within a single cluster, demonstrated that response times could be improved by utilising alternative nodes when the closest node is occupied with other requests. These findings directly informed the multi-cluster approach proposed in Chapter 6, which enables the utilisation of neighbouring clusters for request processing based on predefined conditions.

Subsequently, the evaluation of the multi-cluster approach revealed limitations in scalability, necessitating a more distributed solution. This insight led to the development of the peer-to-peer distributed approach presented in Chapter 7, which addresses the scalability constraints identified in the earlier centralised and multi-cluster architectures.

This iterative progressionfrom single cluster to multi-cluster to fully distributed approachesdemonstrates how empirical results systematically guided the evolution of the middleware design towards increasingly sophisticated and scalable solutions.

The following section discusses the methods which are being used to achieve the objectives outlined in section 1.3.

3.2 Feature Analysis and Selection

In the first stage of the research, a study was conducted to ascertain the properties and specifications of the proposed middleware. Several researchers have reviewed the resource management problem in fog computing [19, 140]. This study aimed to define the problem of resource management in fog computing clearly and outline the specifications required for proposed solutions. Thus, a review of the literature on fog resource management was conducted. Additionally, proposed approaches to resource management in fog, IoT and cloud computing have been studied to understand existing approaches. The goal of this study was to address the first question for the research study: *What features should a middleware for resource management in fog computing have?*. Literature was used from the following repositories: IEEE Xplore, ACM Library, Elsevier/Science Direct and Springer journals.

To answer the question, aspects of fog resource management are selected from the literature on fog resource management. Then, an analysis of the resource management problem in fog computing was conducted. Based on the analysis, the properties and features of the middleware were selected. This process is documented and presented in chapter 4. Other researchers have used the approach adopted. For example, in [141], the authors surveyed adaptive resource management techniques in cloud computing as part of a process to develop a brownout approach to cloud resource management. Since there are no fully deployed fog systems, there is a dearth of empirical data and findings on fog computing. Consequently, the approach used in this study relies on existing fog resource management proposals.

3.3 Middleware Approach to Fog Resource Management

A detailed framework for the middleware is developed based on the analysis of the findings in step one and the chosen features. The framework includes the fog middleware's architecture, model, and features. As the research was conducted using an experimental approach, the middleware design was revisited iteratively based on the outcomes of evaluating middleware features. The final middleware design presented in the next chapter is a culmination of the iterative process and is illustrated in the feedback loop in figure 3.1.

3.4 Design and Development of Middleware Features

The third aspect of the research involves the development of the properties and features of the middleware. For each component developed, the problem it will address was analysed and stated, followed by a system model, the proposed solution (the middleware feature), and an evaluation of the solution. The approach is illustrated in figure 3.2. It is used for the evaluation of the middleware features presented, which are detailed in chapters 5, 6 and 7.

1. **System Model and Environment:** defines the system model or environment that the middleware feature operates it.
2. **Problem Definition:** states the problem the middleware will be addressing in the system for the given environment or scenario.
3. **Proposed Solution:** the algorithm, technique, or framework developed to solve the problem.

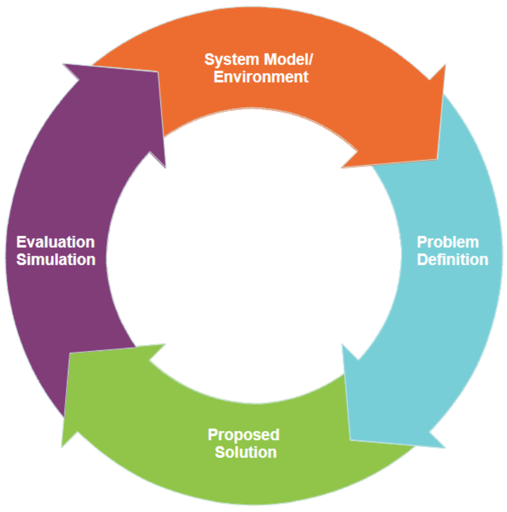


FIGURE 3.2: Middleware Feature Design and Evaluation Approach

4. **Evaluation:** The proposed systems in this thesis are evaluated via simulation experiments. The simulation tools used and the rationale for their selection are presented in the next section.

3.5 Evaluation Metrics and Results

The final stage of the research approach was the testing and evaluation of the middleware components. The evaluation was not a terminal stage but is part of the development of each module as discussed in the previous section. The goal of the evaluation was to improve the performance of modules based on relevant metrics. The objective of the evaluation is to study and improve the reliability of the application involved.

Reliability is defined by ISO [142] as the *ability of a system or component to perform its functions under stated conditions for a specified period of time.* The performance metrics used include response time(delay), utilisation, network usage and energy consumption

Response time

measures the delay from when data leaves its source until the response/output arrives at the processing point. It is the sum of the processing delays and propagation delays. The response time is the most common metric in fog and edge simulations [143]. Response time is measured in seconds (in milliseconds, microseconds, etc.). As discussed in earlier chapters, a key objective for fog computing is to minimise response time. The response time is expressed as:

$$R = \sum_{i=1}^{N} T_{p,i} + \sum_{j=1}^{M} T_{prop,j} \tag{3.1}$$

where R is the total response time; $T_{p,i}$ is the processing delay for the i-th device, including computation and queuing delays; $T_{prop,j}$ is the propagation delay for the j-th connection, representing the time taken for data to travel between nodes; N is the total number of processing devices; M is the total number of network connections.

The objective in fog computing is to minimise R.

Utilisation

Measures the resources consumed by devices within the system. Mathematically [144],

$$U_i = X_i S_i \tag{3.2}$$

where U_i is *utilisation* of device i, X_i is throughput (the rate per unit time at which requests can be serviced by the device) and S_i is service time(the time taken to service a request).

In the studies presented in this thesis, the utilisation of systems under different configurations are measured to compare how well resources are utilised.

Network Usage

The amount of data transmitted and processed within the system is measured as network usage, which allows us to measure the system's scalability. The system's network usage is measured in megabytes.

Energy Consumption

Energy consumption measures the total energy utilised by nodes in the system during their operation. This is critical for fog nodes because of the limited power supply available to them. The dynamic energy consumption of nodes is calculated using:

$$E = P_{\text{static}} \times T + \sum_{i=1}^{N} P_{\text{dynamic},i} \times T_i \tag{3.3}$$

where E is the total energy consumption (in joules), P_{static} is the static (idle) power consumption (in watts), $P_{\text{dynamic},i}$ is the dynamic power consumption for task i (in watts), T is the total time the system is operational (in seconds) and T_i is the time taken for task i (in seconds)

The selection of these evaluation metrics is guided by two fundamental considerations. First, the chosen metrics align with established evaluation frameworks in fog computing research, particularly in the domain of resource management. Metrics such as response time, network usage, and energy consumption represent the most widely adopted performance indicators for evaluating fog architectures [19, 97, 145, 146], ensuring that the results of this study can be meaningfully compared with existing research and contribute to the broader body of knowledge in the field.

Second, the inclusion of utilisation metrics directly addresses the core objective of this research: to evaluate the proposed middleware framework and its associated services for efficient resource management in fog computing. Utilisation metrics enable the assessment of how effectively the middleware optimises resource allocation across fog nodes, providing quantitative evidence of the framework's ability to maximise resource efficiency. These metrics are essential for determining the performance impact of different approaches on both application responsiveness and fog node efficiency.

Furthermore, this combination of performance and efficiency metrics provides a comprehensive evaluation framework that captures both user-centric outcomes (response time) and system-centric benefits (resource utilisation, energy consumption, network usage), enabling a holistic assessment of the middleware's effectiveness in achieving the research objectives.

3.6 Rationale for Evaluation Approach

Performance evaluation in computing is usually done by mathematical analysis, simulation or experimentation [147]. Ideally, a combination of these approaches is used. Analytical evaluation is done using mathematical equations. It is used to test simple models and for the generalisation of models. As this research follows an experimental approach through simulations, the discussion of mathematical analysis is set aside at this point. The two approaches–simulation and experimentation (test bed)–are examined, with a justification for choosing simulations for the experiments in this study.

A simulation uses a model of the system being studied to show how the system will behave given a set of inputs. For two reasons, simulation experiments are less reliable than experimentation on an actual system or test-bed. First, most simulators run in discrete time, whereas life is in continuous time. The simulation results are, therefore, not a true reflection of how the system will behave in the real world. Secondly, a highly controlled simulation environment cannot show unpredictable events that may occur in a live environment. By contrast, a physical experimentation tests the real-world implementation of a system or its prototype and produces more reliable results.

On the other hand, experimentations are often not practicable in specific fields compared to simulation. Building physical experiments takes much longer and often depends on other considerations not controlled by the researcher, such as equipment availability. Also, the cost involved is often higher than running simulation software on a single computer system. Furthermore, it is almost impossible to test for the scalability of a big system through experiments [148]. By contrast, simulation can predict a system's behaviour at scale. Thus, although not ideal, simulations are preferred for evaluating network systems and protocols [147]. The best approach is to use a combination of approaches [144]. Experiments show that the system works as expected, and simulations show it can work at scale.

For this thesis, carrying out both simulations and physical experimentation is impractical. First, as previously alluded to, there are no real implementations of fog computing to carry out actual tests. Even if there were, the researcher faced the classic challenge in computer network experimentation--running active experiments on a live network system [148]. This would have been impractical given the active nature of the techniques proposed in this work. Furthermore, the time and budget constraints imposed by the research program mean that such an approach would further delay the study.

Moreover, simulations have been the most common means of evaluation for fog computing [143, 149]. Also, the simulation tools developed for fog computing are based on well-established frameworks which have been used for cloud computing and have been widely accepted by various research communities.

Table 3.1: Some Fog Simulators

Simulator		**Year**	**Language(s) or tool(s) used**
Name	**Paper**		
FogTorch	[117]	2017	Java
iFogSim	[150]	2017	Java, CloudSim, JSON
MyiFogSim	[151]	2017	iFogSim
FogNetSim++	[152]	2018	OMNeT++
YAFS	[153]	2019	Python, JSON
MobFogSim	[154]	2020	iFogSim

3.6.1 Simulation Tools for Fog Computing

Most fog computing techniques have been evaluated using simulations [54, 119]. Few middleware for fog computing have been evaluated through experiments [35, 129], however, as these are made for specific use cases and scenarios, they do not have the challenges discussed above. The work in [114] presented a fog middleware which was evaluated using a mathematical model.

Several simulation tools are available for fog computing [155]. iFogsim [150] extends the CloudSim [156] simulator. It is one of the popular tools used for fog simulation. iFogSim is written in Java, and the source code is freely available on Github[1]. The simulators presented in [151, 154] are extensions of iFogSim. FogNetSim++ [152] is based on OMNeT++ which is written in C++. FogNetSim++ includes a fog broker node for managing other nodes in the fog system. In 2019, Lera *et al.* [153] developed *Yet Another Fog Simulator* of YAFS. Written in Python, YAFS uses graphs to represent topologies and data flow. Both iFogSim and YAFS allow users to import custom topologies and modules using JSON. Table 3.1 summarises information on some fog simulators. iFogSim and a developed simulation tool based on FogNetSim++ are used in this study. The developed environment is described in the next section. The choice of a tool depends on the system model for the study under consideration and how each tool supports the model. Details of each simulation tool used are presented with the studies they are used in subsequent chapters.

Rationale for Choosing OMNeT++

Unlike other simulation tools available for fog computing research, OMNeT++ offers several distinctive features that make it particularly well-suited for middleware evaluation in this study.

Modular Architecture and Component-Based Design: OMNeT++ employs a modular, component-based simulation architecture that aligns perfectly

[1]https://github.com/Cloudslab/iFogSim

with the middleware evaluation requirements of this research. This modular design enables the systematic analysis of specific middleware features and their individual effects on system performance, allowing for granular evaluation of different architectural components such as resource managers, load balancers, and communication protocols. This capability is essential for understanding how each middleware component contributes to overall system behaviour.

Comprehensive Network Protocol Support: The INET framework integrated within OMNeT++ provides a complete implementation of the Internet protocol stack, encompassing all network layers from physical to application. This comprehensive network modelling capability is particularly valuable for this research because it enables accurate simulation of the heterogeneous network environments typical in fog computing, where multiple communication protocols and network technologies coexist. The ability to model complete network architectures is crucial for evaluating the proposed middleware across different deployment scenarios including peer-to-peer, clustered fog, and cloud-based architectures.

Scalability and Performance Analysis: OMNeT++ supports large-scale network simulations with sophisticated statistical data collection and analysis capabilities. The integration of Python and Pandas for data analysis, as mentioned in the developed simulation environment, enables comprehensive performance evaluation and result interpretation. This scalability is essential for evaluating middleware performance across different network sizes and configurations.

Extensibility and Customisation: The C++ programming environment and Network Description Language (NED) provide extensive customisation capabilities, allowing for the implementation of specific middleware functionalities and communication protocols not available in other fog computing simulators. This flexibility was crucial for implementing the novel peer-to-peer and clustered architectures proposed in this research.

Established Research Foundation: Building upon FogNetSim++, which extends OMNeT++, provides a solid foundation that has been validated by the research community while allowing for the specific enhancements required for middleware evaluation.

Hardware Requirements

Processing Power: Multi-core processors (minimum 8 cores recommended) are essential for handling the parallel processing demands of large-scale network simulations, particularly when evaluating peer-to-peer architectures with multiple concurrent nodes.

Memory Requirements: A minimum of 16GB RAM is recommended, with 32GB preferred for large-scale simulations involving hundreds of fog nodes and thousands of IoT devices. The memory requirements scale with the network size and the complexity of the middleware operations being simulated.

Storage Capacity: Sufficient storage space (minimum 200GB) is required for simulation data, logs, and result files, particularly when conducting extensive parametric studies across multiple scenarios and configurations.

Operating System: The simulations were conducted on Linux-based systems (Ubuntu 20.04 LTS) to leverage the optimal performance and compatibility of OMNeT++ with Unix-based environments.

These hardware specifications ensure that the simulation experiments can be conducted efficiently while maintaining the reliability and reproducibility of results across different evaluation scenarios.

3.6.2 Simulation Environment

A simulation environment was developed in OMNeT++ to evaluate peer-to-peer, clustered fog, and cloud-based architectures. The implementation builds upon FogNetSim++, using the INET framework to support all layers of the network model. The simulation components and modules are programmed in C++, while network topologies and module designs are defined using the Network Description Language (NED). The simulation framework includes various features such as random number generators, topology discovery, routing mechanisms, queuing models, and statistical data collection. Additionally, the latest features in OMNeT++, such as data analysis using Python and Pandas, are integrated to enhance performance evaluation and result interpretation. A detailed description of the simulation tool and its components is provided here, with the full implementation available on GitHub [2].

System Model

End-user devices generate and transmit requests to fog nodes for processing. These users can be deployed as a collection of IoT devices in a sensor network connected to a fog computing infrastructure, where requests are sent through a common gateway. Alternatively, users may consist of mobile devices that dynamically change location within the network, transitioning between different fog environments. Both stationary and mobile users are supported, with communication facilitated over wired or wireless connections.

An MQTT-based application running on UDP is included in the simulation to enable lightweight publish-subscribe messaging. Additionally, the INET framework supports various other application-layer protocols, including HTTP, SMTP, CoAP, AMQP, and FTP, allowing further extension of the tool to evaluate different application scenarios and transport protocols in a fog computing environment.

Fog Nodes

Fog nodes serve as intermediate processing units that handle requests from end-users. Figure 3.3 shows the NED components of the fog nodes. Three primary classes of fog nodes are implemented:

[2]https://github.com/hsamwini/FogSimulator.git

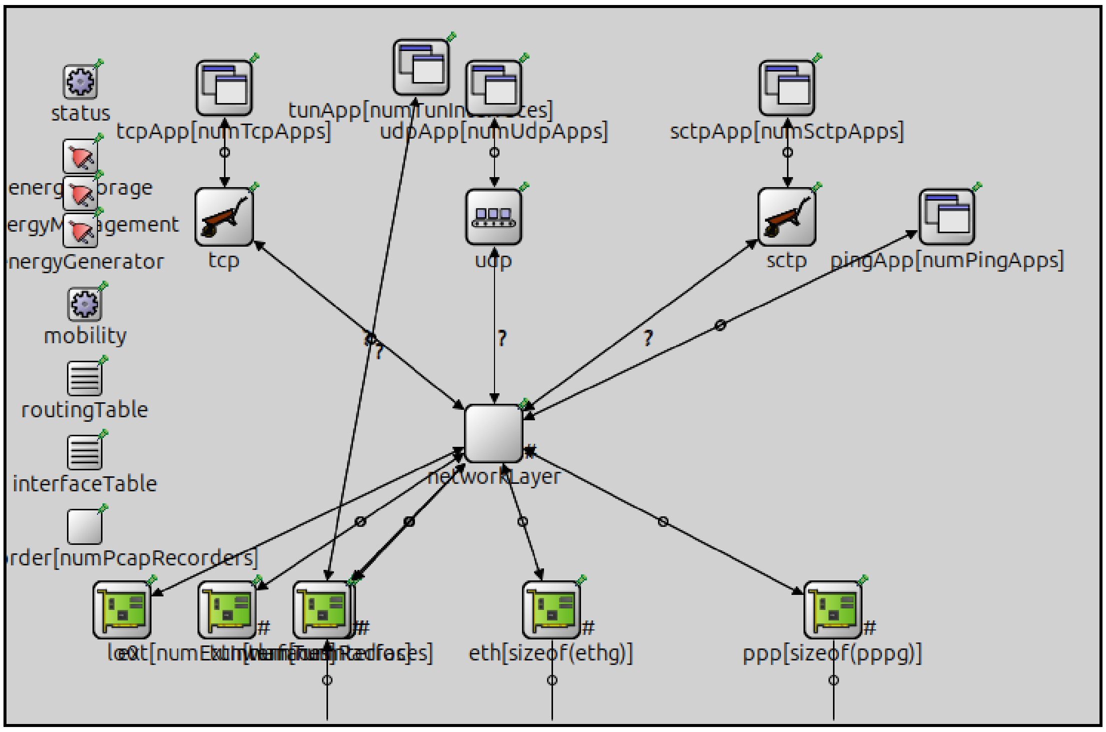
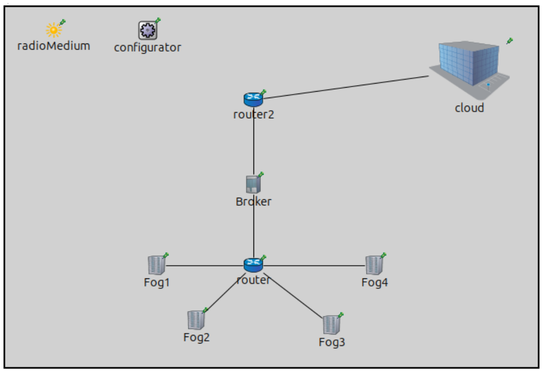


FIGURE 3.3: The Fog Node Module

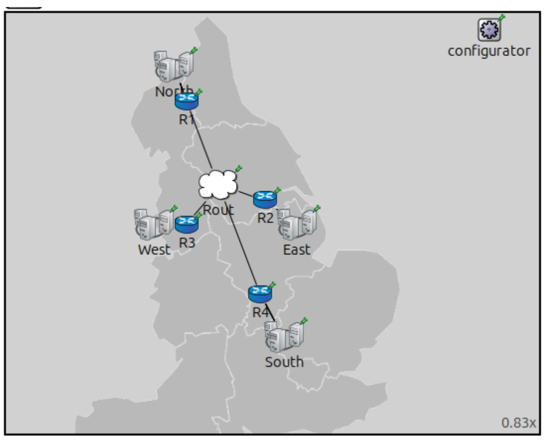
Peer-to-Peer Fog Nodes These nodes communicate directly with each other, either processing requests locally or forwarding them to another fog node. In the current implementation, peer assignments are static, with predefined connections between nodes. Future enhancements will introduce dynamic peer selection, allowing nodes to establish connections based on user-defined protocols and network conditions.

Clustered Fog Nodes These nodes operate as worker nodes, executing tasks assigned by a centralised broker or cluster controller. This architecture enables efficient resource utilisation by distributing workloads across multiple fog nodes.

Controller/Broker The broker receives user requests and manages task distribution among fog nodes based on predefined scheduling policies. In a centralised or clustered fog deployment, the broker also incorporates resource management and middleware components. Figure 3.4 illustrates an example of a clustered fog network configuration.



(A) An single cluster



(B) A Clustered Architecture

FIGURE 3.4: GUI view of the simulation (clustering example)

Cloud Data Centres

In addition to fog nodes, a cloud computing environment is simulated as a data centre consisting of multiple high-performance servers. These servers execute the same applications as fog nodes but provide significantly greater computational resources, measured in Millions of Instructions Per Second (MIPS). The simulation supports multiple cloud infrastructures deployed across different locations, allowing researchers to compare latency, resource utilisation, and service delivery between fog and cloud computing paradigms.

3.7 Summary

This chapter introduced the research design used in this study, detailing the approach taken to investigate middleware for resource management in fog computing. The research is structured into four key stages: feature analysis and selection, middleware framework development, design and implementation of middleware features, and evaluation. Each stage is designed to address specific objectives and ensure a systematic progression toward the final middleware design.

The chapter also outlined the key evaluation metrics used to assess the proposed middleware, including response time, resource utilisation, network usage, and energy consumption. These metrics were selected based on their relevance to the reliability and efficiency of fog computing environments. Additionally, the rationale for choosing a simulation-based evaluation approach over physical experimentation was discussed. It also emphasises the feasibility, scalability, and cost-effectiveness of simulations for evaluation in the absence of real-world fog computing deployments.

Furthermore, an overview of available fog computing simulators was provided; it highlights their capabilities and justifies the selection of iFogSim and Omnet++ for this study. The choice of these tools was driven by their ability to model fog environments accurately and support custom implementations required for this research.

The next chapter builds on the research design presented here by detailing the middleware framework, including its architecture, core components, and the specific resource management techniques implemented. The iterative nature of the middleware development process, as illustrated in the research design, ensures that the final solution is refined based on performance evaluations. Through this structured approach, the study aims to contribute a middleware solution for improving the reliability and efficiency of fog computing applications.

CHAPTER 4

Middleware for Fog Resource Management

This chapter proposes a middleware framework for fog computing based on a review of the state-of-the-art resource management in fog computing. The review forms the foundation for identifying the principles and features of the proposed middleware. Consequently, this chapter connects the existing research on fog resource management with the middleware framework, addressing the key research question: What role will middleware play in fog architecture? The proposed middleware framework provides a framework for efficient resource allocation, task scheduling, load balancing, and service orchestration in a heterogeneous fog computing environment.

4.1 Review of Fog Resource Management

Resource Management in a distributed system, as defined by [157], consists of three tasks:

1. Planning and organising the provision of resources, which includes:
    - Resource location
    - Resource availability

- Resource cost

2. Controlling the use of and access to resources, including allocation, optimisation, and authorisation.

3. Task management, ensuring resource reliability and failure detection.

The goal of resource management in distributed systems is to maintain optimal performance [158]. In their work on resource management in large distributed systems, Goscinski *et al.* [159] proposed a two-tier approach to resource management in distributed systems. A lower layer is for local resource export and allocation, and an upper layer is for non-local resources or services.

Consequently, a resource management system for fog computing should organise and control resources and manage tasks to ensure reliable service performance. In the fog environment, local resources refer to those on a fog node or a cluster of fog nodes under common control. Non-local resources or services may originate from other fog nodes, the cloud, IoT devices, or users. To develop a resource management system for fog computing, it is essential to identify and define the resources involved and the characteristics of the fog environment. Furthermore, the trade-offs associated with various aspects of resource management must be analysed to ascertain the optimal approach to middleware for fog computing.

Resource management at the fog layer is critical for two main reasons. Unlike the cloud, the fog nodes are resource-constrained. Most devices at the fog layer often serve a primary function beyond fog computing [62]. Also, fog nodes have smaller form factors and processors compared to the cloud [113]. Next, fog nodes are heterogeneous in various ways. They differ in architecture, resource availability, speed, etc. Therefore, the fog layer's processing, network and storage resources must be managed to achieve the desired performance levels. The CPU, memory, network, virtual machines, and energy resources at the fog layer must be managed to ensure that processing can be done by fog nodes without impacting the networking or other functions of fog nodes and without violating Service Level Agreements (SLA) for applications. Furthermore, several fog nodes in a domain may share resources at the fog layer for efficient execution of tasks [26, 98, 114]. In such a scenario, the resources within the fog domain must be managed for efficient utilisation. Based on these factors, resource management has been a focus of the fog research community.

The authors in [113] present a review of resource management in fog and edge computing. Their review groups papers on resource management in architectures, infrastructure, and algorithms. Middleware is classed under resource management infrastructure in this work. The authors note that resource discovery at the edge is not easy, and most edge researchers in edge/fog resource management assume resource discovery as a basis for their work.

Martinez's survey [160] is based on four stages of fog implementation. The four stages are design and dimensioning, selecting resource allocation methods, building a framework for resource management, and evaluation. The resource management aspect of this work focuses on resource provisioning and IoT resource allocation; other aspects of resource management are not considered.

Kaur *et al.* [161] review load balancing approaches in fog computing. Load balancing techniques are classified into static or dynamic approaches.

TABLE 4.1: Resource Management in Fog Computing

Paper	TO	AP	RS	LB	RA	RD
Mouradian et al. [19]	✓	×	✓	✓	×	×
Yousefpour et al. [59]	✓	✓	✓	✓	×	✓
Hong and Varghese [113]	×	✓	×	✓	×	✓
GhobaeiArani et al. [18]	✓	✓	✓	✓	✓	×
Bendechache et al. [162]	×	×	×	✓	✓	✓
Mijuskovic et al. [163]	✓	✓	✓	✓	✓	✓
Kaur and Aron [161]	×	×	×	✓	×	×
Martinez et al. [160]	×	×	×	×	✓	×
Fahimullah et al. [164]	✓	✓	✓	✓	✓	×
Kansal et al. [165]	✓	✓	✓	✓	✓	✓
Alsadie [166]	✓	✓	✓	✓	✓	×

TO: Task Offloading; AP: Application Placement; RS: Resource Scheduling; LB: Load Balancing; RA: Resource Allocation; RD: Resource Discovery

In an early work on resource provisioning in fog computing, Agarwal *et al.* [115] use a fog server manager to allocate resources on fog nodes to service requests from clients. In their architecture, requests that a fog node cannot service are forwarded to the cloud, and there is no collaboration between fog nodes. Kimovski et al. [116] proposed an architectural approach modelled after the human brain to achieve adaptive resource management at the fog layer.

Task offloading across different layers in the fog architecture has been investigated by researchers as a resource management approach. Papers [117, 118] adopt greedy heuristic approaches to offloading tasks. Mahmud *et al.* [119] tackle the problem from a Quality of Experience perspective. They use fuzzy logic to prioritise application placement requests. In [62], ENORM, a framework for dynamically managing resources on edge nodes, is presented. Resource Management on ENORM, however, involves the cloud and is therefore not ideal for delay-sensitive applications. Also, ENORM focuses on resource management on a single fog node; it does not do so for multiple nodes cooperatively.

In paper [98], Zhang *et al.* adopt a hierarchical approach to resource management in a cooperative fog computing system for intelligent transportation systems. Resources are managed by a coordinator fog node, which collects relevant information from other nodes within a domain. However, reliance on a coordinator server is a single point of failure for the system.

Table 4.1 summarises the tasks of resource management that researchers have focused on based on reviews of resource management in fog computing.

4.1.1 Resource Management Tasks

Based on the analysis above, resource management in fog computing could be defined as the organisation and control of heterogeneous fog devices to deliver cloud services to end users in a dynamic, scalable, and robust manner to meet user requirements. Table 4.2 summarises the key resource management tasks in fog computing based on this review.

Task	Description	Common Approaches
Task Offloading	Offloading computations from resource-constrained fog nodes to nearby fog nodes or the cloud.	Single-type: fog nodes offload tasks to a single fog node; multiple-type approach: tasks are sent to multiple nodes as sub-tasks for parallel processing [167]
Load Balancing	Distributing tasks across fog nodes to prevent bottlenecks and optimise performance.	Static, dynamic and adaptive approaches [165]
Resource Allocation and Provisioning	Assigning tasks to fog nodes while ensuring QoS compliance.	Heuristic, approximation and Reinforcement-Learning Algorithms [146]
Application Placement	Deciding where services should be deployed within a fog network.	Centralised, decentralised and hierarchical [146]
Resource and Task Scheduling	Scheduling tasks on fog nodes to optimise execution order and efficiency.	Static, Dynamic, hybrid and artificial intelligence approaches to task scheduling for fog computing [168, 169]
Resource Discovery	Identifying available resources dynamically in a large-scale fog network.	Peer-to-peer, Wi-Fi, MQTT, DNS-based discovery [170]

Table 4.2: Summary of Resource Management Tasks in Fog Computing.

Task Offloading

Task Offloading refers to approaches to free up resources on resource-constrained fog nodes by sending workloads to other fog nodes with more resources or to the cloud to process them. It involves direct interaction between fog nodes. Two approaches have been used for task offloading in fog computing [167]; single- and multiple-type approaches. In the single-type approach, fog nodes offload tasks to a single fog node, while in the multiple-type approach, tasks are sent to multiple nodes as sub-tasks for parallel processing.

Load Balancing

Load balancing seeks to evenly distribute tasks among nodes to avoid over-utilisation or under-utilisation [171]. Load balancing aims to reduce response time and increase throughput [172]. Approaches have been classified into static, dynamic and adaptive approaches [165].

Resource Allocation and Provisioning

Resource Allocation and Provisioning refers to provisioning resources for the users request. It involves allocating user tasks to a set of fog nodes to achieve the tasks' QoS requirements. Two aspects of the problem are separated into resource allocation and provision in some proposals such as [146]. The resource allocation problem involves allocating fog nodes to services, which is a double-matching problem involving fog and cloud on the one hand and fog and users on the other. Resource provisioning is the problem of making fog resources available for services. It focusses on only individual fog nodes or groups of nodes. Heuristic, approximation and Reinforcement-Learning Algorithms have been proposed for resource allocation in fog computing. In this review the two aspects are considered as one as the distinction is not relevant for the goal of determining middleware features.

Application Placement

Application Placement refers to the placement of applications or services on fog nodes. Some researchers have also referred to it as orchestration [173]. It involves interaction between the cloud and fog devices. Application placement techniques have been categorised into centralised, decentralised and hierarchical [146].

Resource and Task Scheduling

Resource and task scheduling has been used to refer to a range of tasks in fog computing. It refers to scheduling tasks on a fog node to meet users' QoS requirements. In some contexts, resource scheduling also refers to the scheduling of resources for user requests to achieve optimal utilisation of fog nodes [164]. There have been static, dynamic, hybrid and artificial intelligence approaches to task scheduling for fog computing [168, 169].

Resource Discovery

Resource Discovery involves finding available resources in the fog environment. These include cloud services/applications, IoT devices/users, and fog nodes. Unlike resource discovery in web services and IoT, which adopt approaches such as broadcast and multicast, in fog computing, such approaches cannot be adopted because of the large-scale nature of the resources involved [174]. Proposals on resource discovery in fog computing focus on the communication aspects. Approaches have involved exploring Wi-Fi, MQTT, DNS, BGP and Peer-to-peer protocols [170]. Resource discovery

Fog computing extends cloud capabilities to the network edge, providing low-latency processing for IoT applications. However, fog nodes differ from cloud data centres in two key ways:

- Resource Constraints: Most fog nodes are embedded in devices with a primary function other than computation (e.g., routers, gateways, vehicles) [62].
- Heterogeneity: Fog nodes vary in CPU capacity, storage, and network bandwidth, making uniform management complex [113].

A fog resource management system must account for these differences while organising, controlling, and coordinating resources. Specifically, it must:

- Manage local resources on individual fog nodes or within a fog cluster.
- Integrate non-local resources, including cloud services, IoT devices, and neighbouring fog nodes.
- Optimise resource usage while ensuring low latency, energy efficiency, and SLA compliance.

4.2 Middleware Framework Requirements

In the fog architecture, middleware will offer services to devices across all layers the cloud, fog and users. Middleware for the discussed resource management environment described above requires dynamic membership, asynchronous communication, a common message ontology, and supporting heterogeneity and mobility. To support these resource management tasks, fog middleware must be designed with the following key principles:

4.2.1 Transparency

Transparency refers to the ability of a system to hide details of its internal workings from users and present a consistent experience to users regardless of changes or modifications within the system. Transparency is important in the IoT-cloud context because of the limited computational resources available to IoT devices. Transparency ensures that IoT devices maintain a limited view of the system and do not require storage and processing resources to keep track of services. Middleware should abstract underlying complexity from applications and users. Transparency can be categorised into:

- Location Transparency Users should not need to know where processing occurs.
- Access Transparency Different fog nodes and cloud services should be accessible uniformly.

- Failure Transparency The system should recover from failures without affecting application performance.

4.2.2 Adaptability

The dynamic nature of fog environments requires middleware that can adapt to changes in resource availability, workload, and network conditions. The fog environment is dynamic, with unreliable and heterogeneous devices. As a result, the architecture of the fog layer, the availability of fog nodes and the behaviour of devices would change over time. Unlike a cloud data centre, which is often under a single entity's control, ensuring predictable behaviour, the fog layer comprises several devices from different sources. Given the collective states of available fog nodes, a given architecture may be preferred over others for a given context. For example, a decentralised clustered architecture may be preferred over a peer-to-peer architecture in a vehicular fog environment where mobile vehicles provide onboard computing and storage resources as fog nodes. Middleware should be able to manage changes in the fog environment and adapt the management of resources to suit the changes.

The fog environment is dynamic; nodes can appear, disappear, or change roles. Middleware must adapt by:

- Configuring itself to include modules and components that are suitable for the fog environment.
- Reconfiguring resource allocation based on current availability.
- Supporting mobility-aware resource management, especially in vehicular fog environments.

4.2.3 Interoperability

Fog nodes belong to different administrative domains, running diverse platforms. Middleware must ensure seamless cross-platform communication using standard protocols (MQTT, CoAP, etc.). Middleware should enable communication and resource sharing among heterogeneous devices, platforms, and protocols. In a cloud data centre, all devices are controlled by a single entity; therefore, communications among components may be planned and controlled. In contrast, devices at the fog layer may belong to different entities with different platforms and systems. To ensure effective collaboration for resource sharing, fog nodes must exchange messages and process requests from each other. Interoperability is the ability of systems running different platforms to work together.

Middleware at the fog layer should make communication and resource sharing among fog nodes from different entities possible.

4.2.4 Context-awareness

Middleware will allow fog nodes to determine the nature of the context in which they have been deployed. The context in which a fog node operates comprises the load from the end-users, neighbouring fog nodes, and the requirements of the cloud(s) available to the fog node. Middleware will manage communication with the different layers, horizontal context from interaction with other fog nodes, and vertical context with end-users and the cloud.

Based on a given context the fog device can adapt its features to ensure it fits into the given context and services IoT requests well.

Middleware should sense environmental changes and adjust resource allocation accordingly. This includes:

- User demand fluctuations (e.g., traffic surges).
- Network conditions (e.g., latency, bandwidth constraints).

- Fog node status (e.g., CPU load, energy levels).

4.3 Middleware Features

This work categorises middleware features along three classification dimensions based on the approaches used to develop the middleware framework for fog resource management: resources, architectural approaches, and management.

Resources

The middleware must effectively manage and coordinate different types of resources within a fog computing environment. These resources can be grouped into three main categories as presented in table 4.3.

Resource Category	Key Contributions
Fog Nodes	- Compute resources - Storage capacity - Network connectivity
Cloud Resources	- Services - Compute power - Storage - Network infrastructure
End-users / IoT Devices	- Interact with the system - Consume services

TABLE 4.3: Categories of Resources and Their Contributions

Architectural Approaches

Middleware solutions can adopt different architectural models for resource management, including centralised, decentralised, and hybrid approaches:

- **Centralised approach**: A single resource manager maintains a resource database and allocates resources based on predefined policies, rules, and user requirements. While this ensures global visibility and policy enforcement, it introduces a traffic bottleneck and a single point of failure.

- **Decentralised approach**: Resource management responsibilities are distributed among multiple nodes or across all participating nodes. This avoids central bottlenecks, enhances scalability, and increases fault tolerance.
- **Hybrid approach**: A combination of centralised and decentralised models, often structured in hierarchical layers, balances global oversight with local autonomy, improving efficiency and resilience.

Management

The management dimension focuses on the actual mechanisms used for resource management within the middleware. These mechanisms include:

- Algorithms for task scheduling, load balancing, and resource allocation.
- Protocols that define communication and coordination strategies among fog nodes.
- Infrastructure components that provide the underlying execution environment for fog applications.

By integrating these dimensions—resource categorisation, architectural models, and management mechanisms—the proposed middleware framework ensures efficient, scalable, and adaptive resource management in fog computing environments.

Dynamic Adaptability

The middleware framework integrates dynamic adaptability into its design:

1. **Lightweight Mode**: Minimal resource consumption, used when fog nodes only process requests. Here, the fog node is a worker node for a central

controller(cluster controller or broker) which assigns tasks to it. Middleware manages the communication with the controller.

2. **Cluster Controller Mode**: Activated when a node manages a fog cluster. The cluster controller manages a group of fog nodes to which it sends processing tasks based on resource management policies and application requirements.

3. **Peer-to-Peer Mode**: Enabled for distributed task sharing. Figure 4.1 shows the structure of fog nodes with the middleware in the peer-to-peer/decentralised scenario.

These modes ensure that middleware overhead remains minimal to maximise resource efficiency.

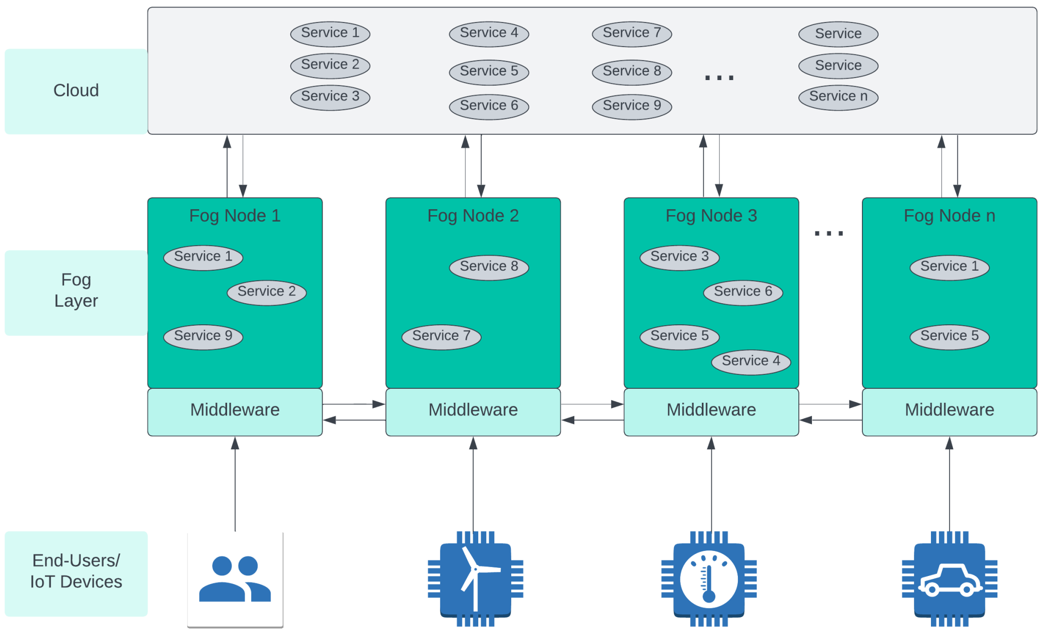


FIGURE 4.1: Decentralised (Peer-to-Peer) Fog Model with Middleware

4.4 Middleware Architecture

Based on an analysis of the requirements and features discussed above, a three-layered architecture is proposed for the fog middleware framework. Each layer is adaptive and includes features which may be active or disabled depending on the context in which the middleware is deployed. The adaptive design also frees up resources on fog nodes for other tasks based on availability and the architectural environment. A layered approach to adaptive frameworks has been used in other middleware in literature [141, 175, 176].

The middleware adopts a three-layered adaptive architecture:

1. Communication Layer: Manages all interactions with fog nodes, IoT devices, and the cloud.
2. Management Layer: This layer provides core resource management capabilities, including task offloading, scheduling, and load balancing.
3. Services Layer: Tracks and records service locations on the node and other associated fog nodes. Checks resource availability on the node to host services.

This adaptive design allows middleware functionalities to be activated or disabled based on the deployment context to ensure scalability and efficient resource utilisation. Figure 4.2 shows the middleware's architecture with all components.

4.4.1 Communication Layer

Manages interactions between different system components:

- **Cloud Interface** Exchanges orchestration messages with the cloud.
- **End-User Interface** Receives IoT requests and sends responses.
- **Fog Interface** Enables peer-to-peer or hierarchical communication between fog nodes.
- **Inter-cluster Interface** Facilitates cross-cluster communication in clustered architectures.

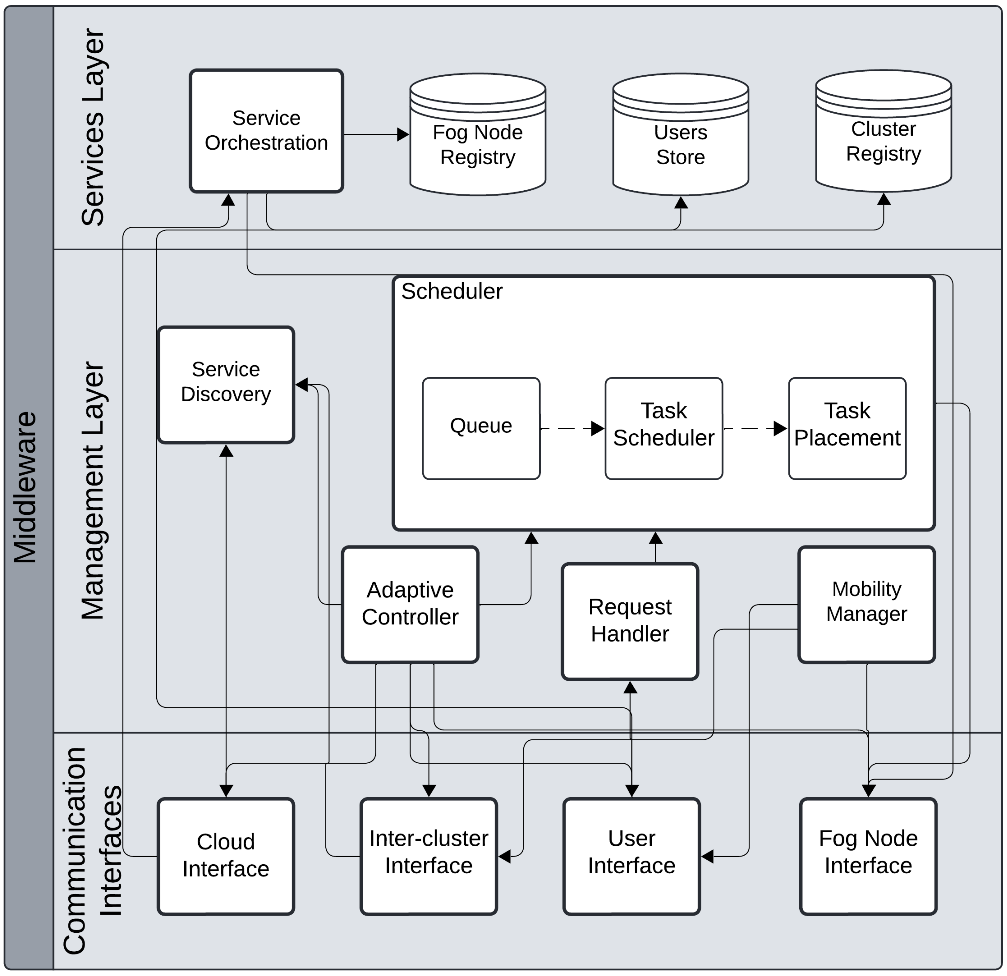


FIGURE 4.2: Proposed Fog Middleware

The first layer of the architecture is the communication layer which has components for managing interaction with other nodes in the system. It comprises a cloud interface to communicate with the cloud, an end-user interface, a fog interface for communication with other fog nodes, and a cluster interface for communication with other clusters. The cluster interface is used in a clustered fog architecture and would be inactive in a different architecture. The fog communication interface is used to exchange messages with other fog nodes in a peer-to-peer architecture and to exchange messages with a fog controller in a clustered architecture. Communication modules send and receive messages using MQTT and/or CoAP.

The lightest version of the middleware has only the user and cloud interfaces for scenarios where fog devices only forward requests from users to the cloud, which is case 4 in the open fog architecture (Figure 2.5) [63].

Cloud Interface

One feature of fog computing distinguishing it from edge-ward computing paradigms such as Edge Computing is that Fog computing does not replace the cloud but rather complements the cloud. Fog nodes only extend the capabilities of the cloud to the edge of the network and, therefore, depend on the cloud. Consequently, fog nodes maintain constant communication with the cloud. The cloud interface is, therefore, meant to send and receive messages from the cloud. Messages are exchanged between the cloud and the fog devices to orchestrate services, verify users, and send application data for further processing. The interaction with the cloud depends on the role the fog node is playing.

Inter-cluster Interface

In a cluster-based architecture, middleware running on fog controllers/brokers exchange messages with controllers from other clusters. The inter-cluster communication interface sends and receives the messages. Controllers update other clusters on their resource availability and forward requests to other clusters for processing. Each fog controller maintains a record of its neighbours. In other architectures or scenarios, such as peer-to-peer or hierarchical fog structure, the inter-cluster interface is disabled.

End-User Interface

The user interface sends and receives messages from the end user. IoT sensors and other end-user devices send requests for processing and receive responses. Application data may be stream data, simple requests, telemetry data etc. The user interface would be part of the middleware for most architectures or scenarios. Users exchange messages with middleware with protocols such as Message Query Telemetry (MQTT) and CoAP.

Fog Interface

In the peer-to-peer and hierarchical architectures, fog nodes exchange messages directly with one another. The fog node interface thus manages the message exchange between devices in these architectures. Fog nodes send updates on their resource availability. In a decentralised clustered architecture, the interface is used by cluster members to exchange messages with their cluster controller or broker.

4.4.2 Management Layer

The second layer manages the resources under the middlewares control. This layer includes the scheduler, which contains task queues, a task scheduler, and task placement modules, as well as a request handling module for processing requests from IoT devices. Additionally, it features a service orchestration module for managing services on the host node and other nodes (fog controller), and a mobility management module. Each module within this layer, or the entire layer, may be active or disabled in a specific context. When the fog node functions as an ordinary node in a cluster, the entire layer would be inactive. Conversely, as a fog controller in a clustered architecture, most modules would be necessary.

Request Handler

This module receives end-user requests, decides where they should be processed and forwards the request. Algorithm 1 shows the request handling algorithm. Requests from IoT devices are first received at the request handler. The request handler looks up the service requested by the IoT device in the service registry. If the service is hosted on a fog node within the cluster and the node has the resources to run it, the request assigned to the node. Otherwise the service will be looked up on the registry of other clusters and sent to an appropriate cluster. Where the service cannot be located, the request is forwarded to the cloud. Algorithm

1 and the cluster controller mechanisms of the middleware are fully evaluated in chapters 5 and 6.

Algorithm 1 Request Assignment Algorithm

```
Require: IoT Request
Ensure: Assign request to a suitable fog node
 1: $TaskMips \leftarrow Request.RequiredMips$
 2: $TaskCost \leftarrow Request.RequiredCost$
 3: while $Request$ NOT assigned do
 4:     if nodes available in cluster then
 5:         for each available node do
 6:             if $node.AvailableMips > TaskMips$ then
 7:                 if $node.Cost < TaskCost$ then
 8:                     assign $Request$ to node
 9:                     send $Request$ to node
10:                     break
11:                 end if
12:             end if
13:         end for
14:     else if NO nodes available in cluster then
15:         for each known cluster do
16:             if $cluster.AvailableMips > TaskMips$ then
17:                 if $cluster.MinimumCost < TaskCost$ then
18:                     Assign $Request$ to cluster
19:                     send $Request$ to cluster
20:                     break
21:                 end if
22:             end if
23:         end for
24:     else
25:         send $Request$ to the cloud
26:         break
27:     end if
28: end while
```

Scheduler

Handles task execution priorities and optimises resource utilisation. The task scheduler prioritises tasks that are processed on the fog node. Tasks are prioritised based on QoS requirements and/or Service Level Agreements. The scheduler keeps track of resources available on the node and allocates tasks to them. Also, the module implements various scheduling schemes which are activated based on context.

Service Discovery

Manages service instances dynamically. It searches for new services when they are requested by IoT devices. New services are discovered through other fog nodes, a fog controller, or the cloud. When a fog node receives high volumes of requests for a service it does not host, it may request to host that service.

Adaptive Controller

Adaptive controller configures the middleware based on the available fog node's environments. The module follows the MAPE-K [177] pattern for self-adaptation as illustrated in figure 4.3. Algorithm 2 is executed during the initial setup phase, where the controller identifies and selects the most suitable fog environment by evaluating connection delays and network structures. Depending on whether the environment operates in a peer-to-peer (P2P) or clustered manner, the middleware adapts by activating only the necessary components. For instance, in a P2P environment, neighbour connections are established, while in a clustered environment, a controller node is contacted for integration. If no suitable fog environment is found, the system defaults to cloud connectivity. This adaptive configuration enhances resource efficiency and optimises fog computing performance. The P2P model is fully evaluated in 7.

Resource Discovery Module

Implements lightweight protocols (MQTT, CoAP) for locating and registering available resources.

If the request cannot be processed on the fog node, it searches for another fog node to send the request to. First, the handler will look up the service in the Service

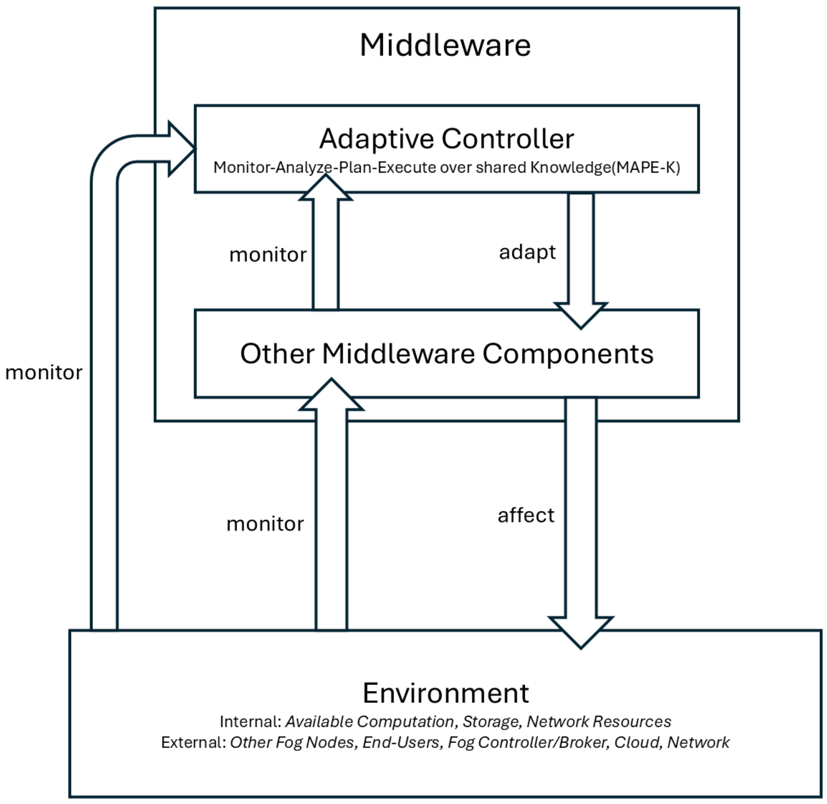

Middleware
Adaptive Controller
Monitor-Analyze-Plan-Execute over shared Knowledge(MAPE-K)
monitor
adapt
Other Middleware Components
monitor
monitor
affect
Environment
Internal: *Available Computation, Storage, Network Resources*
External: *Other Fog Nodes, End-Users, Fog Controller/Broker, Cloud, Network*

FIGURE 4.3: Middleware self-adaptive system (based on [177, 178])

Registry, if the service is hosted on a known node (a node listed on the registry), the request is forwarded to that node.

Communication Interfaces

Four communication interfaces handle communication between the middleware and other devices.

- The user/IoT device interface has components and protocols for exchanging messages with user devices. Users send requests in messages to the middleware, and the request handler processes them.
- The cloud interface sends and receives messages from the cloud for service orchestration and forwarding data/requests for processing.
- The Fog Interface communicates with other fog nodes. In the case of a clustered architecture, the fog interface forwards user requests to appropriate

Algorithm 2 Initial Configuration Algorithm

```
1: Input: $x \in \mathbb{R}^+$ (response wait time)
2: Output: Optimal fog node selection
3:
4: Send multicast message to all reachable fog nodes
5: Wait $x$ time for responses
6: $FogNeighbours \leftarrow []$ {List of fog node details}
7: for each response $r \in Responses$ do
8:     $delay \leftarrow r.connectionDelay \in \mathbb{R}^+$
9:     $envType \leftarrow r.environmentType \in \{P2P, Cluster, Null\}$
10:    if $envType = P2P$ then
11:        $neighbours \leftarrow r.getNeighbours() \in List$
12:        $neighbourDelays \leftarrow r.getNeighbourDelays() \in \mathbb{R}^+$
13:    else if $envType = Cluster$ then
14:        $controller \leftarrow r.getController() \in Node$
15:    end if
16:    Append $(r, delay, envType, neighbours, controller)$ to $FogNeighbours$
17: end for
18:
19: $bestNode \leftarrow selectBest(FogNeighbours)$
20: if $bestNode.envType = P2P$ then
21:     Send connection request to 2 neighbour peers to join as neighbour
22: else if $bestNode.envType = Cluster$ then
23:     Request controller to join cluster
24: else
25:     Request connection to the cloud
26: end if
```

fog nodes for processing and receives update messages from the fog nodes. In a peer-to-peer architecture, the interface exchanges messages with peers.

- The cluster interface exchanges messages with middleware hosted on other cluster controllers. The cluster interface is not present in the peer-to-peer case. The interface is for receiving neighbour cluster updates and forwarding requests to neighbour clusters when there are no available resources in the middlewares cluster.

4.4.3 Services Layer

The services layer stores records of services, nodes, clouds and clusters within the system. The management and communication layers use this layer to update and

Algorithm 3 Service Search Algorithm

```
Require: IoT Request
Ensure: Match Request to Service
 1: Retrieve requested service
 2: if service is hosted on the fog node then
 3:     if node can meet the request then
 4:         Assign task to this node
 5:     end if
 6: else if Service NOT hosted on fog node then
 7:     Lookup Service in Service Registry
 8:     if service is in Service Registry AND Host Node is free then
 9:         Send request to Host Node
10:     else
11:         Send Service Lookup Request to Domain Controller
12:     end if
13: end if
```

retrieve records of other nodes' and services' status within the system. The layer comprises of a service registry for storing a record of available services and where they are located, a fog node registry for storing a record of available fog nodes and a cluster registry for records of neighbouring clusters in a clustered architecture.

Maintains records of available services, fog nodes, and clusters:

- **Service Registry**: Stores available services and their locations. The middleware maintains records of services and fog nodes which are hosting them. The records are updated regularly as services are removed and added by fog nodes according to demand and availability.

  The performance of the Service Search(Algorithm 3) depends on the implementation of the service lookup in the registry, and can range from $O(1)$ to $O(S)$. If the registry uses an efficient data structure like a hash table, the lookup is $O(1)$, making the entire algorithm $O(1)$. If the registry is a list, the lookup could be $O(S)$, leading to $O(S)$ worst-case complexity. Thus, the overall time complexity depends on the service lookup method:

  – Best case (hash-based lookup): $O(1)$

  – Worst case (linear search in registry): $O(S)$

- **Fog Node Registry** Tracks resource availability across fog nodes.
- **Cluster Registry** Manages inter-cluster collaboration.

4.5 Summary

This chapter has presented a comprehensive overview of resource management in fog computing and proposed a middleware framework to address the challenges of managing resources in the fog environment. Through a detailed review of the state-of-the-art in fog computing, several key principles and features of resource management were identified, such as task offloading, application placement, resource scheduling, load balancing, resource allocation, and resource discovery.

The chapter emphasised the critical need for middleware in fog environments, which must handle heterogeneous, resource-constrained fog nodes, as well as non-local resources from the cloud and IoT devices. It highlights the importance of managing resources effectively to ensure expected performance, especially given the varying constraints and heterogeneous nature of the fog environment.

In conclusion, the proposed middleware framework aims to bridge the gap between current resource management solutions and the specific needs of fog computing to ensure efficient resource usage, improved service delivery, and support for diverse applications in the fog environment.

CHAPTER 5

Transparent Task Processing with Middleware in Fog Computing

This chapter evaluates the middleware approach to transparent request processing in fog computing environments. The proposed middleware provides location transparency by dynamically tracking available fog nodes and directing service requests accordingly. It also enhances failure transparency by reallocating services in the event of node failure or unavailability, ensuring uninterrupted service availability.

Abstraction is one of the key features contributing to cloud computings success[179]. The ability to pool resources together within and across data centres makes it possible for the user to experience their expected quality of experience without any knowledge of disruptions or failures to nodes in the system. This kind of transparency is possible in the cloud because data centres are often owned and managed by the same cloud service provider. However, in a fog computing environment, individual fog nodes may belong to a third party, such as a local ISP or mobile carrier. They could be leased out to run a service on behalf of the cloud to improve the Quality of Service for users. Two challenges arise from this. Firstly, the limited availability of resources on fog nodes means there must be alternative arrangements in the event of failure. The alternative cannot be at the cloud layer since this would defeat the purpose of processing close to the user. Secondly, since the location of services may change at any time, users must locate the service they require when they need it without resorting to the cloud—especially for emergency or delay-sensitive applications.

Although middleware has been used in distributed systems to provide abstraction and transparency, its use in fog computing has not been fully investigated. The findings here contribute to the broader field of edge computing by providing a framework for improving service availability and resource utilisation in dynamic environments.

The middleware framework receives user requests at the fog layer, processes them based on predefined criteria, such as Quality of Service (QoS) requirements and service availability, and directs them to the optimal fog node. The chapter details the middlewares service discovery and request handling mechanisms, evaluates its performance using simulation experiments with remote EEG monitoring as a case study, and compares it with traditional fog and cloud computing paradigms. The chapter is organised as follows: the next section provides a brief description of the problem, then the proposed solution is presented, the third section details the remote EEG monitoring case study, section four presents and discusses the results and the final chapter is the conclusion.

5.1 Problem Description

Modern IoT applications require timely and efficient access to services hosted across heterogeneous fog environments. However, these services may not always be available within the local fog domain to which an IoT device is connected. When this happens, current solutions either rely heavily on cloud access—resulting in increased latency and network load—or lack the interoperability needed for efficient fog-to-fog service access. The core issue addressed in this study is how to enable low-latency, cross-domain fog service access for resource-constrained IoT devices without depending on the cloud, while ensuring interoperability and scalability across diverse fog domains.

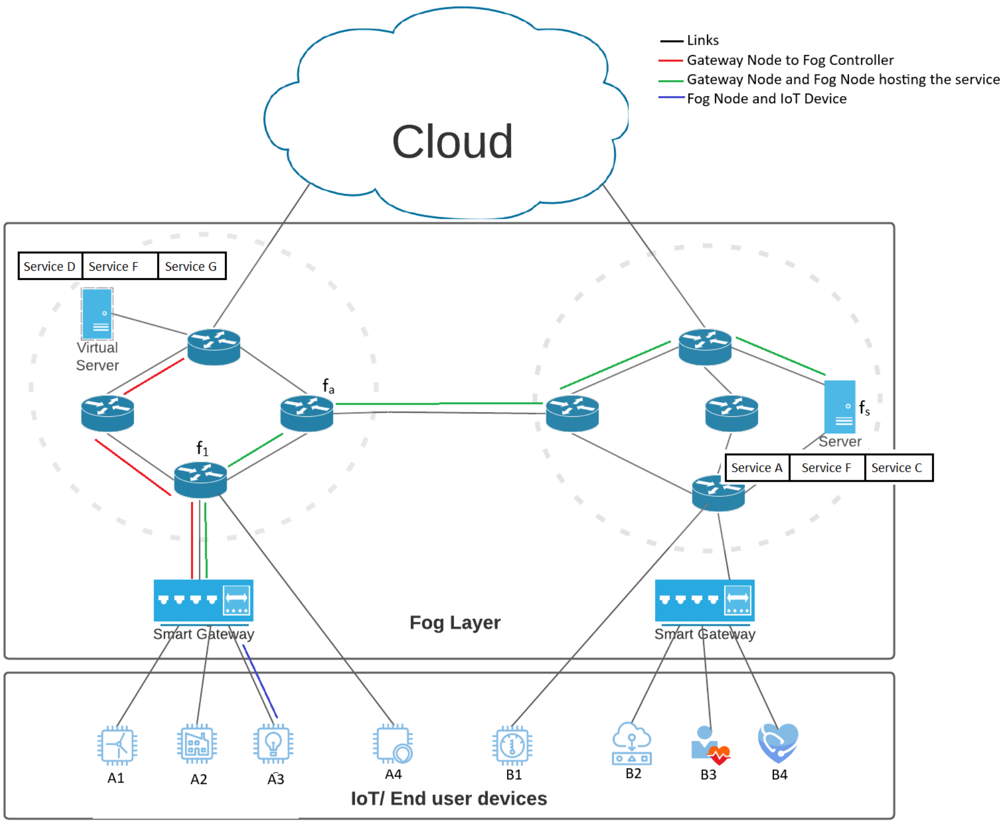


FIGURE 5.1: Problem Scenario

This research models a solution based on a three-layer fog architecture, as discussed in Section 2.4.3, and illustrated in Figure 5.1.

The bottom layer, or End Devices Layer, contains IoT, smart, and user devices that generate data and request services. These devices are typically resource-constrained and have diverse Quality of Service (QoS) needs. The volume, frequency, and persistence of data produced at this layer vary widely, influencing the nature and timeliness of the required service processing.

In the scenario shown in Figure 5.1, device A3 attempts to access Service C, which is unavailable in its local fog domain (Domain A) but is present in a neighbouring domain (Domain B). The challenge lies in enabling device A3 to access the service on a fog node in Domain B without routing through the cloud.

To address this, the proposed approach uses middleware distributed across fog nodes to support service discovery, request forwarding, and resource negotiation across domains. The process is as follows:

1. Device A3 sends a service request to the closest fog node, f_1, in the form:

$$R_i(id_i, s_c, m_i, l_o, p_i) \tag{5.1}$$

   where id_i is the user ID, s_c is the requested service, m_i indicates mobility status, l_o is the expected response time, and p_i is the priority based on SLA.

2. f_1 searches its local domain for s_c but does not find it.

3. f_1 contacts the domain controller, f_a, which identifies the location of s_c on fog node f_5 in Domain B.

4. f_1 forwards the request to f_5 and handles the conversion of the IoT message to a service request.

If latency or performance issues arise, f_1 may request f_a to migrate or replicate the service locally, depending on resource availability and cloud orchestration. Middleware plays a critical role throughoutenabling fog-to-fog communication, converting messages, and managing service registries dynamically.

The proposed solution is explored and evaluated through middleware-enabled modules in Chapter 4 using a remote EEG application as a case study.

5.2 Proposed Solution

Fog nodes may be any device between the cloud and the end user. Computational resources at different levels of the network (gateway, access, core) are made available for pre-processing or semi-processing of IoT data. Various entities own and manage fog nodes. Also, fog nodes may be organised into domains or run as stand-alone systems. Additionally, the computational resources made available by nodes vary and may be increased or reduced depending on availability and workload. Consequently, a fog node may not always have the resources to process IoT data sent to it.

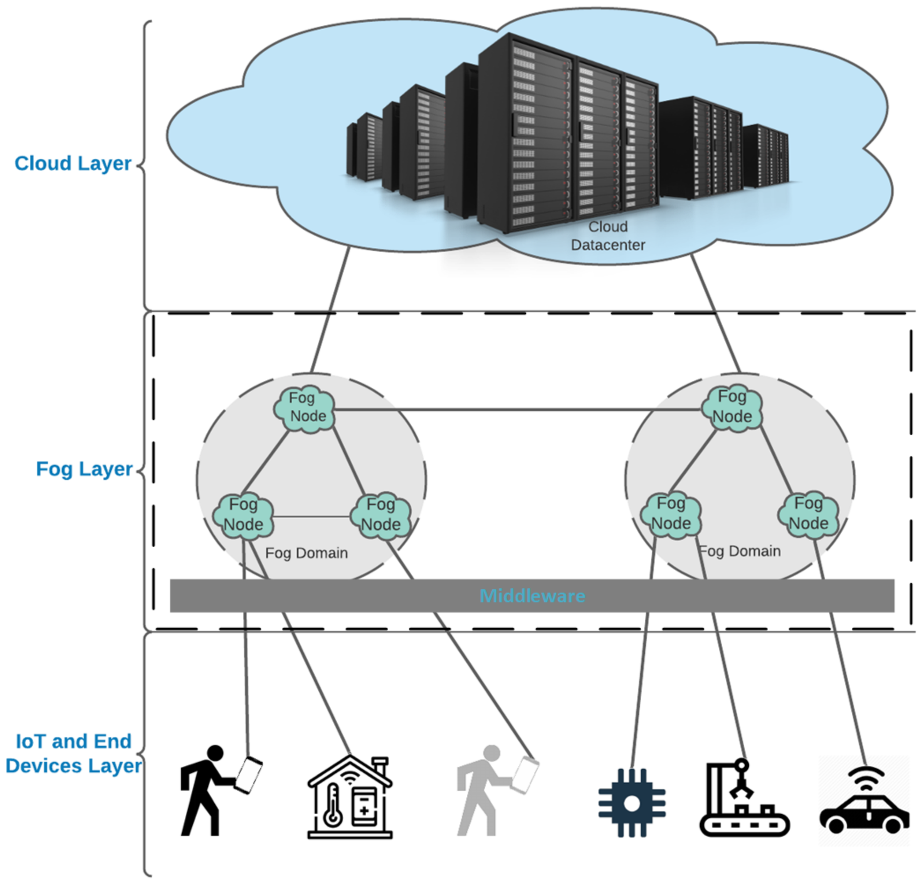


FIGURE 5.2: Architecture of the proposed model with middleware

Middleware is hosted on fog nodes and interacts with IoT devices, the cloud and other fog devices. Fog nodes may be any device with computational resources at any level of the network from consumer devices to dedicated servers. Consequently, their resources are not comparable to the significant computational resources available in the cloud.

Since fog nodes are managed and owned by different entities, they may run different operating systems or platforms and may not be interoperable. The role of the middleware is to make it possible for heterogeneous fog nodes to interact with each other for resource sharing to improve the system's reliability.

Services are applications or parts of applications which are used by IoT devices. Services perform single functions and may be chained to form a complete application.

Users/IoT devices request services from the cloud or fog devices. They access services by sending their user ID, service details and payload to the nearest fog node. IoT devices are resource-constrained.

The cloud is resource-rich and provides services for end users and IoT devices. All services are available in the cloud; however, some services are fully or partially shared with fog devices to improve their performance.

As presented in chapter 4, middleware is designed for communication among fog devices at the same level, lower or upper levels. The interaction between fog nodes is mainly for information exchange, request forwarding and load balancing. Fog nodes share data on the services they currently host and their resource availability. They also forward IoT requests to other fog nodes for processing and offload tasks to other fog nodes when overloaded. Moreover, middleware is designed to also communicate with the cloud. Communication with the cloud is for orchestration, exchanging end-user information, sending data for further processing, etc. Fog nodes perform various tasks for the cloud. This is an important distinction between fog computing and other edge-based paradigms, such as edge computing. In fog computing, fog nodes do not work independently of the cloud. The role of the middleware is to interact with the cloud to define and set up the expected role of the node. The middleware features involved in the work described here include the request handler, task scheduler, service registry, service discovery and communication modules. These modules presented in section 4.4.2 are evaluated in the next section using remote EEG application as a case study.

5.3 Case Study: Remote EEG Monitoring

To evaluate the proposed approach, this section presents a remote electroencephalography (EEG) application and compares its performance in three scenarios. The first scenario is direct processing on a single fog node closest to the user, the second scenario is using the middleware to improve application availability and

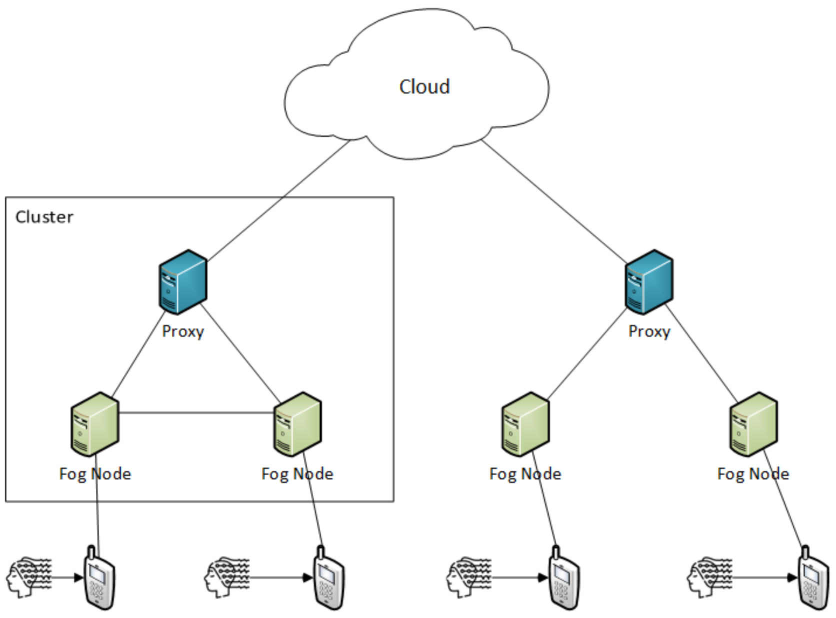


FIGURE 5.3: Architecture used in the simulation

the third scenario is processing in the cloud. Experiments run using iFogSim simulator [150], a popular Java-based simulation tool for fog computing as discussed in chapter 3. An electroencephalography (EEG) application that receives EEG signals from a user, processes the signals (on fog nodes or in the cloud), and sends feedback to the user is modelled for this evaluation. Fig. 5.3 shows the simulation setup.

Wireless Body Area networks make long-term physiological monitoring possible outside of the hospital [180]. Wireless Electroencephalography(WESN) provides the possibility for early detection, monitoring and treatment of diseases such as epilepsy, Parkinsons and Alzheimers. The WESN is made of an array of EEG sensing nodes each of which consists of an electrode array, signal processing unit and a transceiver for communication. Amplitude-integrated EEG may be useful in monitoring infants to predict the possibility of future or subsequent brain damage[181]. The EEG application is modelled with three modules: a client module, pre-processing and diagnosis, as shown in figure 5.4. The EEG sensor generates signals up to 200Hz, within the range of EEG signal monitoring. As EEG data is often noisy and often affected by artefacts on the patient or the environment, pre-processing is necessary to remove the noise from the signal [182].

Signals are cleaned and filtered by the client and pre-processing modules before being analysed by the diagnosis module.

TABLE 5.1: Values of simulation parameters

Parameters	Cloud	Proxy Node	Fog Node	EEG Sensor
Level	0	1	3	4
Rate per MIPS	0.01	0.0	0.0	0.0
RAM (MB)	40,000	4,000	4,000	1,000
Idle Power	16 × 83.25	83.43	83.43	82.44
Downlink Bandwidth (MB)	10,000	10,000	10,000	-
Uplink Bandwidth (MB)	100	10,000	1000	1000
CPU Length (MIPS)	44,800	5,600	2,800	500
Busy Power (Watt)	16 × 103	107.339	107.339	87.53

Simulations were run with three scenarios to evaluate the application's performance with the middleware's service discovery and location transparency. In the first scenario, modules are hosted on a single fog node (with no collaboration with other fog nodes). In the second scenario, the user request is forwarded to another fog device when the first node does not host the service. The third scenario runs the application completely in the cloud. Data is sent from one node to another after processing, and the feedback or actuation signal is sent back to the user. The Sensor and Display may be an EEG device and mobile phone attached to the patient or user, respectively. Fig. 5.4 shows the application design. Table 5.1 details the simulation parameters used.

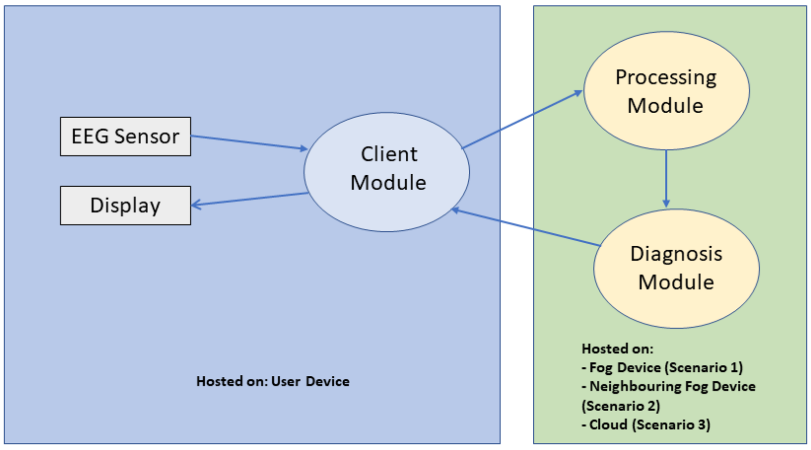


FIGURE 5.4: Application Model for the simulation

5.4 Results and Discussion

The results of the experiments are discussed in this section. Three metrics were used for the evaluation: the response time, network usage and energy consumption. The following subsections discuss the results of each of them and how they compare with other related results in the literature.

5.4.1 Response Time

Figure 5.5 illustrates the delay for the EEG application as the number of users per fog node increases for the three scenarios. Data must travel across multiple nodes, and with increasing network latency towards the cloud, the cloud-based scenario has the highest latency. Also, processing on the fog node closest to the user has a lower latency than processing on another fog in the same cluster. However, when the number of users increased to 5, processing on another fog node in the same cluster/domain produced a lower latency since the requests were processed on devices with more processing resources than the fog device that first received the request. In this case, the latency improvement is primarily due to the reduced queuing and scheduling delays, which outweigh the propagation delay incurred during task offloading. As the number of concurrent users increases, the processing delay on the neighbouring fog node surpasses the network delay associated with forwarding requests to a more capable fog node within the same cluster, making distributed load balancing more effective. These results also highlight the need to limit the load on fog nodes through load balancing. As shown in the figure, when 4 nodes are connected to the fog node the delay is higher than the delay from processing in the cloud by two users.

5.4.2 Network Usage

The network usage for the application in the three scenarios is shown in figure 5.6. Network usage increases significantly when processing is done in the cloud. As a

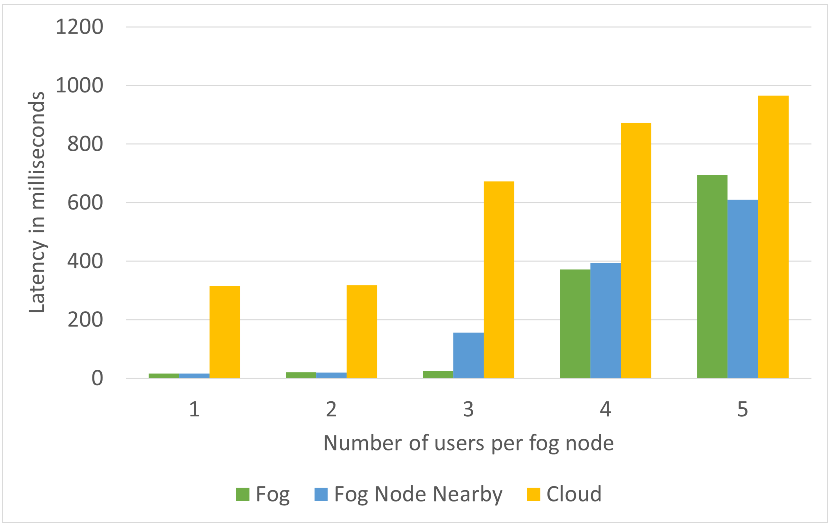


FIGURE 5.5: Comparison of application response time for the three scenarios

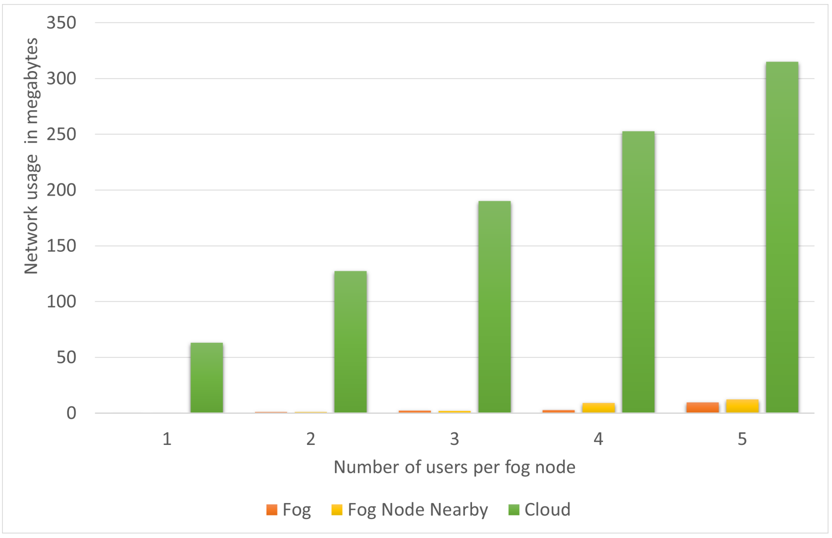


FIGURE 5.6: Comparison of network usage for the three scenarios

result of traversing multiple nodes and links and the increased latency link to the cloud, network usage in the cloud scenario is much greater compared to processing on fog nodes. This also shows the scalability of processing at the fog layer [150]. Moreover, processing on a fog node within the same cluster is done on the node closest to the user. Consequently, As the volume of EEG requests grows, increasing the density of fog nodes and implementing efficient load distribution mechanisms among these nodes can enhance system reliability, mitigating the need to send requests to the cloud.

5.4.3 Energy Consumed

Figure 5.7 presents the energy consumption of various devices in the simulation, including the cloud data centre, for a scenario with five users. The energy usage for fog devices is the same when requests are run at the fog layer but reduce to the same level as the proxy(neighbouring fog node) when requests are run in the cloud. When processing is done in the cloud, the energy usage of fog nodes is the same as that of the proxy since they all fog nodes as network devices and only forward traffic to nodes above or below them. Energy consumed by user devices remains the same for all scenarios because they process the client module of the application in all scenarios.

Comparing the results with [183, 184] reveals that while the middleware does introduce some energy overhead due to its service discovery and request routing mechanisms, this overhead is negligible compared to the baseline fog node consumption. More importantly, the energy efficiency gains achieved through intelligent resource allocationwhere the middleware enables local processing instead of unnecessary cloud offloadingsubstantially outweigh any additional consumption from the middleware components themselves. This demonstrates that the middleware not only has minimal negative impact on energy usage but actually improves overall system energy efficiency by ensuring requests are processed at the most appropriate layer.

The cloud data centre exhibits significantly higher energy consumption due to the extensive computational resources it maintains, as well as its substantial baseline energy usage, even in idle states. This consumption increases further when application processing is offloaded to the cloud, reflecting the additional computational workload imposed on the data centre infrastructure without fog processing.

Results for adopting a middleware approach to transparency in fog computing has been presented in this section. The evaluation uses simulations of an EEG application in iFogSim simulator. In [185], a remote pain monitoring application using fog computing was proposed and evaluated. The application sends surface

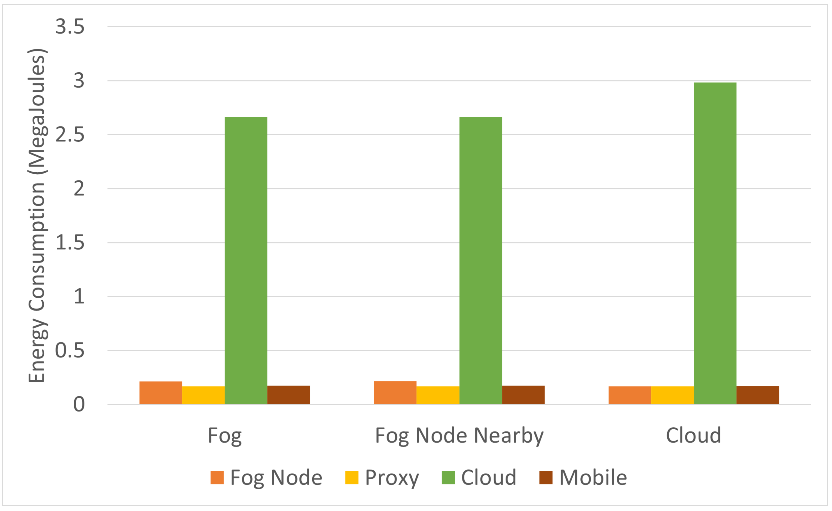


FIGURE 5.7: Comparison of energy consumption for devices in each scenario

electromyogram (sEMG) and electrocardiogram (ECG) signals and uses fog nodes for preliminary processing.

The authors in [186] propose a framework that provides platform transparency between application components and middleware. They used fog computing as a use case for the proposed framework and demonstrated that placing computational tasks at the right locations reduced CPU and network load. Their evaluation, however, focuses on CPU and network utilisation and does not consider response time or energy consumption.

5.5 Conclusion

This chapter has explored the use of middleware to enable efficient resource sharing in fog computing environments. The proposed middleware framework addresses two challenges in fog computing: location transparency and failure transparency. By dynamically tracking service availability across fog nodes and optimising request forwarding, the middleware ensures that users can access services regardless of underlying system changes.

Simulation results highlight the advantages of the middleware-enabled approach over traditional fog computing and cloud-based processing. The middleware reduces response times, optimises network usage, and enhances system reliability. By implementing load-balancing mechanisms and a middleware-based coordination strategy, the proposed solution ensures efficient resource utilisation. It improves service delivery, particularly for latency-sensitive applications such as EEG-based health monitoring.

CHAPTER 6

Middleware-enabled Cluster Interoperability

At the fog layer, nodes may be mobile, transitory or volunteer nodes made available for a brief unpredictable period [22]. Also, both fog nodes and IoT devices are heterogeneous in various ways–including size, availability and available resources. As shown in the previous chapter, offloading tasks to nearby fog nodes improves the reliability of applications at the fog layer and reduces the reliance on the cloud. This chapter extends those findings by introducing a clustered architecture at the fog layer. Clusters are modelled based on the concept of Autonomous Systems in the internet—a group of devices that are managed by a common entity.

Research on clustering in fog computing has focused on cluster formation and optimisation, but not on resource management and interoperability. This chapter covers these less-evaluated research topics and presents: i) an approach to resource sharing within and across clusters at the fog layer, controlled by middleware and ii) addresses interoperability in fog computing by presenting the notion of inter- and intra-cluster load balancing—an important contribution to increasing service availability in fog computing.

As the deployment of fog computing architectures expands, the need for efficient resource management and interoperability among heterogeneous devices becomes increasingly crucial. At the fog layer, nodes exhibit diverse characteristics in terms of mobility, resource availability, and computational capacity. Unlike traditional cloud-based computing, fog computing operates at different layers of the network (access, edge and core), providing localised processing to reduce latency and improve reliability.

Middleware provides an abstraction layer that manages inter-cluster resource allocation, ensuring that tasks are efficiently distributed across fog nodes and reducing reliance on distant cloud infrastructures. Furthermore, this chapter presents a comprehensive evaluation of the proposed approach, demonstrating its effectiveness in handling task scheduling and load balancing in various network topologies.

6.1 Clustering in Fog Computing

Given the decentralised nature of fog environments, clustering has emerged as an essential strategy for organising and managing resources effectively. Clustering aligns naturally with the hierarchical structure of the internet, where networks operate as Autonomous Systems (AS), each managed independently but collaborating through predefined agreements.

Existing clustering approaches in fog computing primarily focus on cluster formation. Several studies optimise clustering based on factors such as power consumption, latency, and resource distribution. However, these approaches often treat clusters as isolated entities, neglecting the potential benefits of interoperability between clusters. Without mechanisms for inter-cluster communication and coordination, fog networks risk inefficiencies and fragmentation.

Middleware can provide a robust framework to address this gap. By enabling interaction between clusters, middleware solutions can improve resource utilisation, fault tolerance and support dynamic workload distribution. This work explores how middleware can bridge the gap in fog clustering through a middleware-enabled collaborative fog computing ecosystem.

Clustering plays a critical role in fog computing by organising computing resources to optimise performance and efficiency. Several studies have investigated various aspects of clustering in fog environments. The work in [187] formulates the clustering problem as an optimisation task, aiming to minimise power consumption while meeting user latency requirements. Their approach organises small cell sites into clusters for processing user requests. Similarly, Martins et al. [188] propose a clustering algorithm based on node location but does not address inter-cluster collaboration.

The research in [189] explores clustering in the context of IoT, grouping IoT devices to manage data flow efficiently. Their approach introduces the concept of fog domains and fog colonies, with hierarchical control mechanisms to manage clusters. Meanwhile, the work in [190] adopts a cloud-centric clustering approach, where clusters are controlled by the cloud and led by designated fog nodes. This study formulates clustering as a mixed-integer linear programming problem and introduces an agent and platform manager, though it does not delve into their operational details.

A common theme across these studies is a focus on cluster formation. However, interoperability between clusters remains largely unexplored. Existing solutions lack mechanisms for seamless collaboration between clusters, which limits their scalability and adaptability in heterogeneous fog environments. Middleware presents a potential solution, providing a structured framework to facilitate interoperability and coordination between clusters.

6.2 Problem Description

Fog computing environments face significant challenges due to the inherent characteristics of fog nodes: they may be mobile, transitory, or volunteer nodes available only for brief, unpredictable periods [22]. This volatility, combined with the heterogeneous nature of both fog nodes and IoT devices, creates a fundamental problem: **How can fog computing systems ensure reliable service delivery**

and efficient resource utilisation when individual nodes are unreliable and resources are fragmented across isolated clusters?

6.2.1 Core Challenges

Current fog computing deployments face three interconnected challenges:

Resource Fragmentation

While clustering has emerged as a natural organisational strategy for fog nodes (mirroring Autonomous Systems in internet architecture), existing approaches treat clusters as isolated entities. This isolation leads to:

- Underutilisation of resources in some clusters while others are overloaded
- Inability to handle peak loads that exceed a single cluster's capacity
- Service unavailability when requested resources exist in neighbouring clusters but cannot be accessed

Lack of Interoperability

Despite extensive research on cluster formation and optimisation, the fog computing community has largely overlooked inter-cluster cooperation. Current clustering solutions:

- Focus exclusively on intra-cluster organisation
- Provide no mechanisms for clusters to share resources or coordinate tasks
- Cannot leverage the collective capacity of multiple clusters to improve reliability

Dynamic Resource Management

The transient nature of fog nodes requires adaptive resource management that current solutions cannot provide:

- Static cluster configurations cannot adapt to nodes joining or leaving
- No mechanisms exist for redistributing loads when nodes fail
- Lack of coordination between clusters prevents optimal task placement

6.2.2 Illustrative Scenario

Consider a smart city deployment where fog clusters are organised by district. During a major event in District A, the local fog cluster becomes overloaded with video analytics tasks from surveillance cameras. Meanwhile, the neighbouring District B cluster has idle resources. Without inter-cluster communication mechanisms:

- District A must either drop tasks or route them to the distant cloud
- District B's resources remain unused despite being geographically proximate
- The overall system fails to meet QoS requirements despite having sufficient aggregate resources

6.2.3 Research Gap

Existing clustering research in fog computing (as reviewed in Section 6.1) has established methods for cluster formation based on factors like power consumption, latency, and geographic location. However, these approaches stop at cluster boundaries, creating artificial barriers that prevent efficient resource utilisation across the fog layer.

This chapter addresses this gap by proposing a middleware-enabled approach that:

1. Enables resource sharing both within and across fog clusters
2. Provides mechanisms for inter-cluster load balancing
3. Maintains the autonomy of individual clusters while enabling cooperation
4. Adapts to the dynamic nature of fog environments

6.3 System Model

The system architecture follows the three-layer fog architecture. The three-layer fog architecture (figure 6.1) consists of the user or IoT layer, fog layer and the cloud layer. The fog layer is modified by introducing a centralised middleware clustered approach as discussed below. Details of the middleware framework and algorithms for cluster management are as presented in 4.

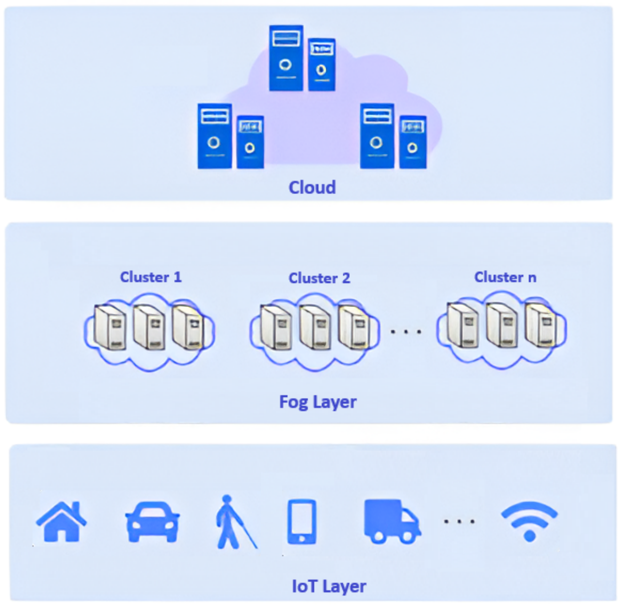


FIGURE 6.1: Fog Clustered Architecture

6.3.1 IoT/User Layer

The first layer has devices which interface with the external environment to collect relevant data. These devices sense environmental stimuli and collect readings over time under little or no direct human control or they may be devices under the direct control of a human user requesting services. The main characteristics of this layer are 1) Heterogeneity (in volume, velocity, value, variety and veracity of data) [191]. 2) Little or no computational power to process the data they produce [3]. Consequently, the data they produce will have to be processed at another layer (Fog or Cloud). 3) Limited power supply. Most IoT devices run on batteries [192].

6.3.2 Cloud Layer

The cloud layer consists of data centre computational, storage and networking resources which are available on-demand for users. The cloud has significant resources to process requests, however, the distant location of the cloud relative to the users means that processing in the cloud would be unproductive for applications requiring lower response times. Furthermore, the volumes of data produced could be prohibitive to send all to the cloud for processing as it would clog the network. The objective is thus to make use of fog nodes for processing requests as much as possible and to leave the cloud for resource-intensive and delay-tolerant applications.

In this model, the cloud is for the orchestration of services and applications. Delay-sensitive data is only sent to the cloud as a last resort when there are no available resources at the fog layer. It deploys services to fog nodes to make them as close as possible to the data source.

6.3.3 Fog Layer

The fog layer consists of processing, storage and networking resources that lie between the data producers (users) and the cloud. In the middleware-based model proposed in this work, fog nodes are organised into clusters based on the Autonomous Systems(AS) model in networking.

The fog layer is modelled as a graph $G(V, E)$ with vertices V as the set of fog nodes and edges E as the set of links connecting fog nodes. We define subgraphs $G_1(V_1, E_1)$, $G_2(V_2, E_2)$ ... $G_n(V_n, E_n)$ - fog clusters, and another subgraph $G'(V', E')$ such that

$$\forall G_i(V_i, E_i), \exists v_i \in V' \tag{6.1}$$

$$\forall G_i(V_i, E_i), \exists e_{ij} \in E' \tag{6.2}$$

Nodes which satisfy (6.1) are called fog cluster controllers/brokers and the links in (6.2) are inter-cluster links.

Each node v_i is characterised by a tuple $\{m_i, c_i\}$ where m_i is the processing capacity of the node in Million Instructions Per Second (MIPS), c_i is the processing cost for node i.

A fog cluster is modelled as an internet Autonomous System (AS), i.e. fog nodes within the same cluster are under common administrative control. ASs establish relationships with other ASs based on agreed terms. Inter-cluster resource sharing and collaboration require a common platform–middleware.

6.4 Evaluation and Results

This section evaluates the proposed middlewares ability to balance load among fog nodes within and across clusters, schedule incoming user requests with different scheduling schemes and communicate across fog clusters with different network topologies. One functionality of the middleware in a fog cluster is to manage the load among fog nodes within the cluster and to send requests to neighbouring clusters for processing. Hence there are interfaces for communicating with other fog devices as well as clusters. Furthermore, it evaluates the performance of different scheduling schemes from the literature with different applications of varying delay-tolerance to evaluate middlewares ability to meet the Quality of Service (QoS) requirements of applications. Lastly, it investigate the performance of the middleware in different network environments in terms of topology and inter-cluster distances.

Middleware components are implemented in a clustered fog environment using OMNeT++[1] as discussed in chapter 3. The simulation is implemented using the INET[2] framework and extending FogNetSim++ [152] to support clustering. OMNeT is a C++ based simulator for modelling communication networks, multi-processors and distributed systems. The INET framework provides protocols and models for modelling the internet stack in OMNeT. In addition, FogNetSim++ provides a framework for simulating fog systems to utilise INET's models.

Table 6.1: Simulation Parameters

Parameter	**Value**
Simulation Tool	Omnet++ 6.0.3
Frameworks	INET 3.6.8, FogNetSim++
Programming Languages	C++, NED Language
Number of Layers/ Fog Clusters/ Fog Nodes per Cluster	3/3/variable
Required MIPS for user requests	10 MIPS
Fog Node MIPS	1000 MIPS
Message Lengths	1024

OMNeT is well suited for this study because whereas other fog simulation tools focus on a specific aspect of fog computing, mostly resource management [143], with OMNeT, INET and FogNetSim it is possible to investigate the behaviour of the system at the network and application level. Also, its C++ implementation means simulations are more scalable and run faster than other tools written in other languages such as Java or Python. Table 6.1 shows details of the parameters used in the simulation study. User requests are modeled as a poison process with interarrival times following the exponential distribution with a rate of $\lambda = 0.2$.

[1]https://omnetpp.org/
[2]https://inet.omnetpp.org/

6.4.1 Inter-cluster Resource Sharing

IoT devices send requests and data to middleware and receive output under expected conditions without knowledge of where the processing occurs. At the fog layer, middleware ensures that the processing load is shared among fog devices within its cluster and with other clusters, avoiding overutilisation or underutilisation of nodes while ensuring that application performance expectations are met. Figure 6.2 shows the simulation setup for the three clusters used in the experiment. The variable x represents the distance between the first and second clusters. The distance between the second and third clusters is five times of x. Also, for this evaluation, all user requests are sent to the first cluster and forwarded to the others.

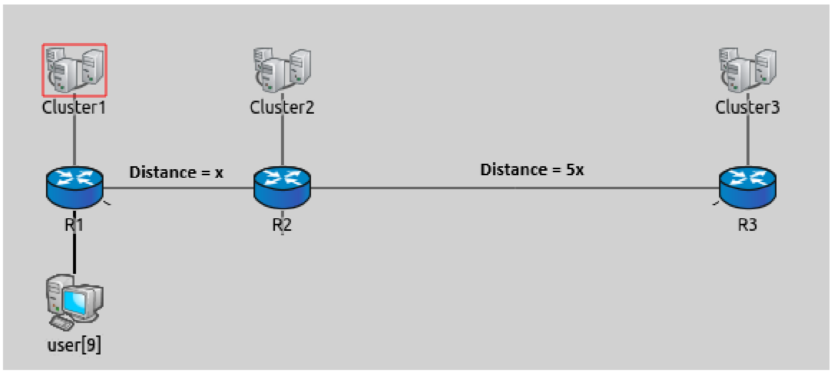


FIGURE 6.2: Inter-cluster distances

User requests at clusters are processed using middleware Algorithms 1 and 3 as discussed in chapter 4. Initial configuration of the middleware is dome with algorithm 2. Users send requests which include the processing time and constraints such as cost and required computational resources (MIPS). The cluster/fog controller searches for a node within the cluster that can process the request and also satisfy the constraints and forwards the request to it. If no node is available within the cluster, the controller searches the list of known clusters for one with resources for the request. The request is forwarded to the cluster with available resources. Sending to the cloud is the final option.

Figures 6.3 show the system's performance with load balancing processing within the same cluster as a benchmark. The results show better results for processing

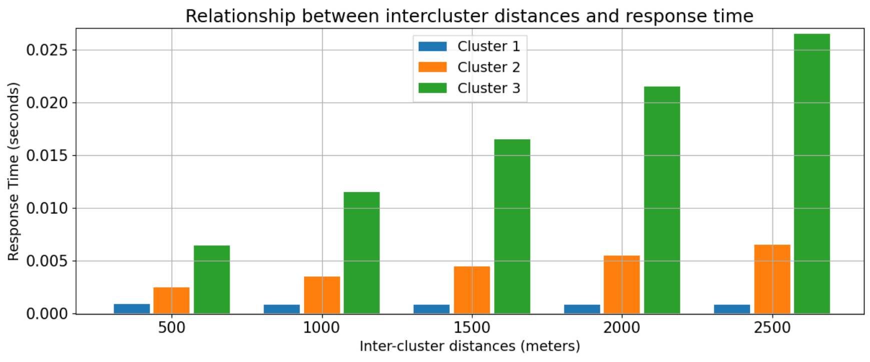


FIGURE 6.3: Response Times for Applications

within the cluster as expected and better results for processing in a neighbouring cluster instead of the cloud.

The response time is constant for processing within the same cluster, which was expected as the distance between fog nodes within the same cluster remains unchanged. The delay varied with distance for processing in the second and third clusters. Response time for processing on the third cluster increased much faster compared to the second cluster. Nonetheless the delay remained under $30ms$ for processing at a cluster which is $2.5km$ from the host cluster.

6.4.2 Scheduling

Task scheduling in fog computing has been studied extensively and is identified as an important resource management function for fog computing [136]. Nonetheless, most of the work on scheduling in fog focuses on node-level scheduling of tasks. To the best of our knowledge, no work has been done to evaluate intra-cluster scheduling. This section evaluates the performance of the middlewares intra-cluster scheduling.

The experiments compare the performance of four traditional scheduling algorithms from the literature for four applications with different priorities. The algorithms in the study are chosen because they are often used as benchmarks against which new proposals are evaluated. Also, due to their static nature, they are useful for evaluating the middleware's queue implementation.

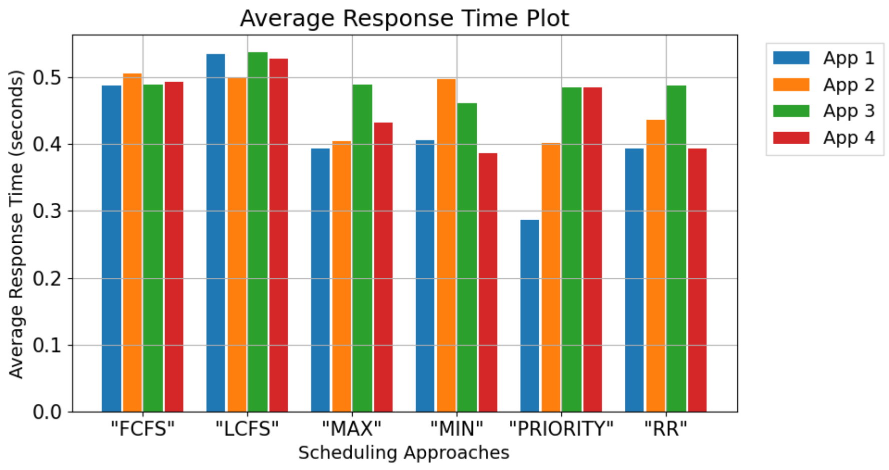


FIGURE 6.4: Response Times for Scheduling Schemes
Application Priorities: App 1: Highest; App 2: High; App 3: Medium; App 4: Low

Scheduling Scheme	**Implementation**
First-Come-First-Served (FCFS)	Requests are executed in the order they arrive.
Last-Come-First-Served (LCFS)	The most recently arrived request is executed first.
Round-Robin (RR)	Requests are executed in a cyclic order.
Max-Min (MAX)	Prioritises requests with higher resource demands first, ensuring large tasks progress while smaller tasks wait.
Min-Min (MIN)	Selects the smallest available request and schedules it first. It reduces overall completion time but may starve larger jobs.
Priority Scheduling (PRIORITY)	Requests are scheduled based on their priority levels. Higher-priority requests run first.

TABLE 6.2: The scheduling schemes used and their implementation approach for the experiments

The results for the simulation show that priority queuing produces the best results overall in terms of response time, although it does not guarantee that applications delay requirements are met. Last-Come-First-Served (LCFS) and First-Come-First-Served (FCFS) produced the worst results. Also, for all the algorithms apart from LCFS, app 1 (the highest priority application) had better delay performance than the other applications, although only priority scheduling gave the best results for it. Figures 6.4 and 6.5 show the results for response time and utilisation for the scheduling schemes evaluated.

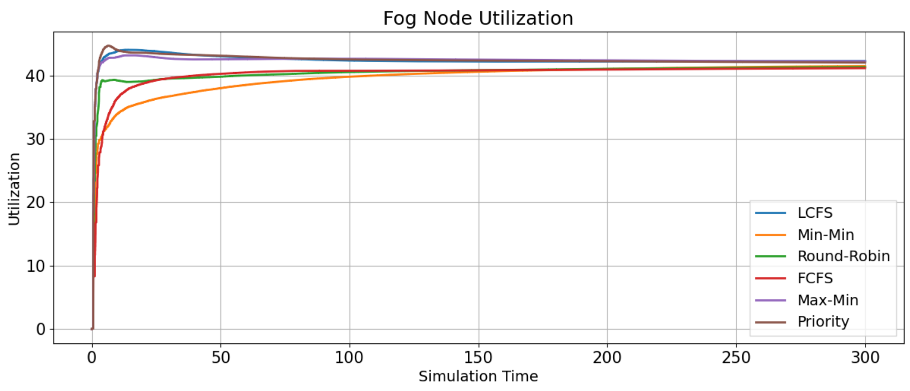


FIGURE 6.5: Fog Node Utilisation for different scheduling schemes

The results from comparing different scheduling schemes show that priority scheduling produces better results overall relative to other other approaches, which is consistent with similar work done in non-clustering fog/edge environments. However, the response time is significantly higher compared to the results from running the application without scheduling (figure 6.3). The increased delay results from the queuing of requests in the scheduling schemes. The inclusion of the queuing time significantly increases the delay performance of applications. This result justifies the need for clustering and load-balancing because as discussed in section 6.4.1, processing at a distant cluster still produces low response times.

6.5 Conclusion

This chapter has introduced a middleware-based framework designed to enhance interoperability and resource management in fog computing environments. By leveraging an Autonomous Systems-inspired clustering approach, the proposed middleware enables efficient intra- and inter-cluster resource allocation. The evaluation results indicate that the middleware effectively balances computational load, minimises response times, and optimises resource utilisation across fog nodes.

The findings show the importance of middleware in addressing the challenges of heterogeneity and dynamic resource availability in fog computing. By managing communication and collaboration between clusters, middleware reduces reliance on cloud resources and improves the availability of fog-based applications.

CHAPTER 7

Middleware-enabled Fog Peer-to-Peer Model

Fog computing, as an emerging paradigm, originally proposed a hierarchical architecture where computational resources increase as one moves from the edge towards the cloud [3, 96, 193, 194]. This model relies on multiple layers of nodes, with lower layers handling the pre-processing of user requests and higher layers, including the cloud, performing more resource-intensive tasks. However, this approach faces significant challenges: processing tasks could end up being routed all the way to the cloud, creating inefficiencies and potentially undermining the very purpose of fog computing reducing latency and increasing real-time processing at the edge. Additionally, the hierarchical design creates potential bottlenecks and a single point of failure, particularly in the centralised nodes that act as intermediary layers for request forwarding.

To address these issues, this chapter introduces a novel solution: a middleware-based, peer-to-peer (P2P) model for the fog layer. Unlike the hierarchical model, the proposed approach enables fog nodes to interact as peers within an unstructured, decentralised network. Fog nodes send requests to nodes in their peers in the overlay network first rather than defaulting to higher layers, thus reducing dependency on nodes at high layers (the core or cloud). The aim of this approach is to improve reliability by increasing the availability of services close to the network edge.

The problem with the hierarchical fog model is its reliance on a centralised structure. Requests cascade upwards to the cloud when lower-layer nodes cannot process them. While early fog computing models aimed to optimise the distribution of computational load, they inadvertently introduced potential failure points and inefficiencies as the system scaled. Therefore, the challenge is how to improve the performance of applications at the fog layer without relying on a single centralised node, which could become a bottleneck or single point of failure.

In response to these concerns, this chapter evaluates a decentralised, peer-to-peer fog architecture, comparing its performance with a cluster-based approach. Both architectures are implemented using the proposed middleware framework. Using the self-adaptive mechanisms presented in section 4.4, middleware can adapt to both architectures. The self-adaptive mechanism allows the framework to switch between different configurations to optimise performance in a distributed environment where nodes may be unreliable or frequently changing. The evaluation presented here highlights the advantages and drawbacks of centralised versus decentralised architectures, contributing insights into how the fog layer can be deployed for better reliability and performance in edge-based applications.

The middleware-based P2P solution presented in this chapter is an important contribution of this thesis. It provides a flexible, decentralised framework for the fog layer. The chapter also includes a detailed simulation-based comparison of the decentralised P2P model and the hierarchical cluster approach. The results underscore the need for adaptive strategies in fog resource management, as proposed throughout this thesis.

7.1 Problem Statement

The fog layer is unique because integrates edge and cloud computing resources to support real-time, low-latency applications. Traditionally, fog computing has been implemented with a hierarchical structure, where nodes with progressively greater resources are higher up in the architecture. This structure, originating from early fog computing definitions, is widely represented in tools like iFogSim. In these architectures, user requests are initially processed at the lowest fog node and forwarded upwards if not handled locally. Eventually, if a request cannot be processed at any layer, it is sent to the cloud. This hierarchical forwarding model, however, suffers from inefficiencies, particularly as requests are often routed to the cloud even when other fog nodes at the same layer might have the necessary resources to process them.

Peer-to-peer (P2P) systems, both centralised and decentralised, offer potential solutions to these issues [195]. Centralised P2P systems rely on a central server to manage the network and provide information about resource locations, while decentralised P2P systems distribute this responsibility across all nodes, ensuring no single point of control. The decentralised nature of P2P networks can offer significant advantages in terms of scalability, fault tolerance, and resource distribution, making them well-suited for fog computing environments. The symmetric role of nodes in a decentralised P2P system eliminates the dependency on a central node, mitigating risks associated with bottlenecks and failures.

A decentralised P2P architecture for the fog layer, where nodes connect and interact as peers in an unstructured network is proposed. This approach allows nodes to forward requests to nearby neighbours first, rather than relying on higher-layer nodes or the cloud. This design not only improves system efficiency but also enhances resilience by ensuring that no single node is critical to the systems operation. By considering the inherent unreliability of fog resources, where nodes frequently join and leave the network, this work explores how such a decentralised architecture can handle dynamic changes in network topology and resource availability without compromising performance or reliability.

The traditional hierarchical architecture of fog computing, while intuitive in its design, presents fundamental limitations that undermine the core benefits of fog computing. This chapter addresses a critical problem: **How can fog computing systems avoid the inefficiencies and vulnerabilities inherent in hierarchical architectures while maintaining effective resource utilisation and service delivery?**

7.1.1 Limitations of Hierarchical Fog Architectures

The hierarchical fog model, as originally proposed [3], organises nodes in layers with increasing computational resources from edge to cloud. While this structure appears logical, it creates several critical problems:

Vertical Routing Inefficiency

In hierarchical systems, requests follow a rigid upward path when local processing is unavailable:

- Requests cascade through multiple layers before reaching capable nodes
- Lateral resources at the same layer remain unused
- Many requests unnecessarily reach the cloud, negating latency benefits
- Network congestion increases due to concentrated vertical traffic

Structural Vulnerabilities

The hierarchical design introduces architectural weaknesses:

- Intermediate layer nodes become bottlenecks under high load

- Single points of failure exist at each hierarchical level
- System resilience depends on the availability of specific nodes
- Scalability is limited by the capacity of upper-layer nodes

Resource Underutilisation

The rigid structure prevents optimal resource usage:

- Peer nodes with available resources cannot directly assist each other
- Load balancing is restricted to vertical distribution
- Dynamic resource allocation across layers is complex and inefficient

7.1.2 Illustrative Scenario

Consider a smart manufacturing environment with fog nodes distributed across a factory floor. In a hierarchical model:

1. Machine A generates a compute-intensive analytics request 2. Its local fog node lacks resources, so the request moves to the area controller 3. The area controller is overloaded, forwarding the request to the plant-level server 4. Eventually, the request reaches the cloud for processing

Meanwhile, Machine B's fog node in the same area has idle resources but cannot assist because the hierarchical structure prevents direct peer communication. This results in: - Increased latency (multiple hops to cloud) - Wasted local resources (Machine B's idle capacity) - Network congestion (vertical traffic concentration) - Potential failure if any intermediate node fails

7.1.3 The Need for a Decentralised Approach

These limitations reveal that the hierarchical model, despite its widespread adoption in fog computing implementations (including popular simulators like iFogSim), fundamentally conflicts with fog computing's goals of:

- Minimising latency through edge processing
- Maximising resource utilisation across distributed nodes
- Ensuring system resilience and fault tolerance
- Supporting dynamic and scalable deployments

This chapter investigates whether a decentralised, peer-to-peer (P2P) architecture can overcome these limitations while maintaining the benefits of fog computing. Specifically, it explores:

1. Can a P2P fog architecture reduce dependency on hierarchical routing?
2. How does peer-based resource sharing compare to hierarchical distribution?
3. What are the trade-offs between centralised coordination and decentralised autonomy?
4. Can middleware effectively support both architectural models?

The following sections present the middleware-enabled P2P solution and provide a comprehensive comparison with the hierarchical cluster-based approach presented in Chapter 6.

7.2 Proposed Model

As discussed in Section 4.4.2, the proposed middleware employs the MAPE-K (Monitor, Analyse, Plan, Execute over shared Knowledge) self-adaptation model to dynamically manage the fog environment. The adaptation algorithm and framework are illustrated in figure 4.3 and algorithm 2. This section evaluates the implementation of the peer-to-peer (P2P) architecture presented in 4 and compares it to the clustered approach.

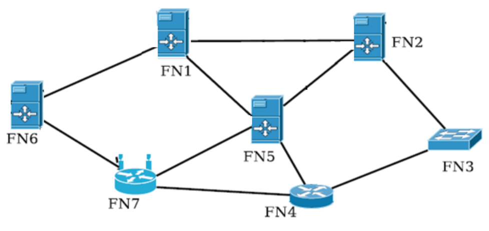
7.3 Cluster-Based Architecture

In the cluster-based architecture (chapter 6), fog nodes are organised into clusters, each managed by a fog controller. Middleware runs both on the fog nodes and on the controller/broker nodes. IoT requests are sent to the controller, which processes the request and selects an appropriate fog node within the cluster to handle it. If no suitable fog node is available within the cluster, the controller forwards the request to a neighbouring cluster with available resources. Clusters share resources by updating neighbouring cluster controllers about their available resources.

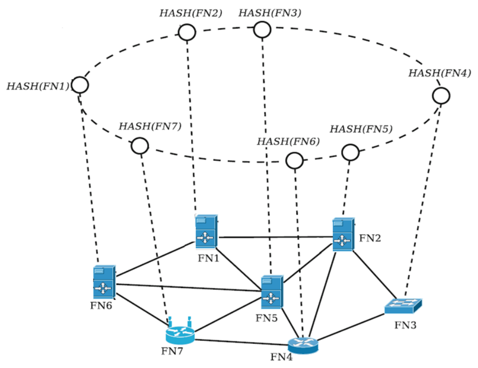
7.4 Peer-to-Peer Architecture

In the decentralised P2P approach, all fog nodes act as peers within an overlay network. Each fog node maintains connections with two other nodes, forming a ring-chain topology. Nodes share resources by forwarding requests to their connected peers. User requests may propagate across multiple peers until a suitable node is found to process them. Middleware manages the P2P system, ensuring efficient request handling and resource allocation.

Figure 7.1 shows the two architectures used in this study. The peer-to-peer network creates an overlay network which is independent of the physical topology. Nodes connect to each other based on availability to join the overlay and proximity to peers. In the cluster case, the central node (FN5) exchanges messages with all other fog nodes as well as from user nodes.



(A) Clustered approach



(B) Peer-to-peer overlay network

FIGURE 7.1: Architectural Approaches

7.4.1 Peer Bootstrapping

Bootstrapping refers to the process by which new nodes to a peer-to-peer system discover existing peers and form connections with them to join the network [196]. The proposed model adopts a decentralised mechanism to avoid reliance on central servers (bootstrapping nodes). When a new node n_i joins the network, it selects two existing nodes n_j and n_k as its peers based on a cost function $C(n_i, n_j)$ that considers factors such as propagation delay d_{ij} between n_i and n_j, and computational capacity R_j of node n_j.

The cost function is defined as:

$$C(n_i, n_j) = \alpha d_{ij} + \beta \frac{1}{R_j}, \tag{7.1}$$

where α and β are weight factors that balance delay and resource availability.

A new node n_i selects its two nearest peers by minimizing $C(n_i, n_j)$:

$$n_j, n_k = \arg\min_{n_x \in N} C(n_i, n_x). \tag{7.2}$$

7.4.2 Request Handling

When a fog node n_i receives an IoT request r, it either processes the request locally or forwards it to one of its connected peers. The forwarding decision is based on:

- Local resource availability R_i
- Request priority $P(r)$
- Estimated processing time $T(r, n_j)$ on peer n_j

If the request cannot be processed locally, it is forwarded to the peer n_j that satisfies:

$$n_j = \arg \min_{n_x \in P_i} T(r, n_x). \tag{7.3}$$

To prevent excessive forwarding, a counter c_r and QoS constraint $Q(r)$ are enforced. The request is dropped if $c_r > Q(r)$.

The bootstrapping process is illustrated in the sequence diagram in figure 7.2. In the figure, fog node 1 is connected to nodes 3 and 2 as its neighbours, while node 4 is a new node attempting to join the network by reaching out to node 1.

7.4.3 Peer Connection Management

Each node maintains only two peer connections. When a node n_i leaves the network, it informs its neighbours n_j and n_k, providing them with each other's addresses. The peers then establish a direct connection or reconfigure their connections using the selection process described in section 7.4.1.

This structured approach ensures a scalable and resilient P2P-based fog computing system, with efficient resource allocation and minimal network overhead.

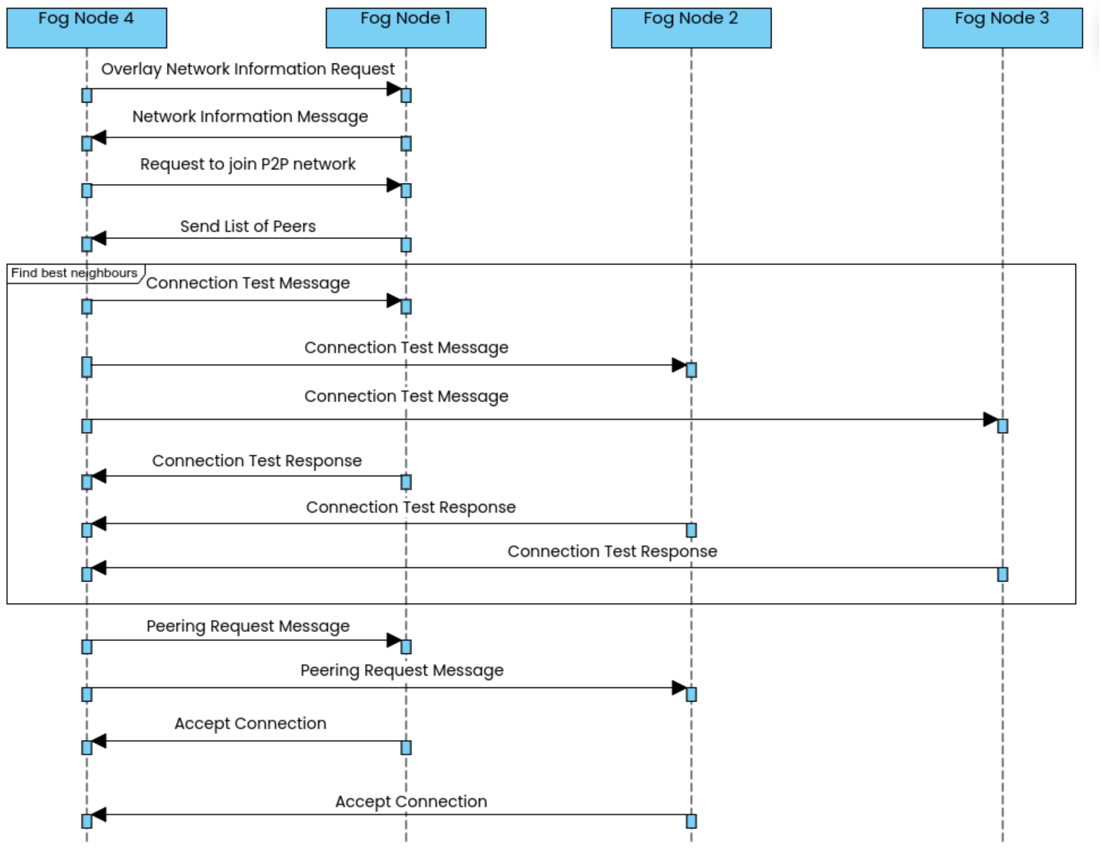


FIGURE 7.2: Bootstrapping process for new peers to join the network.

7.5 Evaluation

To evaluate the proposed middleware, simulations are conducted in OMNeT++ for both cluster-based and peer-to-peer architectures. OMNeT++ is a framework designed for network simulations, with a modular approach and flexibility that make it well-suited for evaluating internet-based architectures. The INET framework extends OMNeT++ for internet protocol simulations, while FogNetSim++ further extends INET to include IoT application protocols such as MQTT.

The simulations include user nodes (IoT devices), fog nodes, and a cloud data centre. Each user node runs a single MQTT application that generates sensor data, forwards it to the next processing layer, and receives a response. Since user nodes have limited processing capabilities, they send MQTT requests to fog nodes for processing. In the cluster-based architecture, these requests first arrive at a

TABLE 7.1: Simulation Parameters

Parameter	**Value**
Simulation Tool	OMNeT++ 6.0.3
Frameworks	INET 3.6.8, FogNetSim++
Programming Languages	C++, NED Language
Required MIPS per IoT request	10 MIPS
Fog Node Capacity	1000 MIPS
Number of IoT Requests	3500
Request Arrival Distribution	Exponential ($\lambda = 100ms$)
Message Length	1024 bytes

broker, which selects an appropriate fog node within its cluster for processing. If resources are unavailable at the fog layer, requests are forwarded to the cloud.

Each fog node has a processing capacity of 1000 MIPS and operates as a single processing unit without virtual machines. The cloud consists of 20 processing units with a combined capacity of 90000 MIPS. The arrival of IoT requests follows an exponential distribution with a mean inter-arrival time of $\lambda = 100ms$. The parameters used for the simulations are shown in table 7.1.

7.6 Analysis of Results

The results of the simulation of the peer-to-peer network show the relationship between the distance between peer nodes in the peer-to-peer architecture and the response time of IoT applications. The results indicate that while processing at the fog layer significantly reduces delay, the distance between fog nodes influences resource-sharing efficiency. When fog nodes are within 5 km of each other, the delay remains below 100 ms. However, for every additional kilometre, the delay increases by approximately 20 ms.

In real-world deployments, peer node distances vary significantly based on the application domain and deployment environment. In fog and edge networks, the physical area covered can contain hundreds of thousands of devices, meaning that network grade and quality can vary by orders of magnitude, from DSL lines to fibre optics, while the distances involved result in much higher latencies between nodes than in cloud data centres. In industrial settings, IoT sensors on factory floors can often use wired connections, minimising distance-related delays. However, mobile resources, such as autonomous vehicles, or isolated resources, such as wind turbines in the middle of a field, require alternate forms of connectivity, where distance becomes a critical factor.

Agricultural deployments face unique challenges, as poor internet connectivity is one of the most common issues in smart farms, especially in rural areas. In these scenarios, fog layers are installed in local farms and are responsible for real-time data analytics such as predicting pests and diseases, yield prediction, weather prediction, and agricultural monitoring automation. The distributed nature of agricultural environments means fog nodes may need to cover extensive geographical areas, making distance-aware deployment strategies essential.

Middleware's adaptive mechanisms ensure that the P2P architecture remains viable across diverse deployment scenarios, from dense urban environments with closely spaced nodes to sparse rural deployments where nodes may be separated by significant distances. The fog computing paradigm's emphasis on proximity to end-users and dense geographical distribution aligns well with our adaptive approach, allowing the middleware to optimise performance based on actual deployment conditions rather than assuming uniform node distribution.

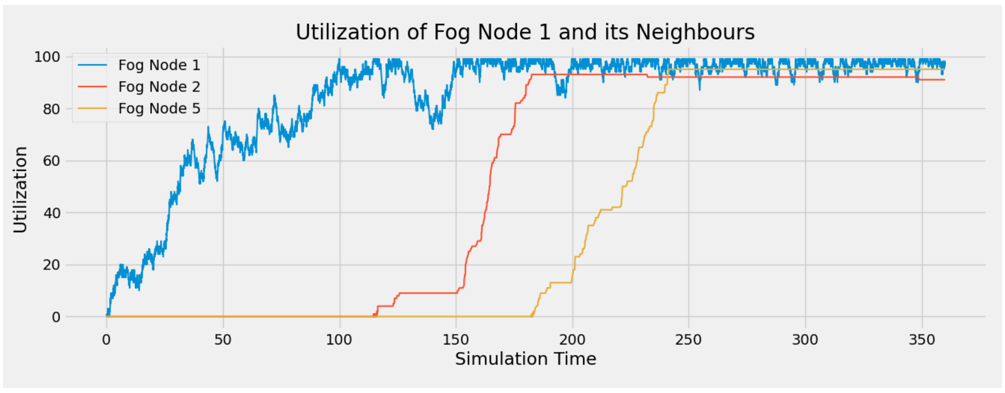


FIGURE 7.3: Utilisation among peer nodes

The results for the utilisation of fog nodes, which are connected as peers, are shown in Figure 7.3. In this figure, fog node 1 has nodes 2 and 5 as neighbours, with node 2 as its closest neighbour. User requests are sent to node 1. As the figure shows, when node 1's utilisation reaches a hundred per cent, it starts sending requests to its best neighbour—node 2. Node 5 receives requests once node 2 is overloaded.

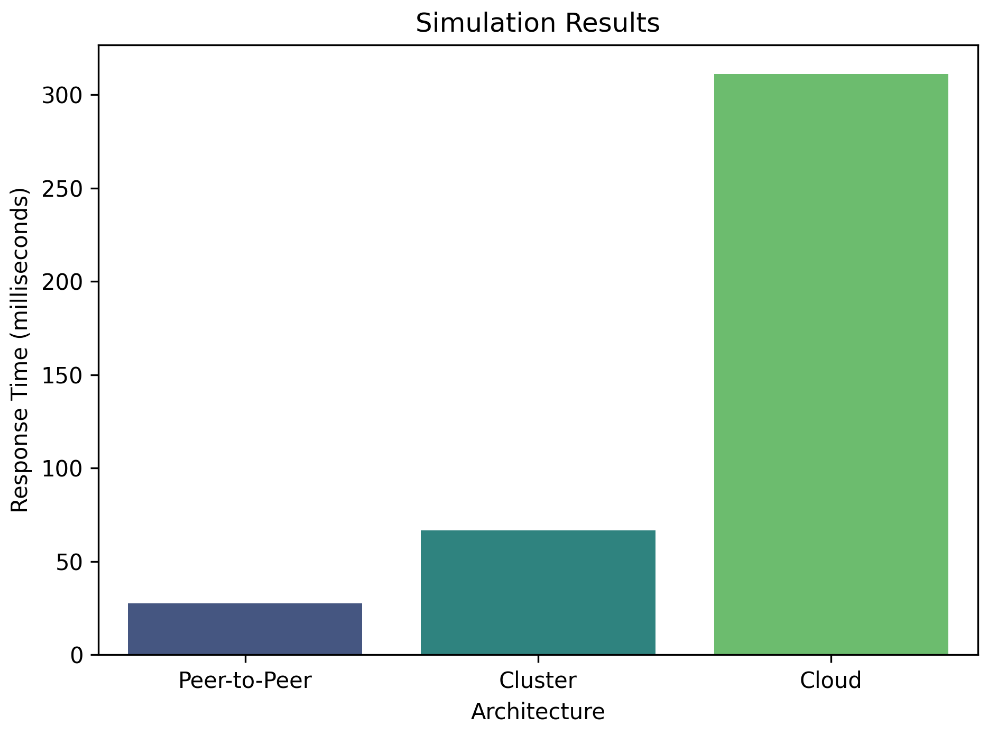
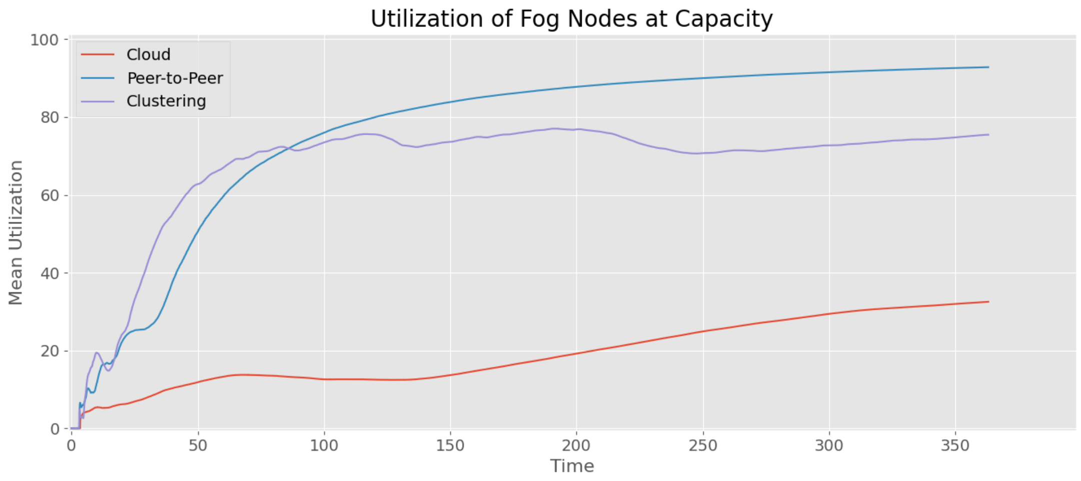


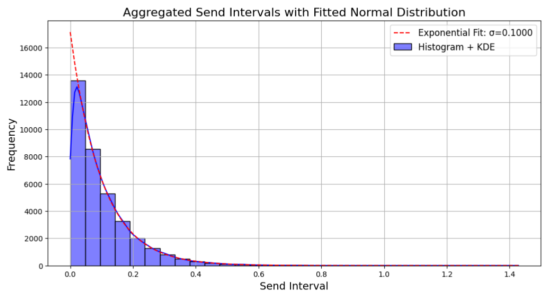
FIGURE 7.4: Response times for Peer-to-Peer, Clustered, and Cloud processing

Now, turning to the comparison of the peer-to-peer and cluster approaches, figure 7.4 compares the response times of the peer-to-peer, clustered, and cloud architectures. The peer-to-peer architecture demonstrates the lowest response time when processing occurs at the first available fog node. In contrast, the clustered architecture exhibits higher response times due to the additional overhead of forwarding requests to a broker before reaching the processing node. Nonetheless, both fog-based architectures significantly outperform cloud processing, which incurs a delay of approximately five times higher. These findings align with previous research, such as [197].

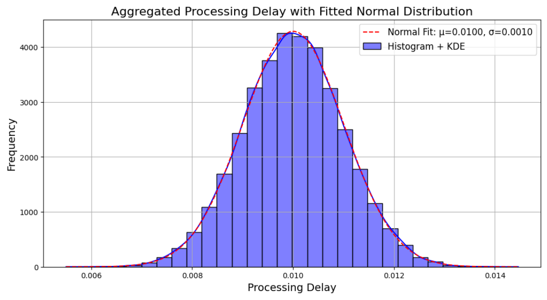
Figure 7.5 presents the mean utilisation across the different architectures and the distributions for the user requests, inter-arrival delays, and processing delays. The cloud server exhibits a slow rise in utilisation, remaining below 40% throughout the simulation. In contrast, the utilisation of fog nodes increases rapidly. In the clustered scenario, utilisation remains below 80%, as the broker node efficiently distributes traffic among fog nodes. However, in the peer-to-peer scenario, utilisation quickly reaches nearly 100% since requests are only forwarded when a node cannot process them. Despite this high utilisation, application performance remains stable, as forwarding only occurs when necessary.



(A) Fog node utilisation over time.



(B) Inter-arrival time distribution.



(C) Processing delay distribution.

FIGURE 7.5: Comparison of utilisation and request distribution characteristics.

7.7 Conclusion

The simulation results demonstrate that the peer-to-peer architecture provides lower response times compared to the clustered architecture and cloud-based processing. While clustered architectures benefit from load balancing via a broker, the additional routing overhead increases the delay. In contrast, the peer-to-peer architecture leverages direct communication between fog nodes, significantly improving response time and resource utilisation.

However, the effectiveness of peer-to-peer architectures is influenced by node proximity, as increasing the distance between peers leads to higher communication delays. The utilisation results indicate that peer-to-peer fog nodes reach full capacity more rapidly, though this does not negatively impact performance, as request forwarding ensures continued processing efficiency.

Future research should explore dynamic peer selection strategies and hybrid architectures that combine clustering with peer-to-peer communication to optimise resource allocation further. Additionally, real-world deployments of the proposed middleware should be investigated to validate the findings under practical conditions.

CHAPTER 8

Discussion and Future Directions

The findings from evaluating the proposed middleware and its components produce promising outcomes. The adaptive design demonstrates the benefits of adapting the fog system based on external and internal factors. The results from evaluating different architectures showed the relative merits and demerits of a centralised and decentralised architecture. This justifies the need for an adaptive self-configuration design.

This section discusses the key findings of the research under three themes: (1) availability and reliability of applications, (2) adaptiveness and distributed architectures, and (3) other factors influencing performance.

8.1 Availability and Reliability

This thesis has demonstrated the role of availability and resource pooling in the context of fog computing and their impact on application performance. The findings in section 5.4 show that high availability leads to improved system reliability, which is relevant in fog computing environments as resources are distributed across multiple nodes and clusters. By extending resource availability from individual nodes to clusters and expanding further through inter-cluster resource pooling, the system's robustness and response time improve [198]. An increase in availability not only improves reliability but also enables the efficient processing of requests at the fog layer, where resources are inherently constrained and unreliable.

At the fog layer, the unreliability of nodes requires constant tracking and monitoring for effective resource management. The middleware framework is critical in this regard, as it provides a transparent way to store, monitor, and utilise resources. The middleware approach increases the fog layers capacity for resource management, which directly leads to increased availability [159]. Middlewares ability to track resources enables dynamic allocation and load balancing and ensures that nodes at the fog layer can collaborate effectively to meet application demands. This mechanism reduces the risk of overloading individual nodes and thus improves application latency.

The results of the performance evaluation, presented in section 6.4, highlight the significant advantages of inter-cluster collaboration in terms of response time and overall application performance. Sharing resources within and across clusters at the fog layer improves the performance of fog applications and offers a more responsive alternative to relying solely on the cloud. Also, balancing loads across different nodes at the same fog layer provides better performance outcomes than processing on a single node, which might lead to queuing delays [25, 199]. Additionally, sharing resources across clusters further reduces resource bottlenecks and the latency associated with reliance on the cloud.

However, while resource pooling and sharing can improve performance, load balancing at the fog layer does not guarantee optimal outcomes in all scenarios [200]. Variations in incoming requests, network conditions, and the dynamic availability of fog nodes can lead to unpredictable performance, indicating that the success of load-balancing strategies depends on the characteristics of user requests and the system's architecture. Despite this, the findings provide valuable insights into the potential of middleware-enabled resource sharing to address these challenges, underscoring its importance in maintaining application performance in dynamic environments.

The evaluation of the middleware in chapter 5 further reinforces the importance of resource sharing as a central component in managing the fog computing environment effectively. As outlined in section 5.4, the middleware facilitates load sharing among fog nodes, ensuring that latency is minimised and application performance is optimised. These results contribute to the growing body of work in edge and distributed computing, offering a comprehensive framework for improving service availability and resource utilisation in dynamic and unpredictable conditions. Through the inter- and intra-cluster resource-sharing mechanisms, this research highlights the relationship between an increased resource pool at the fog layer and application performance and reliability, thereby providing a robust foundation for future developments in edge-based computing architectures.

In summary, this research emphasises the role of availability, resource pooling, and middleware in the performance of fog computing systems. By extending the resource pool at the fog layer and using effective load-balancing strategies, the findings demonstrate how these elements collectively improve the performance of applications in dynamic environments. This contribution paves the way for further advancements in the field, offering practical solutions for addressing resource management challenges in distributed edge computing.

8.2 Adaptiveness and Distributed Architecture

A novel contribution of this study is the use of the MAPE-K self-adaptation framework in middleware for fog resource management (section 4.4.2). The adaptive approach enables the dynamic implementation of different architectures at the fog layer based on environmental conditions. The behaviour of fog nodes is modified using the self-adaptive mechanism in middleware. Dynamically changing the fog architecture to suit the environment is essential to fog computing. Prior studies have demonstrated the usefulness of different applications and environments at the fog layer [185, 189, 190].

On the question of the usefulness of an adaptive architectural approach to fog computing, the results presented in Chapter 7 highlight the relative benefits of centralised and distributed architectures in a fog scenario. The unstructured design choice used for the decentralised approach enables the system to capitalise on the inherent benefits of P2P architectures, such as improved scalability, flexibility for diverse applications, and a lower barrier to entry. This is particularly relevant for dynamic applications like vehicular fog computing, where fog nodes are mobile and subject to frequent changes in connectivity and availability [201]. The adaptability of the proposed middleware allows for real-time reconfiguration of the fog architecture to ensure continuous service delivery despite environmental fluctuations.

While a distributed or peer-to-peer fog architecture offers high scalability and flexibility, a centralised or clustered approach presents advantages in terms of resource utilisation and load management. In a centralised system, the coordination of resources is under tighter control. This leads to a more efficient workload distribution and enhanced reliability as control allows the implementation of resource management techniques to suit application needs. The trade-offs between these architectures are significant: whereas a P2P system promotes decentralisation and ease of deployment, a centralised architecture ensures greater control and stability.

In the end, the choice between a P2P and a centralised fog architecture depends on the specific requirements of the application. For highly dynamic and large-scale environments, a distributed model provides better adaptability and scalability. On the other hand, for applications that require strict control of resources and reliability, a centralised approach may be more appropriate. This is further justification for the adaptive middleware approach proposed. Middleware that adapts to the given fog environment plays a critical role in allowing dynamic architecture selection.

8.3 Factors Contributing to Performance

Other factors at the fog layer influence application performance. The study shows that one key factor affecting application performance is the proximity between fog nodes, which directly impacts propagation delay. Increased propagation delay leads to higher response times, reducing the overall efficiency of fog-based applications.

Different topologies at the fog layer exhibit varying performance characteristics. The traditional hierarchical approach has been widely adopted in fog computing [3]; however, alternative approaches proposed here, such as clustering and peer-to-peer (P2P) configurations, offer distinct advantages that improve application performance. The clustering-based architecture improves availability and reliability by ensuring that a pool of fog nodes is available to process requests, reducing potential failures and improving resource utilisation.

Similarly, a P2P approach to fog computing enhances the utilisation of fog nodes by enabling direct communication and resource sharing among nodes without a strict hierarchical structure. This reduces bottlenecks often associated with centralised control and improves scalability.

Additionally, scheduling techniques at the fog layer play a crucial role in maintaining application performance. Traditional scheduling approaches, such as First-Come, First-Served (FCFS), do not effectively prioritise critical applications, which would lead to suboptimal performance in time-sensitive scenarios. Priority scheduling mechanisms have improved performance by ensuring that higher-priority tasks receive timely processing, thus improving the quality of service for latency-sensitive applications.

Overall, while the hierarchical approach has been commonly employed in fog computing, adopting clustering and P2P strategies can yield substantial benefits in terms of reliability, availability, and resource utilisation. Future work should explore hybrid models that leverage the strengths of multiple architectures to optimise fog computing environments further.

Middleware is necessary at the fog layer to improve reliability of applications

8.4 Comparative Analysis with Existing Middleware Solutions

The middleware framework proposed in this thesis addresses several limitations identified in existing fog middleware solutions (Table 2.2). This section briefly compares the proposed approach with existing solutions to highlight its contributions and acknowledge its limitations.

8.4.1 Key Differentiators

Unlike existing fog middleware that typically focus on specific functionalities or applications, the proposed framework provides comprehensive resource management through three distinguishing features:

Architectural Flexibility: While most existing solutions are designed for fixed architectures—such as the hierarchical approach in [129, 130] or the distributed approach in [114]—this middleware supports dynamic switching between hierarchical, clustered, and P2P architectures through the MAPE-K self-adaptation mechanism. This flexibility allows the system to adapt to changing environmental conditions rather than being constrained to a single architectural paradigm.

Cross-Domain Resource Sharing: Unlike middleware solutions that limit resource sharing within single domains [129, 130] or require prior knowledge for mobility support [138], the proposed framework enables transparent inter-cluster and cross-domain resource sharing without predetermined mobility patterns. This capability significantly expands the available resource pool and improves service availability.

Generic Application Support: While several existing solutions target specific applicationssuch as greenhouse IoT [35], healthcare privacy [133], or seismography [114]—this middleware provides a generic framework that supports diverse applications through its services layer, making it suitable for various fog computing scenarios.

8.4.2 Limitations and Trade-offs

Despite these advantages, the proposed middleware has certain limitations compared to specialised solutions:

Security Overhead: Security-focused middleware like [132] and privacy-specific solutions like [133] provide more robust security features than the current implementation. Applications requiring stringent security guarantees may benefit more from these specialised solutions until security features are fully integrated into the framework (as outlined in Future Directions).

Mobile Edge Computing Scenarios: Middleware designed specifically for Mobile Edge Computing [125–128] may perform better in scenarios with highly mobile end-devices like smartphones, as they are optimised for mobile-specific challenges. The proposed middleware, while supporting mobility, is primarily designed for IoT and stationary fog node scenarios.

Complexity vs. Simplicity: The adaptive nature of the proposed middleware introduces additional complexity compared to single-purpose solutions. For simple, static deployments with predictable workloads, lighter-weight middleware focusing on specific tasks (e.g., data filtering [35]) may be more efficient due to lower overhead.

8.4.3 Optimal Use Scenarios

The proposed middleware is particularly well-suited for:

- Dynamic environments where fog node availability fluctuates
- Applications requiring resource sharing across administrative domains
- Scenarios demanding architectural flexibility based on workload characteristics
- Large-scale deployments where centralised and decentralised approaches must coexist

Conversely, existing specialised middleware may be preferable for:

- Single-application deployments with well-defined requirements
- Highly secure or privacy-sensitive applications requiring domain-specific compliance
- Pure mobile edge computing scenarios with predominantly smartphone clients
- Small-scale deployments where architectural adaptability is unnecessary

This comparative analysis underscores that while the proposed middleware provides a comprehensive and adaptive solution for fog resource management, the choice of middleware should ultimately align with specific application requirements and deployment constraints.

8.5 Future Directions

While the proposed middleware framework addresses the identified resource management gaps in fog computing, several issues remain open and require future research. Future work in this domain should explore the following areas:

Evaluation in a Testbed or Real-World System

The effectiveness of the proposed framework has been demonstrated through simulation-based evaluations. However, real-world deployment and validation on a physical testbed or live fog computing system would provide deeper insights into its practical applicability, performance, and scalability under real-world constraints. Future studies should consider deploying the middleware in diverse environments, such as smart cities, industrial IoT, or vehicular fog computing, to assess its robustness in dynamic and heterogeneous settings.

Convergence with Emerging Technologies

Fog computing continues to evolve alongside emerging technologies such as Software-Defined Networking (SDN), Network Function Virtualisation (NFV), blockchain, federated learning, and quantum computing. Integrating fog resource management frameworks with these technologies introduces new opportunities and challenges. Future research should explore how fog middleware can adapt to these advancements to ensure interoperability, efficient resource allocation, and enhanced security while leveraging the capabilities of next-generation computing paradigms.

Extending the Peer-to-Peer Model

The peer-to-peer (P2P) model in fog computing enables users to contribute computational, storage, and networking resources in exchange for access to shared resources. However, a key challenge is mitigating the dominance of free-riding peers—nodes that consume resources without proportionally contributing to the system [202]. This problem is particularly interesting for dynamic environments such as vehicular fog computing, where nodes exhibit high mobility and unpredictable availability.

A promising future direction involves extending the P2P model to incorporate incentivisation mechanisms that encourage active participation and discourage resource exploitation. This could involve integrating economic models such as token-based incentive systems or credit-based reputation mechanisms [203]. Additionally, convergence with volunteer computing paradigms [22] — where users offer computational resources in return for rewards or recognition — could further strengthen the sustainability of resource sharing in fog networks.

Machine Learning for Intelligent Resource Management

The adaptive capabilities of the proposed middleware could be improved through the integration of machine learning techniques. While the current MAPE-K framework provides rule-based adaptation, machine learning could enable more sophisticated and predictive resource management strategies.

Future research should explore several ML-driven improvements:

Predictive Resource Allocation: Machine learning algorithms could analyse historical patterns of resource usage, application demands, and node availability to predict future resource requirements. This would enable proactive resource allocation rather than reactive responses, potentially reducing response times and improving overall system efficiency. Time-series forecasting models could predict when specific fog nodes are likely to become overloaded or unavailable, allowing the middleware to redistribute loads pre-emptively.

Intelligent Architecture Selection: Instead of using predefined rules for switching between hierarchical, clustered, and P2P architectures, machine learning classifiers could learn optimal architecture selection based on current network conditions, application characteristics, and historical performance data. Reinforcement learning approaches could continuously improve architecture selection decisions based on observed outcomes.

Anomaly Detection and Self-Healing: ML-based anomaly-detection could identify unusual patterns in fog node behaviour, potentially indicating failures, security threats, or performance degradation. This would enable the middleware to implement self-healing mechanisms before critical failures occur, improving system reliability beyond current threshold-based monitoring approaches.

Dynamic QoS Optimisation: Machine learning could optimise Quality of Service parameters by learning the relationship between resource allocation decisions and application performance metrics. This would be particularly valuable for applications with varying QoS requirements, enabling the middleware to make more nuanced trade-offs between competing demands.

The integration of machine learning presents challenges including the computational overhead of running ML models on resource-constrained fog nodes and the need for sufficient training data in dynamic fog environments. Future work should investigate lightweight ML approaches suitable for fog computing contexts and distributed learning techniques that preserve privacy while enabling collaborative intelligence across fog domains.

Security Features in Middleware

Security remains a critical concern in fog computing, particularly regarding the trustworthiness of fog nodes and protection against malicious actors. Effective resource management must incorporate mechanisms to ensure data integrity, access control, and secure communication across fog nodes. Future extensions of middleware should focus on:

- Trust and reputation mechanisms to assess and verify node credibility dynamically.
- Lightweight encryption and authentication techniques tailored for resource-constrained fog nodes.
- Resilience against cyber threats, such as Distributed Denial-of-Service (DDoS) attacks, rogue node infiltration, and unauthorised data access.

As security is an essential aspect of fog computing, future middleware architectures must integrate adaptive security measures that evolve alongside emerging threat landscapes while maintaining efficiency and scalability.

It is a significant challenge to draw conclusive proof of the benefits of a framework or system solely based on the findings from experimental studies because, as discussed in chapter 3, the ideal scenario would be to evaluate a new approach within the real-world environment where the system it will be deployed. However, the findings from the simulation experiments and evaluation provide a strong basis to justify the usefulness of an adaptive middleware for fog computing.

CHAPTER 9

Conclusion

This research presents a comprehensive middleware framework that addresses key challenges in fog computing, including heterogeneity, resource sharing, and dynamic environment adaptability. By developing middleware-enabled clustered and P2P fog architectures, this work demonstrates how resource pooling and adaptive architectural selection improve system availability, reliability, and performance. Integrating the MAPE-K self-adaptation framework within the middleware ensures that fog systems can dynamically adjust to environmental changes, improving resource utilisation and maintaining service quality.

The evaluation results validate the effectiveness of middleware-enabled clustering and peer-to-peer approaches in reducing latency, balancing workloads, and enhancing application responsiveness. Furthermore, the study highlights the importance of inter-cluster resource sharing and adaptive scheduling mechanisms in sustaining fog computing performance. The development of a simulation tool for clustered and P2P fog architectures further contributes to the field by providing a platform for ongoing research and experimentation.

Future research should explore hybrid models that combine the strengths of multiple architectural approaches, integrating hierarchical, clustered, and P2P configurations to further optimise fog computing environments. Extending adaptive middleware capabilities to incorporate AI-driven predictive resource allocation strategies could enhance system responsiveness and efficiency. By addressing these areas, future work can build upon this research to further refine and advance fog computing paradigms.

9.1 Contributions

9.1.1 Middleware for Fog Computing

This research has demonstrated the importance of middleware as a key enabler of efficient resource management in fog computing. The proposed middleware platform addresses critical challenges such as heterogeneity, service discovery, resource sharing, and the dynamic nature of fog environments. The middleware ensures integrated interaction between fog nodes and improves overall system performance by providing a structured intermediary layer in multi-layered distributed system architectures. The study further highlights the role of middleware in enabling different architectural environmentshierarchical, clustered, and peer-to-peereach of which offers distinct advantages depending on application requirements and environmental conditions.

9.1.2 Middleware-enabled Clustered Fog Architecture

A significant contribution of this work is the development and evaluation of a clustered fog architecture, modelled after the Internets Autonomous System concept. This approach introduces interoperability across clusters and allows efficient load balancing and resource scheduling both within and across clusters. The findings demonstrate that inter- and intra-cluster resource management improves availability which increases reliability and reduces response time. Additionally, the research emphasises the role of fog node proximity, showing that factors such as inter-cluster distance, often overlooked in prior literature, play a role in load balancing strategies at the fog layer. The clustered fog architecture, enabled by middleware, effectively expands resource availability and improves application performance in distributed environments.

9.1.3 Frameworks and Algorithms for Resource Management among Fog Clusters

This research also presents novel frameworks and algorithms for resource discovery, load balancing, and request handling within clustered and peer-to-peer fog architectures. Unlike existing approaches, which focus on isolated resource management techniques, this study integrates these aspects within an adaptive middleware framework. The evaluation results highlight the benefits of middleware-enabled resource pooling, which facilitates efficient request distribution and minimises queuing delays. These findings offer a comprehensive understanding of resource management in fog computing and lay the foundation for future research on hybrid approaches that further optimise system efficiency.

9.1.4 Middleware-enabled Peer-to-Peer Fog Architecture

Beyond clustering, this research explores the potential of peer-to-peer (P2P) architectures for fog computing, managed through middleware. The P2P model offers significant advantages in terms of scalability and resource utilisation, particularly in highly dynamic environments such as vehicular fog computing. By enabling direct communication and resource sharing among fog nodes without relying on a hierarchical structure, the middleware-enabled P2P approach minimises bottlenecks, improves fault tolerance, and increases system responsiveness. The findings highlight the importance of adaptive middleware in managing decentralised fog environments, ensuring continuous service availability despite fluctuations in connectivity and resource availability.

9.1.5 Simulation Tool

To support further advancements in fog computing research, this work extends the FogNetSim++ simulation tool in OMNeT++ to include clustered and peer-to-peer environments. This contribution provides a platform for researchers to analyse and experiment with different fog computing architectures under various conditions. By enabling simulation-based evaluations of middleware-enabled architectures, this tool facilitates deeper insights into fog computing paradigms and supports the development of future resource management strategies.

9.2 Final Remarks

In conclusion, this thesis has contributed to the advancement of fog computing by proposing and evaluating a middleware-based approach to resource management. The findings highlight the benefits of middleware in improving availability, reliability, adaptiveness, and performance in fog environments. By integrating clustering and peer-to-peer architectures, developing novel resource management frameworks, and extending simulation tools, this research provides a comprehensive foundation for future innovations in fog computing. The proposed middleware framework paves the way for more resilient and efficient fog-based systems, ultimately improving the reliability and responsiveness of applications in distributed environments.

References

[1] M. Chui, M. Collins, and M. Patel, "The Internet of Things: Catching up to an accelerating opportunity," McKinsey Company, Tech. Rep., 2021. [Online]. Available: https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/iot-value-set-to-accelerate-through-2030-where-and-how-to-capture-it

[2] Statistica, "Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2023, with forecasts from 2024 to 2028," 2024. [Online]. Available: https://www.statista.com/statistics/871513/worldwide-data-created/

[3] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing - MCC '12*. New York, New York, USA: ACM Press, 2012, p. 13.

[4] M. Armbrust, I. Stoica, M. Zaharia, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, and A. Rabkin, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, p. 50, apr 2010.

[5] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing," EECS Department, University of California, Berkeley,

Tech. Rep. UCB/EECS-2009-28, feb 2009. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

[6] X. Li, X. Jiang, P. Garraghan, and Z. Wu, "Holistic energy and failure aware workload scheduling in Cloud datacenters," *Future Generation Computer Systems*, vol. 78, no. 1, pp. 887–900, 2018.

[7] C. Doctorow, "Big data: Welcome to the petacentre," *Nature*, vol. 455, no. 7209, pp. 16–21, sep 2008.

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

[9] P. Schulz, M. Matthe, H. Klessig, M. Simsek, G. Fettweis, J. Ansari, S. A. Ashraf, B. Almeroth, J. Voigt, I. Riedel, A. Puschmann, A. Mitschele-Thiel, M. Muller, T. Elste, and M. Windisch, "Latency Critical IoT Applications in 5G: Perspective on the Design of Radio Interface and Network Architecture," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 70–78, 2017.

[10] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog Computing: A Platform for Internet of Things and Analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments*, N. Bessis and C. Dobre, Eds. Springer, Cham, 2014, pp. 169–186.

[11] A. Chougule, V. Chamola, A. Sam, F. R. Yu, and B. Sikdar, "A Comprehensive Review on Limitations of Autonomous Driving and Its Impact on Accidents and Collisions," *IEEE Open Journal of Vehicular Technology*, vol. 5, no. October 2023, pp. 142–161, 2024.

[12] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric Computing," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 5, pp. 37–42, sep 2015.

[13] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, “Context Aware Computing for The Internet of Things: A Survey,” *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 414–454, 2014.

[14] W. Xu, H. Zhou, N. Cheng, F. Lyu, W. Shi, J. Chen, and X. Shen, “Internet of vehicles in big data era,” *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, pp. 19–35, 2018.

[15] J. Daskal, “The un-territoriality of data,” *Yale Law Journal*, vol. 125, no. 2, pp. 326–398, 2015.

[16] P. Hu, S. Dhelim, H. Ning, and T. Qiu, “Survey on fog computing: architecture, key technologies, applications and open issues,” *Journal of Network and Computer Applications*, vol. 98, pp. 27–42, nov 2017.

[17] A. Sadri, A. Rahmani, M. Saberikamarposhti, and M. Hosseinzadeh, “Fog data management: A vision, challenges, and future directions,” *Journal of Network and Computer Applications*, vol. 174, 2021.

[18] M. GhobaeiArani, A. Souri, F. Safara, and M. Norouzi, “An efficient task scheduling approach using mothflame optimization algorithm for cyber-physical system applications in fog computing,” *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 2, feb 2020.

[19] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, “A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges,” *IEEE Communications Surveys Tutorials*, vol. 20, no. 1, pp. 416–464, 2018.

[20] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, and R. Ranjan, “Fog Computing: Survey of Trends, Architectures, Requirements, and Research Directions,” *IEEE Access*, vol. 6, pp. 47 980–48 009, aug 2018.

[21] L. Bittencourt, R. Immich, R. Sakellariou, N. Fonseca, E. Madeira, M. Curado, L. Villas, L. DaSilva, C. Lee, and O. Rana, “The Internet of Things,

Fog and Cloud continuum: Integration and challenges," *Internet of Things*, vol. 3-4, pp. 134–155, oct 2018.

[22] T. M. Mengistu and D. Che, "Survey and Taxonomy of Volunteer Computing," *ACM Computing Surveys*, vol. 52, no. 3, pp. 1–35, may 2020.

[23] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, "IoT Middleware: A Survey on Issues and Enabling Technologies," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 1–20, 2017.

[24] A. Farahzadi, P. Shams, J. Rezazadeh, and R. Farahbakhsh, "Middleware technologies for cloud of things: a survey," *Digital Communications and Networks*, vol. 4, no. 3, pp. 176–188, 2018.

[25] H. Samwini, M. Awais, and E. Pereira, "Middleware for Resource Sharing in Fog Computing with IoT Applications," in *IEEE EUROCON 2023 - 20th International Conference on Smart Technologies*, no. May. IEEE, 2023, pp. 508–513.

[26] H. Samwini, M. Raza, E. Pereira, U. Khan, and M. Awais, "Critical analysis of resource sharing and optimization in fog clustering," in *2024 International Conference on Emerging Trends in Smart Technologies (ICETST).* Karachi, Pakistan,: IEEE, oct 2024, pp. 1–6.

[27] K. Ashton and Others, "That internet of things' thing," *RFID journal*, vol. 22, no. 7, pp. 97–114, 2009.

[28] International Telecommunication Union (ITU), "ITU Internet Report 2005: Internet of Things," International Telecommunication Union (ITU), Geneva, Switzerland, Tech. Rep., nov 2015. [Online]. Available: www.itu.int/publications/bookshop/.

[29] S. B. Nath, H. Gupta, S. Chakraborty, and S. K. Ghosh, "A Survey of Fog Computing and Communication: Current Researches and Future Directions," *arXiv*, no. April, apr 2018. [Online]. Available: http://arxiv.org/abs/1804.04365

[30] I. Bose and R. Pal, “Auto-ID: managing anything, anywhere, anytime in the supply chain,” *Communications of the ACM*, vol. 48, no. 8, pp. 100–106, aug 2005.

[31] R.-I. Ciobanu, V. Cristea, C. Dobre, and F. Pop, “Big Data Platforms for the Internet of Things,” in *Big Data and Internet of Things: A Roadmap for Smart Environments*, N. Bessis and C. Dobre, Eds. Springer, Cham, 2014.

[32] R. Istepanian, A. Sungoor, A. Faisal, and N. Philip, “Internet of M-health Things 'm-IOT',” *IET Seminar on Assisted Living 2011*, 2011.

[33] Xi Chen, Jianming Liu, Xiangzhen Li, Limin Sun, and Yan Zhen, “Integration of IOT with smart grid,” in *IET International Conference on Communication Technology and Application (ICCTA 2011)*, no. 1. IET, 2011, pp. 723–726.

[34] S. Muthuramalingam, A. Bharathi, S. Rakesh kumar, N. Gayathri, R. Sathiyaraj, and B. Balamurugan, “IoT Based Intelligent Transportation System (IoT-ITS) for Global Perspective: A Case Study,” in *Internet of Things and Big Data Analytics for Smart Generation*, V. E. Balas, V. K. Solanki, R. Kumar, and M. Khari, Eds., 2019, ch. 13, pp. 279–300.

[35] A. Ordonez-Garcia, E. V. Nunez, M. Siller, and M. G. S. Cervantes, “IoT system for agriculture: Web technologies in real time with the Middleware paradigm,” in *2018 IEEE International Autumn Meeting on Power, Electronics and Computing (ROPEC).* IEEE, nov 2018, pp. 1–4.

[36] S. Misra, C. Roy, and A. Mukherjee, *Introduction to Industrial Internet of Things and Industry 4.0.* CRC Press, jan 2021.

[37] M. Aazam, S. Zeadally, and K. A. Harras, “Deploying Fog Computing in Industrial Internet of Things and Industry 4.0,” *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4674–4682, oct 2018.

[38] D. Evans, “The Internet of Things - How the Next Evolution of the Internet Is Changing Everything,” Cisco Internet Business Solutions Group (IBSG), San Jose, Tech. Rep., apr 2011.

[39] J. Landt, “The history of RFID,” *IEEE Potentials*, vol. 24, no. 4, pp. 8–11, oct 2005.

[40] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, oct 2010.

[41] R. Yugha and S. Chithra, “A survey on technologies and security protocols: Reference for future generation IoT,” *Journal of Network and Computer Applications*, vol. 169, no. June, p. 102763, nov 2020.

[42] Telecommunication Standardization Sector of ITU, “Overview of the Internet of Things,” Geneva, Switzerland, 2012.

[43] L. D. Xu, W. He, and S. Li, “Internet of Things in Industries: A Survey,” *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, nov 2014.

[44] E. Borgia, “The Internet of Things vision: Key features, applications and open issues,” *Computer Communications*, vol. 54, pp. 1–31, dec 2014.

[45] F. J. Corbató and V. A. Vyssotsky, “Introduction and overview of the multics system,” in *Proceedings of the November 30–December 1, 1965, fall joint computer conference, part I on XX - AFIPS ’65 (Fall, part I)*. New York, New York, USA: ACM Press, nov 1965, p. 185.

[46] Amazon, “Announcing Amazon Elastic Compute Cloud (Amazon EC2) - beta,” aug 2006. [Online]. Available: https://aws.amazon.com/about-aws/whats-new/2006/08/24/announcing-amazon-elastic-compute-cloud-amazon-ec2---beta/

[47] G. Piro, M. Amadeo, G. Boggia, C. Campolo, L. A. Grieco, A. Molinaro, and G. Ruggeri, “Gazing into the Crystal Ball: When the Future Internet

Meets the Mobile Clouds," *IEEE Transactions on Cloud Computing*, vol. 7, no. 1, pp. 210–223, jan 2019.

[48] P. M. Mell and T. Grance, "SP 800-145. The NIST Definition of Cloud Computing," National Institute of Standards Technology, Gaithersburg, MD, United States, Tech. Rep., 2011. [Online]. Available: https://dl.acm.org/citation.cfm?id=2206223

[49] S. S. Chauhan, E. S. Pilli, R. Joshi, G. Singh, and M. Govil, "Brokering in interconnected cloud computing environments: A survey," *Journal of Parallel and Distributed Computing*, vol. 133, pp. 193–209, nov 2019.

[50] A. Barnawi, S. Sakr, W. Xiao, and A. Al-Barakati, "The views, measurements and challenges of elasticity in the cloud: A review," *Computer Communications*, vol. 154, pp. 111–117, mar 2020.

[51] A. Fox, "The Potential of Cloud Computing : Opportunities and Challenges Cost Associativity and Elasticity," in *2010 US Frontiers of Engineering Symposium*, 2010, pp. 1–7.

[52] A. A. Khan and M. Zakarya, "Energy, performance and cost efficient cloud datacentres: A survey," *Computer Science Review*, vol. 40, p. 100390, may 2021.

[53] A. Singh and K. Chatterjee, "Cloud security issues and challenges: A survey," *Journal of Network and Computer Applications*, vol. 79, pp. 88–115, feb 2017.

[54] S. S. Gill, R. C. Arya, G. S. Wander, and R. Buyya, "Fog-Based Smart Healthcare as a Big Data and Cloud Service for Heart Patients Using IoT," in *Lecture Notes on Data Engineering and Communications Technologies.* Springer, 2019, vol. 26, pp. 1376–1383.

[55] N. Dhingra, "Challenges, Limitation and security issues on mobile computing," *International journal of current engineering and technology*, vol. 4, pp. 3459–3462, 2014.

[56] M. Satyanarayanan, “Pervasive computing: vision and challenges,” *IEEE Personal Communications*, vol. 8, no. 4, pp. 10–17, 2001.

[57] P. Mach and Z. Becvar, “Mobile Edge Computing: A Survey on Architecture and Computation Offloading,” *IEEE Communications Surveys and Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[58] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, and O. Rana, “Fog Computing for the Internet of Things,” *ACM Transactions on Internet Technology*, vol. 19, no. 2, pp. 1–41, may 2019.

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

[60] S. Antipolis, “ETSI Multi-access Edge Computing starts second phase and renews leadership team,” 2017. [Online]. Available: https://www.etsi.org/newsroom/news/1180-2017-03-news-etsi-multi-access-edge-computing-starts-second-phase-and-renews-lead

[61] E. Baccarelli, P. G. V. Naranjo, M. Scarpiniti, M. Shojafar, and J. H. Abawajy, “Fog of Everything: Energy-Efficient Networked Computing Architectures, Research Challenges, and a Case Study,” *IEEE Access*, vol. 5, pp. 9882–9910, 2017.

[62] N. Wang, B. Varghese, M. Matthaiou, and D. S. Nikolopoulos, “ENORM: A Framework For Edge NOde Resource Management,” *IEEE Transactions on Services Computing*, vol. 46, no. November, pp. 1–1, 2019.

[63] IEEE Communications Society, *IEEE Std 1934-2018 : IEEE Standard for Adoption of OpenFog Reference Architecture for Fog Computing.*, 1st ed., J. K. Zao, T. Zhang, and J. Zhou, Eds. New York: IEEE, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8423800

[64] P. Habibi, M. Farhoudi, S. Kazemian, S. Khorsandi, and A. Leon-Garcia, "Fog Computing: A Comprehensive Architectural Survey," *IEEE Access*, vol. 8, pp. 69 105–69 133, 2020.

[65] M. Iorga, L. Feldman, R. Barton, M. J. Martin, N. Goren, and C. Mahmoudi, "Fog computing conceptual model," National Institute of Standards and Technology, Gaithersburg, MD, Tech. Rep., mar 2018.

[66] S. P. Singh, A. Nayyar, R. Kumar, and A. Sharma, "Fog computing: from architecture to edge computing and big data processing," *The Journal of Supercomputing*, vol. 75, no. 4, pp. 2070–2105, apr 2019.

[67] S. Aggarwal and N. Kumar, "Fog Computing for 5G-Enabled Tactile Internet: Research Issues, Challenges, and Future Research Directions," *Mobile Networks and Applications*, vol. 28, no. 2, pp. 690–717, apr 2023.

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

[69] Z. Rejiba, X. Masip-Bruin, and E. Marín-Tordera, "A Survey on Mobility-Induced Service Migration in the Fog, Edge, and Related Computing Paradigms," *ACM Computing Surveys*, vol. 52, no. 5, pp. 1–33, sep 2020.

[70] F. Sharifi, A. Rasaii, A. Pasdar, S. Hessabi, and Y. C. Lee, "On the Effectiveness of Fog Offloading in a Mobility-Aware Healthcare Environment," *Digital*, vol. 3, no. 4, pp. 300–318, dec 2023.

[71] C. A. R. L. Brennand, F. D. da Cunha, G. Maia, E. Cerqueira, A. A. Loureiro, and L. A. Villas, "FOX: A traffic management system of computer-based vehicles FOG," in *2016 IEEE Symposium on Computers and Communication (ISCC)*. Messina, Italy: IEEE, jun 2016, pp. 982–987.

[72] Y. Lai, F. Yang, L. Zhang, and Z. Lin, "Distributed Public Vehicle System Based on Fog Nodes and Vehicular Sensing," *IEEE Access*, vol. 6, pp. 22 011–22 024, 2018.

[73] C. Huang, R. Lu, and K.-K. R. Choo, “Vehicular Fog Computing: Architecture, Use Case, and Security and Forensic Challenges,” *IEEE Communications Magazine*, vol. 55, no. 11, pp. 105–111, nov 2017.

[74] X. Vilajosana, P. Tuset, T. Watteyne, and K. Pister, “OpenMote: Open-Source Prototyping Platform for the Industrial IoT,” in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, 2015, vol. 155, no. October, pp. 211–222.

[75] G. Caiza, M. Saeteros, W. Oñate, and M. V. Garcia, “Fog computing at industrial level, architecture, latency, energy, and security: A review,” *Heliyon*, vol. 6, no. 4, p. e03706, apr 2020.

[76] P. O’Donovan, C. Gallagher, K. Bruton, and D. T. O’Sullivan, “A fog computing industrial cyber-physical system for embedded low-latency machine learning Industry 4.0 applications,” *Manufacturing Letters*, vol. 15, pp. 139–142, jan 2018.

[77] M. Dabrowska, “Power of predictive maintenance with IoT: Reducing downtime and costs,” 2024.

[78] Y. K. Teoh, S. S. Gill, and A. K. Parlikad, “IoT and Fog-Computing-Based Predictive Maintenance Model for Effective Asset Management in Industry 4.0 Using Machine Learning,” *IEEE Internet of Things Journal*, vol. 10, no. 3, pp. 2087–2094, feb 2023.

[79] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, and C. Chen, “Data-Driven Intelligent Transportation Systems: A Survey,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1624–1639, dec 2011.

[80] T. S. J. Darwish and K. Abu Bakar, “Fog Based Intelligent Transportation Big Data Analytics in The Internet of Vehicles Environment: Motivations, Architecture, Challenges, and Critical Issues,” *IEEE Access*, vol. 6, pp. 15 679–15 701, 2018.

[81] BBC, "UK must tackle 'astonishing' cost of congestion," 2018. [Online]. Available: https://www.bbc.co.uk/news/uk-42948259

[82] EU-Mobility and Transport, "Transport in the European Union Current Trends and Issues," European Commission, Brussels, Tech. Rep., 2018. [Online]. Available: https://ec.europa.eu/transport/sites/transport/files/2018-transport-in-the-eu-current-trends-and-issues.pdf

[83] S. Eichler, "Performance Evaluation of the IEEE 802.11p WAVE Communication Standard," in *2007 IEEE 66th Vehicular Technology Conference*. Baltimore, MD, USA: IEEE, sep 2007, pp. 2199–2203.

[84] O. Kaiwartya, A. H. Abdullah, Y. Cao, A. Altameem, M. Prasad, C.-T. Lin, and X. Liu, "Internet of Vehicles: Motivation, Layered Architecture, Network Model, Challenges, and Future Aspects," *IEEE Access*, vol. 4, pp. 5356–5373, 2016.

[85] M. Sookhak, F. R. Yu, Y. He, H. Talebian, N. Sohrabi Safa, N. Zhao, M. K. Khan, and N. Kumar, "Fog Vehicular Computing: Augmentation of Fog Computing Using Vehicular Cloud Computing," *IEEE Vehicular Technology Magazine*, vol. 12, no. 3, pp. 55–64, sep 2017.

[86] Z. Ning, J. Huang, and X. Wang, "Vehicular Fog Computing: Enabling Real-Time Traffic Management for Smart Cities," *IEEE Wireless Communications*, vol. 26, no. 1, pp. 87–93, feb 2019.

[87] A. J. V. Neto, Z. Zhao, J. J. P. C. Rodrigues, H. B. Camboim, and T. Braun, "Fog-Based Crime-Assistance in Smart IoT Transportation System," *IEEE Access*, vol. 6, pp. 11 101–11 111, 2018.

[88] E. E. Ali, L. Chew, and K. Y.-L. Yap, "Evolution and current status of mhealth research: a systematic review," *BMJ Innovations*, vol. 2, no. 1, pp. 33–40, jan 2016.

[89] B. T. Mi, X. Liang, and S. S. Zhang, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communication Surveys and Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2018.

[90] ETSI, "TR 102 732 - V1.1.1 - Machine-to-Machine Communications (M2M); Use Cases of M2M applications for eHealth," European Telecommunications Standards Institute, Tech. Rep., sep 2013. [Online]. Available: http://portal.etsi.org/chaircor/ETSI_support.asp

[91] B. Woodward, *MHealth: Fundamentals and Applications*, R. S. Istepanian, Ed. Hoboken, NJ, USA: Wiley, nov 2016.

[92] X. Wu, R. Dunne, Z. Yu, and W. Shi, "STREMS: A Smart Real-Time Solution toward Enhancing EMS Prehospital Quality," in *2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE)*. IEEE, jul 2017, pp. 365–372.

[93] E. Karavatselou, M.-A. Fengou, G. Mantas, and D. Lymberopoulos, "Profile Management System in Ubiquitous Healthcare Cloud Computing Environment," in *Broadband Communications, Networks, and Systems*, V. Sucasas, G. Mantas, and S. Althunibat, Eds. Cham: Springer International Publishing, 2019, pp. 105–114.

[94] Z. Á. Mann, "Notions of architecture in fog computing," *Computing*, vol. 103, no. 1, pp. 51–73, jan 2021.

[95] J. Wen, C. Ren, and A. K. Sangaiah, "Energy-Efficient Device-to-Device Edge Computing Network: An Approach Offloading Both Traffic and Computation," *IEEE Communications Magazine*, vol. 56, no. 9, pp. 96–102, 2018.

[96] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, and Q. Yang, "A hierarchical distributed fog computing architecture for big data analysis in smart cities," in *ACM International Conference Proceeding Series*, vol. 07-09-Ocob. Association for Computing Machinery, oct 2015.

[97] A. V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya, "Fog Computing: Principles, Architectures, and Applications," in *Internet of Things: Principles and Paradigms*, R. Buyya and A. V. Dastjerdi, Eds. Cambridge, Massachusetts, USA: Elsevier Inc., jan 2016, ch. 4, pp. 61–78.

[98] W. Zhang, Z. Zhang, and H.-C. Chao, "Cooperative Fog Computing for Dealing with Big Data in the Internet of Vehicles: Architecture and Hierarchical Resource Management," *IEEE Communications Magazine*, vol. 55, no. 12, pp. 60–67, dec 2017.

[99] V. Stantchev, A. Barnawi, S. Ghulam, J. Schubert, and G. Tamm, "Smart Items, Fog and Cloud Computing as Enablers of Servitization in Healthcare," *Sensors Transducers*, vol. 185, no. 2, pp. 121–128, 2015.

[100] V. Kumar and R. Vidhyalakshmi, "Cloud Reliability," in *Reliability Aspect of Cloud Computing Environment.* Singapore: Springer Singapore, 2018, pp. 29–49. [Online]. Available: http://link.springer.com/10.1007/978-981-13-3023-0_2

[101] B. Treynor, M. Dahlin, V. Rau, and B. Beyer, "The calculus of service availability," *Communications of the ACM*, vol. 60, no. 9, pp. 42–47, aug 2017.

[102] Y. Izrailevsky and C. Bell, "Cloud Reliability," *IEEE Cloud Computing*, vol. 5, no. 3, pp. 39–44, 2018.

[103] Y.-s. Dai, B. Yang, J. Dongarra, and G. Zhang, "Cloud Service Reliability : Modeling and Analysis," 2010, pp. 1–17.

[104] Q. Zhu, R. Wang, Q. Chen, Y. Liu, and W. Qin, "IOT gateway: Bridging wireless sensor networks into Internet of Things," *Proceedings - IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, EUC 2010*, pp. 347–352, 2010.

[105] M. Aazam, P. P. Hung, and E.-N. Huh, "Smart gateway based communication for cloud of things," in *2014 IEEE Ninth International Conference on*

Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP). Barcelona, Spain: IEEE, apr 2014, pp. 1–6.

[106] M. Aazam and E.-N. Huh, "Fog Computing and Smart Gateway Based Communication for Cloud of Things," in *2014 International Conference on Future Internet of Things and Cloud.* Barcelona, Spain: IEEE, aug 2014, pp. 464–470.

[107] M. Aazam, I. Khan, A. A. Alsaffar, and E.-N. Huh, "Cloud of Things: Integrating Internet of Things and cloud computing and the issues involved," in *Proceedings of 2014 11th International Bhurban Conference on Applied Sciences Technology (IBCAST) Islamabad, Pakistan, 14th - 18th January, 2014.* IEEE, jan 2014, pp. 414–419.

[108] A. M. Rahmani, T. N. Gia, B. Negash, A. Anzanpour, I. Azimi, M. Jiang, and P. Liljeberg, "Exploiting smart e-Health gateways at the edge of healthcare Internet-of-Things: A fog computing approach," *Future Generation Computer Systems*, vol. 78, pp. 641–658, jan 2018.

[109] A. Carrega, M. Repetto, P. Gouvas, and A. Zafeiropoulos, "A Middleware for Mobile Edge Computing," *IEEE Cloud Computing*, vol. 4, no. 4, pp. 26–37, jul 2017.

[110] S. Nastic, S. Sehic, D.-H. Le, H.-L. Truong, and S. Dustdar, "Provisioning Software-Defined IoT Cloud Systems," in *2014 International Conference on Future Internet of Things and Cloud.* IEEE, aug 2014, pp. 288–295.

[111] M. Vogler, J. Schleicher, C. Inzinger, S. Nastic, S. Sehic, and S. Dustdar, "LEONORE – Large-Scale Provisioning of Resource-Constrained IoT Deployments," in *2015 IEEE Symposium on Service-Oriented System Engineering*, vol. 30. IEEE, mar 2015, pp. 78–87.

[112] S. Nastic, H.-L. Truong, and S. Dustdar, "A Middleware Infrastructure for Utility-Based Provisioning of IoT Cloud Systems," in *2016 IEEE/ACM Symposium on Edge Computing (SEC).* IEEE, oct 2016, pp. 28–40.

[113] C.-H. Hong and B. Varghese, “Resource Management in Fog/Edge Computing: A Survey on Architectures, Infrastructure, and Algorithms,” *ACM Computing Surveys*, vol. 52, no. 5, pp. 1–37, 2019.

[114] J. Clemente, M. Valero, J. Mohammadpour, X. Li, and W. Song, “Fog computing middleware for distributed cooperative data analytics,” in *2017 IEEE Fog World Congress (FWC).* IEEE, oct 2017, pp. 1–6.

[115] S. Agarwal, S. Yadav, and A. K. Yadav, “An Efficient Architecture and Algorithm for Resource Provisioning in Fog Computing,” *International Journal of Information Engineering and Electronic Business*, vol. 8, no. 1, pp. 48–61, jan 2016.

[116] D. Kimovski, H. Ijaz, N. Saurabh, and R. Prodan, “Adaptive Nature-Inspired Fog Architecture,” in *2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC).* IEEE, may 2018, pp. 1–8.

[117] A. Brogi and S. Forti, “QoS-Aware Deployment of IoT Applications Through the Fog,” *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1185–1192, oct 2017.

[118] L. Chen, J. Wu, X. Long, and Z. Zhang, “ENGINE:Cost Effective Offloading in Mobile Edge Computing with Fog-Cloud Cooperation,” *arXiv*, nov 2017. [Online]. Available: http://arxiv.org/abs/1711.01683

[119] R. Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya, “Quality of Experience (QoE)-aware placement of applications in Fog computing environments,” *Journal of Parallel and Distributed Computing*, vol. 132, pp. 190–203, oct 2019.

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

[121] Y. F. Chen, D. H. Huang, C. F. Huang, and Y. K. Lin, "Reliability Evaluation for a Cloud Computer Network with Fog Computing," *Proceedings - Companion of the 2020 IEEE 20th International Conference on Software Quality, Reliability, and Security, QRS-C 2020*, pp. 682–683, 2020.

[122] F. Popentiu-Vladicescu and G. Albeanu, "Software reliability in the fog computing," in *2017 International Conference on Innovations in Electrical Engineering and Computational Technologies (ICIEECT).* IEEE, apr 2017, pp. 1–4.

[123] K. Dantu, S. Y. Ko, and L. Ziarek, "RAINA: Reliability and Adaptability in Android for Fog Computing," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 41–45, apr 2017.

[124] M. Aazam and E.-N. Huh, "Fog Computing: The Cloud-IoT/IoE Middleware Paradigm," *IEEE Potentials*, vol. 35, no. 3, pp. 40–44, may 2016.

[125] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt, "Adaptive Application Configuration and Distribution in Mobile Cloudlet Middleware," in *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering.* Springer, Berlin, Heidelberg, nov 2013, vol. 65 LNICST, pp. 178–191.

[126] J. Rodrigues, E. R. B. Marques, L. M. B. Lopes, and F. Silva, "Towards a middleware for mobile edge-cloud applications," in *Proceedings of the 2nd Workshop on Middleware for Edge Clouds Cloudlets - MECC '17*, vol. 6. New York, New York, USA: ACM Press, dec 2017, pp. 1–6.

[127] G. Orsini, D. Bade, and W. Lamersdorf, "Computing at the Mobile Edge: Designing Elastic Android Applications for Computation Offloading," in *2015 8th IFIP Wireless and Mobile Networking Conference (WMNC).* IEEE, oct 2015, pp. 112–119.

[128] ——, "CloudAware: A Context-Adaptive Middleware for Mobile Edge and Cloud Computing Applications," in *2016 IEEE 1st International Workshops*

on Foundations and Applications of Self Systems (FAS*W).* IEEE, sep 2016, pp. 216–221.

[129] N. Mohamed, J. Al-Jaroodi, I. Jawhar, S. Lazarova-Molnar, and S. Mahmoud, “SmartCityWare: A Service-Oriented Middleware for Cloud and Fog Enabled Smart City Services,” *IEEE Access*, vol. 5, pp. 17 576–17 588, 2017.

[130] N. Mohamed, S. Lazarova-Molnar, I. Jawhar, and J. Al-Jaroodi, “Towards Service-Oriented Middleware for Fog and Cloud Integrated Cyber Physical Systems,” in *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW).* IEEE, jun 2017, pp. 67–74.

[131] P. Bellavista, J. Berrocal, A. Corradi, S. K. Das, L. Foschini, and A. Zanni, “A survey on fog computing for the Internet of Things,” *Pervasive and Mobile Computing*, vol. 52, pp. 71–99, jan 2019.

[132] B. Mukherjee, S. Wang, W. Lu, R. L. Neupane, D. Dunn, Y. Ren, Q. Su, and P. Calyam, “Flexible IoT security middleware for end-to-end cloudfog communication,” *Future Generation Computer Systems*, vol. 87, pp. 688–703, oct 2018.

[133] A. M. Elmisery, S. Rho, and D. Botvich, “A Fog Based Middleware for Automated Compliance With OECD Privacy Principles in Internet of Healthcare Things,” *IEEE Access*, vol. 4, pp. 8418–8441, 2016.

[134] Y. Nakamura, H. Suwa, Y. Arakawa, H. Yamaguchi, and K. Yasumoto, “Design and Implementation of Middleware for IoT Devices toward Real-Time Flow Processing,” in *2016 IEEE 36th International Conference on Distributed Computing Systems Workshops (ICDCSW).* IEEE, jun 2016, pp. 162–167.

[135] P. Bellavista, L. Foschini, N. Ghiselli, and A. Reale, “MQTT-based Middleware for Container Support in Fog Computing Environments,” in *2019 IEEE Symposium on Computers and Communications (ISCC).* IEEE, jun 2019, pp. 1–7.

[136] M. Pore, V. Chakati, A. Banerjee, and S. K. S. Gupta, "Middleware for Fog and Edge Computing: Design Issues," in *Fog and Edge Computing: Principles and Paradigms*, R. Buyya and N. S. Srirama, Eds. Hoboken, NJ, USA: John Wiley Sons, Inc., jan 2019, pp. 123–144.

[137] J. Al-Jaroodi, N. Mohamed, I. Jawhar, and S. Mahmoud, "CoTWare: A Cloud of Things Middleware," in *2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW)*. IEEE, jun 2017, pp. 214–219.

[138] S. Shekhar, A. Chhokra, H. Sun, A. Gokhale, A. Dubey, and X. Koutsoukos, "URMILA: A Performance and Mobility-Aware Fog/Edge Resource Management Middleware," in *2019 IEEE 22nd International Symposium on Real-Time Distributed Computing (ISORC)*. Valencia, Spain: IEEE, may 2019, pp. 118–125. [Online]. Available: https://ieeexplore.ieee.org/document/8759356/

[139] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, *Distributed Systems: Concepts and Design*, 5th ed. USA: Addison-Wesley Publishing Company, 2011.

[140] B. Varghese and R. Buyya, "Next generation cloud computing: New trends and research directions," *Future Generation Computer Systems*, vol. 79, pp. 849–861, 2018.

[141] Q. Xu and J. Zhang, "PiFogBed: A Fog Computing Testbed Based on Raspberry Pi," *2019 IEEE 38th International Performance Computing and Communications Conference, IPCCC 2019*, 2019.

[142] International Organization for Standardization, *International Standard ISO/IEC/ IEEE 24765*. ISO/IEC/ IEEE, 2017, vol. 2017. [Online]. Available: https://www.iso.org/standard/71952.html

[143] M. Fahimullah, G. Philippe, S. Ahvar, and M. Trocan, "Simulation Tools for Fog Computing: A Comparative Analysis," *Sensors*, vol. 23, no. 7, p. 3492, mar 2023.

[144] R. Jain, *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling.* john wiley sons, 1990.

[145] J. Singh, P. Singh, and S. S. Gill, "Fog computing: A taxonomy, systematic review, current trends and research challenges," *Journal of Parallel and Distributed Computing*, vol. 157, pp. 56–85, nov 2021. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0743731521001349

[146] M. Ghobaei-Arani, A. Souri, and A. A. Rahmanian, "Resource Management Approaches in Fog Computing: a Comprehensive Review," *Journal of Grid Computing*, vol. 18, no. 1, pp. 1–42, mar 2020.

[147] M. Barcellos, G. Facchini, and H. Muhammad, "Bridging the Gap between Simulation and Experimental Evaluation in Computer Networks," in *39th Annual Simulation Symposium (ANSS'06)*, vol. 2006. IEEE, 2006, pp. 286–293.

[148] S. Fernandes, *Performance Evaluation for Network Services, Systems and Protocols.* Cham: Springer International Publishing, 2017.

[149] A. A. T. R. Coutinho, E. O. Carneiro, and F. Greve, "Simulation and Modeling Tools for Fog Computing," in *Fog Computing: Concepts, Frameworks, and Applications.* Boca Raton: Chapman and Hall/CRC, may 2022, pp. 51–84.

[150] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, sep 2017.

[151] M. M. Lopes, W. A. Higashino, M. A. Capretz, and L. F. Bittencourt, "MyiFogSim," in *Companion Proceedings of the10th International Conference on Utility and Cloud Computing.* New York, NY, USA: ACM, dec 2017, pp. 47–52.

[152] T. Qayyum, A. W. Malik, M. A. Khan Khattak, O. Khalid, and S. U. Khan, "FogNetSim++: A Toolkit for Modeling and Simulation of Distributed Fog Environment," *IEEE Access*, vol. 6, pp. 63 570–63 583, 2018.

[153] I. Lera, C. Guerrero, and C. Juiz, "YAFS: A Simulator for IoT Scenarios in Fog Computing," *IEEE Access*, vol. 7, pp. 91 745–91 758, 2019.

[154] C. Puliafito, D. M. Gonçalves, M. M. Lopes, L. L. Martins, E. Madeira, E. Mingozzi, O. Rana, and L. F. Bittencourt, "MobFogSim: Simulation of mobility and migration for fog computing," *Simulation Modelling Practice and Theory*, vol. 101, no. December 2019, 2020.

[155] S. V. Margariti, V. V. Dimakopoulos, and G. Tsoumanis, "Modeling and simulation tools for fog computing-A comprehensive survey from a cost perspective," *Future Internet*, vol. 12, no. 5, 2020.

[156] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[157] S. J. Mullender, "Distributed Systems Management in Wide-Area Networks," in *NGI-SION 1984 Informatica Symposium 1984*. Nederlands Genootschap voor Informatica (NGI), apr 1984, pp. 415–424.

[158] T. Dey, S. Bera, A. Mukherjee, and D. De, "Resource Management in Distributed Computing," in *Resource Management in Distributed Systems*, A. Mukherjee, D. De, and R. Buyya, Eds. Springer, 2024, pp. 1–15. [Online]. Available: https://link.springer.com/10.1007/978-981-97-2644-8_1

[159] A. Goscinski and M. Bearman, "Resource management in large distributed systems," *Operating Systems Review (ACM)*, vol. 24, no. 4, pp. 7–25, 1990.

[160] I. Martinez, A. S. Hafid, and A. Jarray, "Design, Resource Management, and Evaluation of Fog Computing Systems: A Survey," *IEEE Internet of Things Journal*, vol. 8, no. 4, pp. 2494–2516, feb 2021.

[161] M. Kaur and R. Aron, "A systematic study of load balancing approaches in the fog computing environment," *The Journal of Supercomputing*, vol. 77, no. 8, pp. 9202–9247, aug 2021.

[162] M. Bendechache, S. Svorobej, P. Takako Endo, and T. Lynn, "Simulating Resource Management across the Cloud-to-Thing Continuum: A Survey and Future Directions," *Future Internet*, vol. 12, no. 6, p. 95, may 2020.

[163] A. Mijuskovic, A. Chiumento, R. Bemthuis, A. Aldea, and P. Havinga, "Resource Management Techniques for Cloud/Fog and Edge Computing: An Evaluation Framework and Classification," *Sensors*, vol. 21, no. 5, p. 1832, mar 2021.

[164] M. Fahimullah, S. Ahvar, and M. Trocan, "A Review of Resource Management in Fog Computing: Machine Learning Perspective," pp. 1–15, sep 2022.

[165] P. Kansal, M. Kumar, and O. P. Verma, "Classification of resource management approaches in fog/edge paradigm and future research prospects: a systematic review," *The Journal of Supercomputing*, vol. 78, no. 11, pp. 13 145–13 204, jul 2022.

[166] D. Alsadie, "Resource Management Strategies in Fog Computing Environment-A Comprehensive Review," *International Journal of Computer Science and Network Security*, vol. 22, no. 4, pp. 310–328, 2022.

[167] A. Mukherjee, D. De, and R. Buyya, "Cloud Computing Resource Management," in *Resource Management in Distributed Systems*, A. Mukherjee, D. De, and R. Buyya, Eds., 2024, pp. 17–37.

[168] M. R. Alizadeh, V. Khajehvand, A. M. Rahmani, and E. Akbari, "Task scheduling approaches in fog computing: A systematic review," *International Journal of Communication Systems*, vol. 33, no. 16, nov 2020.

[169] N. Alshammari, S. S. Gill, H. Pervaiz, Q. Ni, and H. Ahmed, "Resource Scheduling in Integrated IoT and Fog Computing Environments: A Taxonomy, Survey and Future Directions," in *Resource Management in Distributed Systems*, A. Mukherjee, D. De, and R. Buyya, Eds. Springer, 2024, pp. 63–77.

[170] A. Bukhari, F. K. Hussain, and O. K. Hussain, "Fog node discovery and selection: A Systematic literature review," *Future Generation Computer Systems*, vol. 135, pp. 114–128, 2022.

[171] S. K. Mishra, B. Sahoo, and P. P. Parida, "Load balancing in cloud computing: A big picture," *Journal of King Saud University - Computer and Information Sciences*, vol. 32, no. 2, pp. 149–158, feb 2020.

[172] M. H. Kashani and E. Mahdipour, "Load Balancing Algorithms in Fog Computing," *IEEE Transactions on Services Computing*, vol. 16, no. 2, pp. 1505–1521, 2023.

[173] B. Costa, J. Bachiega, L. R. de Carvalho, and A. P. F. Araujo, "Orchestration in Fog Computing: A Comprehensive Survey," *ACM Computing Surveys*, vol. 55, no. 2, pp. 1–34, feb 2023.

[174] J. Gedeon, S. Zengerle, S. Alles, F. Brandherm, and M. Muhlhauser, "Sunstone: Navigating the Way Through the Fog," in *2020 IEEE 4th International Conference on Fog and Edge Computing (ICFEC)*. IEEE, may 2020, pp. 49–58.

[175] G. S. Blair, G. Coulson, and P. Grace, "Research directions in reflective middleware," in *Proceedings of the 3rd workshop on Adaptive and reflective middleware -*. New York, New York, USA: ACM Press, 2004, pp. 262–267.

[176] F. Kon, F. Costa, G. Blair, and R. H. Campbell, "The Case for Reflective Middleware," *Communications of the ACM*, vol. 45, no. 6, pp. 33–38, 2002.

[177] D. Weyns, B. Schmerl, V. Grassi, S. Malek, R. Mirandola, C. Prehofer, J. Wuttke, J. Andersson, H. Giese, and K. M. Göschka, "On Patterns for

Decentralized Control in Self-Adaptive Systems," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2013, vol. 7475 LNCS, pp. 76–107.

[178] P. Arcaini, E. Riccobene, and P. Scandurra, "Modeling and Analyzing MAPE-K Feedback Loops for Self-Adaptation," in *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. IEEE, may 2015, pp. 13–23.

[179] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud Computing and Grid Computing 360-degree compared," *Grid Computing Environments Workshop, GCE 2008*, 2008.

[180] A. Bertrand, "Distributed Signal Processing for Wireless EEG Sensor Networks," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 23, no. 6, pp. 923–935, 2015.

[181] L. Hellstrom-Westas, I. Rosen, and N. W. Svenningsen, "Predictive value of early continuous amplitude integrated EEG recordings on outcome after severe birth asphyxia in full term infants." *Archives of Disease in Childhood - Fetal and Neonatal Edition*, vol. 72, no. 1, pp. F34–F38, jan 1995. [Online]. Available: https://fn.bmj.com/lookup/doi/10.1136/fn.72.1.F34

[182] A. Delorme, "EEG is better left alone," *Scientific Reports*, vol. 13, no. 1, p. 2372, feb 2023.

[183] D. RAHBARI and M. NICKRAY, "Low-latency and energy-efficient scheduling in fog-based IoT applications," *TURKISH JOURNAL OF ELECTRICAL ENGINEERING COMPUTER SCIENCES*, vol. 27, no. 2, pp. 1406–1427, mar 2019. [Online]. Available: http://online.journals.tubitak.gov.tr/openDoiPdf.htm?mKodu=elk-1810-47

[184] M. Dadashi Gavaber and A. Rajabzadeh, "MFP: an approach to delay and energy-efficient module placement in IoT applications based on multi-fog," *Journal of Ambient Intelligence and Humanized Computing*,

vol. 12, no. 7, pp. 7965–7981, 2021. [Online]. Available: https://doi.org/10.1007/s12652-020-02525-7

[185] S. R. Hassan, I. Ahmad, S. Ahmad, A. Alfaify, and M. Shafiq, "Remote Pain Monitoring Using Fog Computing for e-Healthcare: An Efficient Architecture," *Sensors*, vol. 20, no. 22, p. 6574, nov 2020.

[186] S. Vanneste, J. de Hoog, T. Huybrechts, S. Bosmans, R. Eyckerman, M. Sharif, S. Mercelis, and P. Hellinckx, "Distributed uniform streaming framework: An elastic fog computing platform for event stream processing and platform transparency," *Future Internet*, vol. 11, no. 7, 2019.

[187] J. Oueis, E. C. Strinati, S. Sardellitti, and S. Barbarossa, "Small Cell Clustering for Efficient Distributed Fog Computing: A Multi-User Case," in *2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall)*, no. VOL. IEEE, sep 2015, pp. 1–5.

[188] V. B. Martins, D. D. J. de Macedo, L. Pioli, and R. Immich, "A Cluster Formation Algorithm for Fog Architectures Based on Mobility Parameters at a Geographically LAN Perspective," in *Lecture Notes in Networks and Systems*, 2023, vol. 571 LNNS, pp. 25–36.

[189] E. Ghaferi, R. Malekhosseini, F. Rad, and K. Bagherifard, "A clustering method for locating services based on fog computing for the internet of things," *The Journal of Supercomputing*, vol. 78, no. 11, pp. 13 756–13 779, jul 2022.

[190] A. Asensio, X. Masip-Bruin, R. Durán, I. de Miguel, G. Ren, S. Daijavad, and A. Jukan, "Designing an efficient clustering strategy for combined Fog-to-Cloud scenarios," *Future Generation Computer Systems*, vol. 109, pp. 392–406, aug 2020.

[191] T. Hosfeld, F. Metzger, and P. E. Heegaard, "Traffic modeling for aggregated periodic IoT data," in *2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. IEEE, feb 2018, pp. 1–8.

[192] L. Xing, “Reliability in Internet of Things: Current Status and Future Perspectives,” *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6704–6721, aug 2020.

[193] D. Charântola, A. C. Mestre, R. Zane, and L. F. Bittencourt, “Component-based Scheduling for Fog Computing,” in *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing Companion - UCC ’19 Companion.* New York, New York, USA: ACM Press, 2019, pp. 3–8.

[194] M. Peixoto, T. Genez, and L. F. Bittencourt, “Hierarchical Scheduling Mechanisms in Multi-Level Fog Computing,” *IEEE Transactions on Services Computing*, vol. XX, no. X, pp. 1–1, 2021.

[195] X. Shen, H. Yu, J. Buford, and M. Akon, Eds., *Handbook of Peer-to-Peer Networking.* Boston, MA: Springer US, 2010. [Online]. Available: https://link.springer.com/10.1007/978-0-387-09751-0

[196] D. Stutzbach and R. Rejaie, “Characterization of P2P Systems,” in *Handbook of Peer-to-Peer Networking.* Boston, MA: Springer US, 2010, pp. 1253–1276.

[197] M. Aazam, S. Zeadally, and K. A. Harras, “Fog Computing Architecture, Evaluation, and Future Research Directions,” *IEEE Communications Magazine*, vol. 56, no. 5, pp. 46–52, 2018.

[198] D. Wischik, M. Handley, and M. B. Braun, “The resource pooling principle,” *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 5, pp. 47–52, sep 2008.

[199] M. Al-Khafajiy, T. Baker, A. Waraich, O. Alfandi, and A. Hussien, “Enabling high performance fog computing through fog-2-fog coordination model,” *Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA*, vol. 2019-Novem, 2019.

[200] K. M. S, N. Sadashiv, and D. K. S. M, “Load Balancing in Fog Computing: A Detailed Survey,” *International Journal of Computing and Digital Systems*, vol. 13, no. 1, pp. 729–750, apr 2023.

[201] T. H. Luan, L. Gao, Z. Li, Y. Xiang, G. Wei, and L. Sun, “Fog Computing: Focusing on Mobile Users at the Edge,” *arXiv preprint arXiv:1502.01815*, feb 2015.

[202] M. Zghaibeh, K. G. Anagnostakis, and F. C. Harmantzis, “The Behavior of Free Riders in Bit Torrent Networks,” in *Handbook of Peer-to-Peer Networking.* Boston, MA: Springer US, 2010, pp. 1207–1230.

[203] D. A. G. Manzato and N. L. S. da Fonseca, “Incentive Mechanisms for Cooperation in Peer-to-Peer Networks,” in *Handbook of Peer-to-Peer Networking*, X. Shen, H. Yu, J. Buford, and M. Akon, Eds. Boston, MA: Springer US, 2010, pp. 631–660.