UNIVERSITY OF TECHNOLOGY SYDNEY
Faculty of Engineering and Information Technology

Intelligent Context-aware Fog Node Discovery and Trust-based Fog Node Selection

by

Afnan Abdulrahman Bukhari

A Thesis Submitted
in Fulfillment of the
Requirements for the Degree

Doctor of Philosophy

Under the supervision of
Professor Farookh Khadeer Hussain

Sydney, Australia

April 2024

Certificate of Original Authorship

I, Afnan Bukhari, declare that this thesis is submitted in fulfilment of the requirements for the award of PhD in the School of Computer Science / Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise reference or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signature: Production Note: Signature removed prior to publication.

Date: 01/04/2024

ABSTRACT

Intelligent Context-aware Fog Node Discovery and Trust-based Fog Node Selection

by

Afnan Abdulrahman Bukhari

In today's highly advanced technological age, edge devices are widely used. By 2030, Cisco predicts that more than 500 billion edge devices (also known in this research as fog consumers) will be in use [1]. Data from all these devices may experience significant delays when handled, processed and stored through cloud computing. To resolve this issue, fog computing is the best solution. With fog computing, processing, storage, and networking are brought to the edge of the network near fog consumers. This reduces latency, network bandwidth, and response times. Researchers have yet to address the critical challenge of identifying and selecting a reliable and relevant fog node to fog consumers. The existing approaches consider the discovery and selection of fog nodes based on the networking point of view. However, no approach addresses the use of AI-driven mechanisms for intelligent fog node discovery and selection. This research aims to propose an intelligent and distributed framework for context-aware fog node discovery and trust-based fog node selection. This research aims to discover the closest fog nodes in a context-aware manner and select a reliable fog node based on the trust value. The proposed approach is based on the distributed Fog Registry Consortium (FRC) between fog consumers and fog nodes that can facilitate the discovery and selection processes of fog nodes. To ensure that the tasks from the fog consumer are processed in a timely manner, one of the crucial aspects to consider for fog node discovery is the geographic distance between the fog node and the fog consumer as this directly impacts latency, re-

sponse time, and bandwidth usage for fog consumers. Thus, location-based context awareness is one of the key decision criteria for fog node discovery to ensure that the QoS metrics are satisfied. In this research, we propose the Fog Node Discovery Engine (FNDE) within the Distributed Fog Registry (DFR), within FRC, as an intelligent and distributed fog discovery mechanism which enables a fog consumer to intelligently discover fog nodes in a context-aware manner. In this research, the KNN, K-d tree and brute force algorithms are used to discover fog nodes based on the location-based context-aware criteria of fog consumers and fog nodes. Fog node selection is a crucial aspect in the development of a fog computing system. It forms the foundation for other techniques such as resource allocation, task delegation, load balancing, and service placement. Fog consumers have the task of choosing the most suitable and reliable fog node(s) from the available options, based on specific criteria. This research presents the intelligent and reliable Fog Node Selection Engine (FNSE), which is an intelligent method to assist fog consumers to select appropriate and reliable fog nodes in a trustworthy manner. This intelligent mechanism predicts the trust value of fog nodes to help the user select a reliable fog node based on its trust value. Our selection approach is based on the trust value of the fog node based on the values of the QoS factors. If the fog node has historical information of the QoS factors provided to this fog node, then the Trust Evaluation Engine (TEE) in the FNSE is responsible to carry out the prediction of the trust value. With the trust value of fog nodes, the FNSE will be able to rank the fog node to select the most reliable fog node in the network. We propose three mechanisms: the TEE mechanism based on fuzzy logic, the TEE mechanism based on logistic regression, and the TEE mechanism based on a deep neural network. However, if the QoS values of the fog node are unknown, this means the FNSE is unable to make a meaningful selection of fog nodes. To solve the problem of the cold-start fog node, we propose the Bootstrapping Engine (BE) which is an intelligent trust-based fog node bootstrapping framework. This framework is designed to address the cold-start problem in fog computing environments which enables fog consumers to make informed and trustworthy decisions when selecting fog nodes for their applications. To address this challenge, the BE employs two key modules, namely the QoS prediction module and

the reputation prediction module. The QoS prediction module utilizes the k-means clustering and KNN algorithms to predict the initial QoS values of new cold-start fog nodes. Additionally, within the reputation prediction module, we propose three AI methods to achieve the best performance and prediction results, namely fuzzy logic-based reputation prediction, regression-based reputation prediction, and deep learning-based reputation prediction to predict and evaluate the trust value of the new cold-start fog nodes. Finally, we present the simulation of the framework and the evaluation results of each proposed engine which highlight the best performance.

Dissertation directed by Professor Farookh Khadeer Hussain
School of Computer Science
Australian Artificial Intelligence Institute
Faculty of Engineering and Information Technology (FEIT)

Dedication

Dedicated to the memory of my beloved father Mr Abdulrahman Bukhari, whose unwavering presence and guiding light continue to inspire me on this remarkable journey. Though you're not here to witness this adventure, your spirit resonates in every page, reminding me that your love and encouragement are forever etched in my heart.

And to my mother, your strength and love have been the foundation upon which I've built this achievement.

Thank you hardly seems enough to express the depth of my gratitude. Your unwavering love, support, inspiration, and guidance have shaped me into who I am today. Your belief in me has been my guiding star, illuminating every step of this path.

Thank you Mom and Dad.

Acknowledgements

I am thankful to Allah Almighty, who has blessed me and supported me during the writing of this thesis under the supervision of Professor Farookh Khadeer Hussain.

I wish to extend my heartfelt appreciation to my family, whose unwavering support and sacrifices have played an instrumental role in shaping me into the person I am today. My parents, Mr Abdulrahman Bukhari and Mrs Sameera Marghalani, have consistently stood by me, making countless sacrifices that I can never adequately repay. To my husband, Faisal, and my cherished children, Ahmed, Dana, and Sara, your unwavering encouragement and sacrifices have been my pillars of strength. My siblings' words of encouragement and unwavering belief in me have been pivotal in the completion of this thesis. To my entire family, I hope this thesis is a harbinger of positive things to come.

Furthermore, I would like to extend my deepest appreciation to my primary supervisor, Professor Farookh Khadeer Hussain, for his steadfast support, boundless compassion, encouragement, and exceptional guidance. This thesis is a culmination of both his efforts and mine.

Lastly, I would like to acknowledge the invaluable support I received from Taif University and the Saudi Arabia Cultural Mission (SACM) in Australia from the inception of my PhD journey. Their unwavering assistance and encouragement have been instrumental in my academic pursuits.

Afnan Bukhari
Sydney, Australia, 2023.

List of Publications

The following is a list of my research papers during my PhD study.

Journal Papers

J-1. **Afnan Abdulrahman Bukhari**, Farookh Khadeer Hussain, and Omar Khadeer Hussain, ”Intelligent context-aware fog node discovery,” *Internet of Things*, vol. 20, pp. 100607, 2022. doi: 10.1016/j.iot.2022.100607.

J-2. **Afnan Abdulrahman Bukhari**, Farookh Khadeer Hussain, and Omar Khadeer Hussain, ”Fog node discovery and selection: A Systematic literature review,” *Future Generation Computer Systems*, vol. 135, pp. 114-128, 2022. doi: 10.1016/j.future.2022.04.034.

Contents

List of Figures

List of Tables

Abbreviation

FRC - Fog Registry Consortium

FN - Fog Node

FC - Fog Consumer

FND - Fog Node Discovery

FNDE - Fog Node Discovery Engine

FNS - Fog Node Selection

FNSE - Fog Node Selection Engine

FR - Fog Registry

CFR - Central Fog Registry

DFR - Distributed Fog Registry

IoT - Internet of Things

QoS - Quality of Service

BE - Bootstrapping Engine

TEE - Trust Evaluation Engine

DNN - Deep Neural Network

OMNeT++ - Objective Modular Network Test bed in C++

EUA - Edge User Allocation

KNN - K-Nearest Neighbor

SLR - Systematic Literature Review

Chapter 1

Introduction

1.1 Introduction

The purpose of this chapter is to introduce the thesis and provide an overview of the research. Initially, it discusses fog computing and its significance, highlighting the limitations in cloud computing and edge computing compared with fog computing. The unique characteristics and architecture of fog computing are also discussed. Section 1.2 presents the background of this research, delving into the fundamentals of fog computing. In Section 1.3, the statement of the problem addressed in this thesis is explained, highlighting the key challenges in fog node discovery and selection. Furthermore, Section 1.4 outlines the research objective and scope, shedding light on the specific goals and boundaries of the study. In Section 1.5, the scientific and social significance of this thesis is detailed, elucidating the novel contributions and broader impact of the proposed research. In Section 1.6, the thesis structure is outlined, together with a description of the chapters to follow.

1.2 Background

The exponential growth of the Internet of Things (IoT) devices has led to an unprecedented increase in data generation at the network edge. Traditional cloud computing, while powerful, faces limitations in dealing with the sheer volume and low-latency requirements of IoT data. Edge computing seeks to alleviate this by bringing computation closer to the data source, enabling faster response times and reduced data transmission to the central cloud. However, edge computing also has its limitations. It often operates under resource-constrained conditions, with limited processing power and energy resources. This limitation hinders the seamless execution of resource-intensive applications, potentially resulting in service degradation or inefficiencies in data processing [2].

Fog computing has emerged as a promising paradigm to overcome the limitations of traditional cloud computing and edge computing. Fog computing extends the cloud's capabilities by distributing computing and storage closer to the edge of the network. This proximity to end-users enables fog computing to address the challenges of latency, network congestion, and real-time data processing, making it a critical enabler for emerging technologies such as IoT, smart cities, and autonomous systems [3] [4]. Fog computing implementation is as an intermediary between edge devices as we called them (fog consumers) and the cloud, filling the gap between their respective capabilities. By deploying fog nodes at strategic locations in proximity to fog consumers, fog computing facilitates efficient data processing and analysis closer to the source. This distributed architecture not only reduces the burden on the cloud but also improves the overall system's responsiveness and scalability [5].

Edge computing and fog computing are architectures that differ from the centralized cloud computing architecture, as they are hierarchical, decentralized, and distributed. Both edge computing and fog computing are designed to provide services in close proximity to end users, making them ideal for latency-sensitive applications. In edge computing, the services are located directly on edge devices, which are the closest to the end users. However, due to the limited resources of edge devices, they may struggle to handle multiple IoT applications simultaneously, leading to resource contention and increased processing latency [6] [3].

On the other hand, fog computing is situated in network edge devices, which are a single or few network hops away from the edge. It seamlessly integrates edge devices with cloud resources, thereby overcoming the limitations faced by edge computing. By coordinating the use of geographically distributed network edge devices and leveraging cloud resources, fog computing optimizes resource utilization and avoids resource contention at the edge. Although both edge computing and fog computing have the same goal of providing services close to end users, they have underlying differences. Edge computing faces resource constraints, limiting its ability to handle multiple IoT applications, while fog computing effectively addresses these limitations by integrating edge devices and cloud resources. This allows fog computing to balance the use of resources and improve overall resource utilization efficiently [4] [7] [2].

Fog computing has several unique characteristics that set it apart from other computing paradigms [3] [8] [7]. These characteristics include:

1. **Proximity to edge devices:** Fog nodes are located closed to the edge devices. This characteristic enables the fog to support latency-sensitive applications that need real-time processing.

2. **Real-time interactions:** Fog nodes ensure real-time service delivery which decreases latency and overloading the network.

3. **Low latency:** Fog nodes decrease latency and response time by processing data close to the edge devices, which decreases data transmission as well.

4. **Edge analytics:** Fog nodes analyse data locally instead of sending them to the cloud.

5. **Scalability:** Fog computing is designed to scale efficiently, supporting a large number of edge devices and data streams. This scalability makes it suitable for handling the massive influx of data generated by the growing number of IoT devices.

6. **Context awareness:** Fog computing is context aware, meaning it takes into account factors such as location, network conditions, and user requirements when processing data. This context awareness enables adaptive and optimized data processing based on specific conditions.

7. **Heterogeneity:** Fog computing supports a wide range of devices with varying computational capabilities, communication protocols, and storage capacities. This heterogeneous nature allows fog nodes to cater for diverse application needs and integrate with different types of edge devices.

The fog computing paradigm consists of three layers, namely the edge layer, the fog layer and the cloud layer [4] [9] [10] as shown in Figure 1.1.

1. **The edge layer:** The edge layer includes end users or edge devices. In this research, these edge devices are called fog consumers, so the fog consumer is any end entity that connects to a fog node and uses the fog service. The entity could be an individual user, application or edge device such as an IoT device [4] [10].

2. **The fog layer:** The second layer is the fog layer where the fog nodes are located and consists of several distributed static or dynamic fog nodes. Fog nodes can process, store, and analyze data locally, reducing latency and improving performance. Fog nodes can be any device with sufficient capacity, computing power, and energy power such as PCs, servers, smart access points, intelligent gateways, routers, etc. Fog nodes provide a large amount of storage and fast computing capabilities [4] [10].

3. **The cloud layer:** The cloud layer is the top-most layer that includes powerful servers and storage devices. This layer has very high-power computing and

storage capabilities. It provides permanent storage for a massive amount of data and undertakes pervasive computation analysis [4][10].

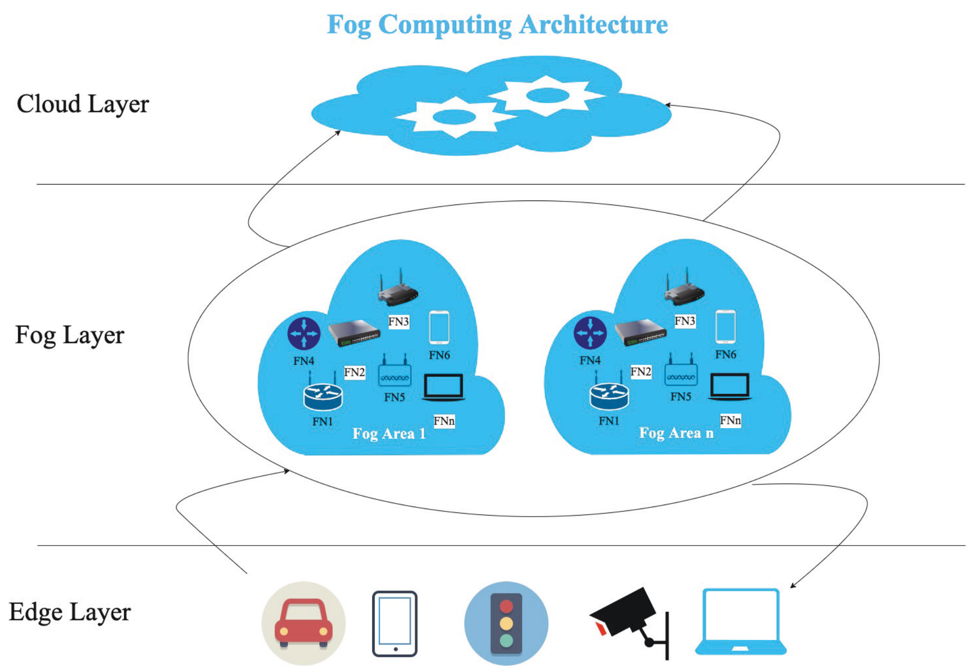


Figure 1.1 : Fog computing architecture

While the fog computing paradigm holds promise, it also comes with its own set of challenges [11]. In addition to the issues inherited from cloud computing, there are specific fog-specific concerns to consider:

1. **Security and Privacy:**

   How can data be kept secure in the fog network? How should fog nodes ensure the security and privacy of data in the fog network? With data being processed at the edge, there is a higher risk of unauthorized access and data breaches. Fog nodes processing data from multiple sources must address privacy concerns. Ensuring data anonymization and access control is essential to protect the privacy of users and organizations [3] [11].

2. **Resource management:**

   There are various challenges involved in discovering fog resources, determining endpoints, allocating resources, and sharing resources in a network. These challenges also apply to fog architecture. Fog resource management is one of the most critical issues, as it determines how analytics application modules are distributed across individual edge devices to maximize performance and minimize latency. It is therefore necessary to develop a platform to evaluate the performance of fog resource management policies on fog computing infrastructures. The use of such a platform will enable fog resource discovery and allocation to be optimized, ensuring optimal task distribution and enhancing fog computing efficiency [3].

3. **Trust evaluation:**

   There are various metrics that are collected and analyzed to determine the trustworthiness of fog resources, including past performance, quality of service (QoS), and user feedback. Reliable fog resource selection will depend on the development of efficient and accurate trust evaluation mechanisms. Implementing robust trust evaluation mechanisms is essential to assess the reliability and reputation of fog resources.

Fog node discovery (FND) plays a crucial role in fog computing systems as it enables the identification and availability of suitable fog nodes within the network. The dynamic nature of fog computing environments, characterized by the addition and removal of nodes and the mobility of devices, necessitates efficient mechanisms for node discovery. Discovering fog nodes in a timely manner is essential for achieving efficient workload distribution, fault tolerance, and resource optimization [12]. By dynamically identifying suitable fog nodes with the right capabilities, proximity, and context awareness, FND enhances resource utilization, reduces latency, and ensures reliable and efficient fog-based services across various industries and applications [12].

Moreover, efficient fog node selection (FNS) plays a pivotal role in fog computing systems for optimal resource utilization and workload distribution. The selection of appropriate fog nodes helps in maximizing the overall system performance by ensuring that computational tasks are allocated to the most suitable resources. Factors such as fog node proximity, computational capabilities, and trustworthiness need to be considered during the future of the FNS process.

1.3 Statement of Problem

The research problem addressed in this thesis revolves around the challenges in FND and FNS within fog computing environments. Specifically, we aim to tackle the complexities of dynamically identifying and selecting appropriate fog nodes for efficient data processing, task allocation, and resource utilization. The challenges include handling the dynamic and heterogeneous nature of fog environments, context-aware FND, ensuring trustworthiness and reliability in fog nodes for FNS, minimizing latency, and optimizing system performance while considering resource constraints and security concerns. We formally describe the research problem in chapter 3.

1.4 Research Objective ans Scope

The objective of this thesis is to address the challenging issues of FND and FNS within fog computing ecosystems. To achieve this, the research aims to develop the FRC framework on the top of fog computing for an intelligent context-aware decision-making mechanism and intelligent trust evaluation technique. The goal is to propose an intelligent, distributed, context aware and reliable approach to optimize the overall performance of the fog computing ecosystem. One of the critical problems is discovering relevant fog nodes that satisfy the fog consumer's requirements, considering factors like geographic distance, and context awareness. In addition, the research focuses on helping fog consumers select trusted fog nodes with verified reliability and capability to ensure timely and efficient task completion. By advancing the field of fog computing through a novel distributed framework and intelligent algorithms, this research has implications for enhancing efficiency and effectiveness across various industries, where fog computing plays a pivotal role in handling diverse data streams related to the IoTs. Ultimately, academia, industry, and society will benefit from improved fog computing capabilities and its practical applications.

1.5 Significance of the Thesis

This thesis makes several scientific and social contributions.

1.5.1 Scientific Contributions

The scientific contributions of this thesis are as follows:

1. The integration of the FRC framework enhances communication and collaboration among fog nodes. This framework enables efficient fog resource management and task allocation within fog computing environments.

2. This research is the first of its type to explore the use of an intelligent AI-driven approach for context-aware FND. There is no approach in the existing literature for context-aware FNS.

3. This research is the first of its type to explore the use of a trust-based mechanism for FNS.

4. This research proposes and validates an intelligent approach for bootstrapping new fog nodes into the fog environment. There is no prior work in the extant

literature on bootstrapping new fog nodes in the fog ecosystem.

5. This research introduces innovative algorithms and architectures for intelligent FND and FNS. The proposed techniques provide valuable insights into addressing the complexities of fog computing environments and optimizing resource usage.

1.5.2 Social Contributions

The social contributions of this thesis are as follows:

1. This research will help fog-driven businesses in multiple sectors such as health, manufacturing, mining, transportation, etc. by enhancing operational efficiency, enabling real-time decision-making.

2. This research will help fog consumers to make trustworthy QoS-based assessments and discover fog nodes in a context-aware manner. This will increase the confidence of the fog consumers in using the fog node and its service.

3. This research contributes to the academic community by advancing the state-of-the-art in fog computing. It introduces a novel framework that can serve as a foundation for further research in the field. Researchers can build upon these findings to explore new directions in context-aware FND and trust-based FNS to enhance the performance of fog-based applications.

4. The implications of this research extend to various industries that rely on fog computing to process data and make decisions efficiently. It can be used in healthcare, transportation, smart cities, and manufacturing industries to optimize resource utilization, reduce latency, and improve the reliability of fog-based services. By leveraging these advancements, industries can enhance their operational efficiency and provide better services to their customers.

The significance of this thesis lies in its novel contributions to the field of fog computing, which not only advance the state-of-the-art but also have broader implications for various industries and the academic community. The proposed intelligent context-aware FND and trust-based FNS approaches enhance the performance of fog computing systems, making them more efficient, reliable, and socially impactful across different domains and applications.

1.6 Thesis Organization

This thesis is organised into nine chapters as shown in Figure 1.2. The current chapter introduces fog computing and the challenges associated with it. Furthermore, the significance of this thesis is briefly presented. The social and scientific contributions are also briefly overviewed. The subsequent chapters of this thesis are structured as follows:

Chapter 2: Chapter 2 provides an extensive review of the existing methods for FND and FNS in the current literature. In this chapter, the sub-tasks of FND and FNS are categorized into the different criteria that they should meet to satisfy their requirements. The existing literature on FND and FNS is identified, and a comparative analysis is undertaken to determine if they consider or address the defined criteria required for FND and FNS. The shortcomings are identified in the existing approaches based on the comparative analysis and these are presented as open issues for future research directions in FND and FNS.

Chapter 3: In Chapter 3, a formal definition is presented for each of the problems that will be explored in this thesis. Key terminologies and keywords used throughout the thesis are also clarified in this chapter. The main research problem is meticulously outlined, and it is further divided into four research issues, which emerged from the existing literature. From these research issues, specific research questions and objectives are formulated to underscore the importance and relevance of this study.

Chapter 4: Chapter 4 briefly highlights the proposed FRC framework. Chapter 4 presents an overview of the solution to each of the issues identified in Chapter 3. Chapter 4 also points to the chapters which elaborate on the detailed solutions for the identified research issues.

Chapter 5: Chapter 5 explains in detail the proposed solution of developing the architecture of the FRC and its pivotal role. This chapter elaborates in detail the activities of the FRC. The working steps of the FRC are explained. Furthermore, Chapter 5 presents the implementation and prototype setup of the FRC.

Chapter 6: Chapter 6 presents a comprehensive explanation of our intelligent approach to discovering fog nodes in a context-aware manner, particularly focusing on location-based considerations. This chapter thoroughly discusses the framework of the FNDE, offering insights into its structure and functionality. The working steps of FNDE is thoroughly explained, accompanied by a detailed algorithm outlining its operation. Additionally, Chapter 6 presents the implementation details of the FNDE framework, discussing the setup and configurations. Extensive experiments and the evaluation process are discussed in detail and the results are analyzed and presented in this chapter.

Chapter 7: Chapter 7 focuses on the detailed solution for an intelligent method aimed at predicting the trust value of fog nodes. Chapter 7 presents the intelligent and reliable FNSE for FNS. This chapter elaborates on the architecture of the TEE framework, providing valuable insights into its design and components. Furthermore, the chapter covers the implementation details of the TEE framework and the prototype setup used for validation. In addition, the comprehensive experiments and the results are detailed and analyzed.

Chapter 8: Chapter 8 presents a comprehensive solution for an intelligent approach aimed at predicting the trust value of fog nodes and facilitating the bootstrapping of new fog nodes with unknown QoS. In detail, the chapter thoroughly explains the architecture of the BE framework, providing a clear understanding of its design and components. Moreover, the implementation of BE, along with the prototype setup used for practical validation, is thoroughly described.

Chapter 9: Chapter 9 summarizes the thesis and potential future research directions are explored.

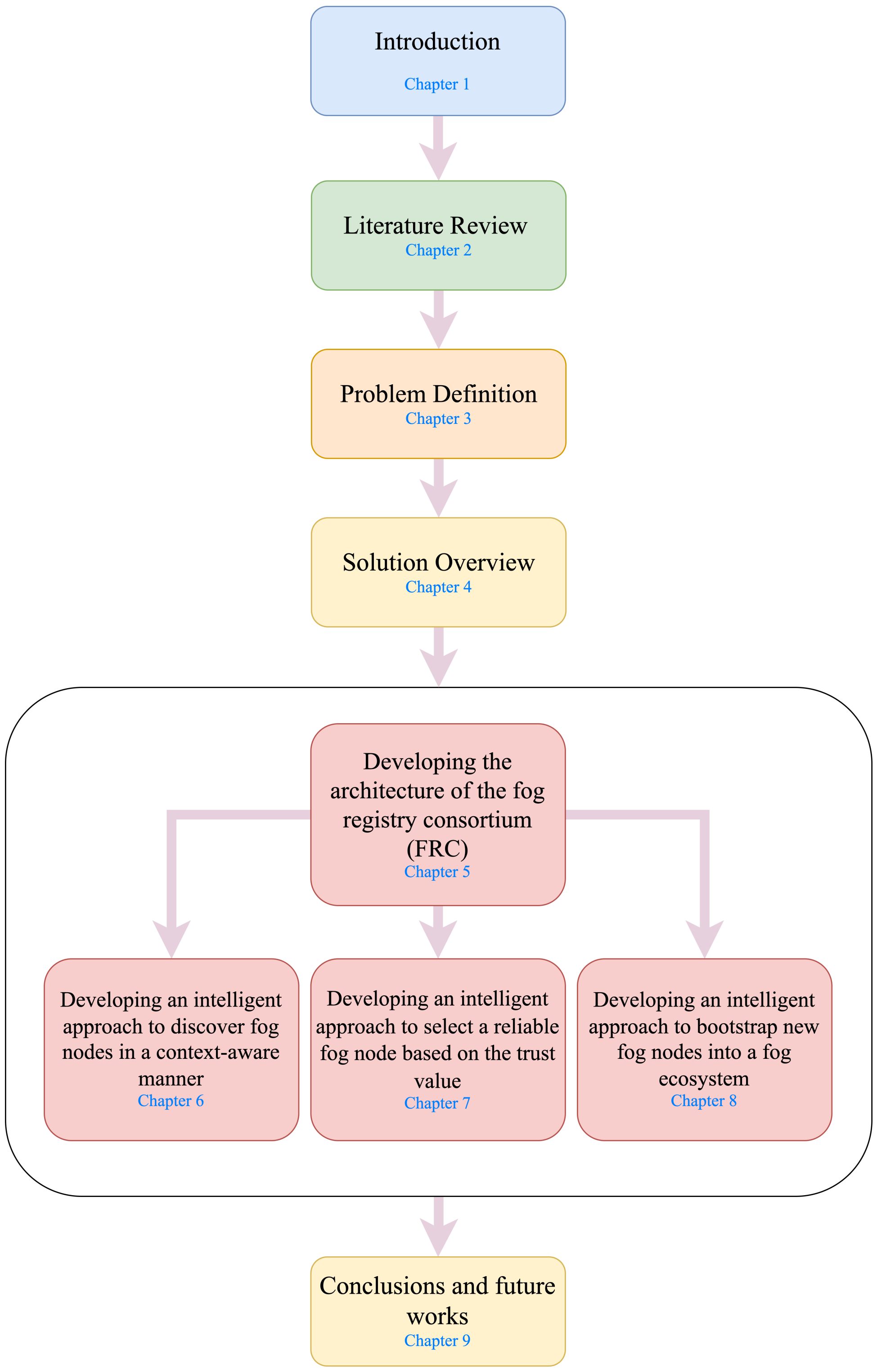

Introduction
Chapter 1
Literature Review
Chapter 2
Problem Definition
Chapter 3
Solution Overview
Chapter 4
Developing the architecture of the fog registry consortium (FRC)
Chapter 5
Developing an intelligent approach to discover fog nodes in a context-aware manner
Chapter 6
Developing an intelligent approach to select a reliable fog node based on the trust value
Chapter 7
Developing an intelligent approach to bootstrap new fog nodes into a fog ecosystem
Chapter 8
Conclusions and future works
Chapter 9

Figure 1.2 : Thesis structure

Chapter 2

Literature Review

2.1 Introduction

This chapter overviews the existing literature on FND and FNS. Section 2.2 provides the background of fog computing while Section 2.3 outlines the key requirements for FND and FNS, and compares the existing literature. Section 2.4 discusses the process adopted to conduct the systematic literature review (SLR) and the shortlisted papers that match the query result. Section 2.5 presents a critical analysis of the existing FND approaches and highlights their shortcomings in meeting the requirements defined in Section 2.3. Similarly, Section 2.6 critically evaluates the existing FNS approaches and identifies the gaps in fulfilling the requirements defined in Section 2.3. Section 2.7 discusses the limitations of the existing FND and FNS approaches that need to be addressed. Finally, Section 2.8 concludes the chapter by summarizing the key findings and implications of this review.

It is important to note that the content of this chapter has been previously published in the journal Future Generation Computer Systems [10].

2.2 Background of fog computing

There has been constant growth in the number of IoT devices over the past decade. According to Cisco, this growth will continue and it is estimated that there will be more than 500 billion IoT devices by 2030 [6]. While these devices (also known as edge devices) generate a massive amount of data, they lack the capability to process it. Such processing is done in a centralized cloud environment, where the data from the edge devices is transferred to them, processed and then sent back. However, the cloud environment has network bandwidth limitations which may increase latency and response time, resulting in delays [13]. To address these drawbacks, fog computing, in which the computations are done by virtualized or non-virtualized devices, has been proposed as an extension to cloud computing [4]. As shown in Figure 1.1, a fog computing paradigm consists of three layers, namely the edge layer, the fog layer and the cloud layer. The edge layer includes devices such as sensors, actuators, smart devices, etc. (termed edge devices) from the physical environment. The fog layer is the intermediate layer and consists of several distributed static or dynamic fog nodes. Fog nodes can be virtual units like virtual machines or physical units such as PCs, servers, smart access points, intelligent gateways, routers, etc. [14]. The cloud layer is the top-most layer that includes powerful servers and storage devices [15]. In fog computing, the raw data from the edge devices are captured in the edge layer and transferred to the fog layer. The fog node closest to the edge devices processes the task and sends the response back while also updating the results in the cloud layer. Such processing and communication between the three layers of fog computing assist in processing latency-sensitive, context-aware and real-time applications, thereby ensuring scalability, decreasing latency and response time while saving bandwidth [6] [15] [16].

A fog service is defined as an association between an edge device and a fog node in which the edge device uses the fog node's computing resources to process its task. Given the benefits of fog computing in processing resource-constrained and time-critical applications, researchers have worked on different aspects needed to facilitate a fog service. For example, techniques have been proposed for resource provisioning [15], resource management [16], task offloading [17] [18] [19], load balancing [13], service placement [20] [21] etc. Researchers have also looked at ensuring the privacy and security of the fog service environment [22] [23][24]. However, all these techniques focus on the processing stage of the fog service. We define the processing stage as that in which the edge device's tasks are processed by the fog nodes. While this is an important stage of the fog node, a pre-requisite for this stage to be successful is the choice stage, in which the edge devices select appropriate fog node/s among those available with which to form a service. The choice stage of the fog node can further be sub-divided into two sub-tasks, namely FND and FNS. FND is a process of finding the relevant fog nodes from the available ones that satisfy the edge device's requirements. This is an important step as not all available fog nodes will satisfy the specific characteristics needed by the edge devices. If these are not considered during the selection process, it will result in selecting a fog node that does not meet the requirements of the edge device and will thus negatively impact the QoS being delivered during the processing stage of the fog node. Once the relevant fog nodes have been discovered in the FND, then the sub-task of FNS selects the best-suited fog node/s from those available that satisfy the required criteria. Thus, FND in combination with FNS assists in meeting the objectives of the choice stage of the fog node which thereby assists in completing the processing stage of the service according to the QoS requirements. Table 2.1 shows a selected snapshot of recent survey articles published in the area of fog computing. From the analysis, we can see that while researchers have surveyed the role of different focus areas in facilitating fog computing, their primary focus is on the processing stage of the fog node in general and not on the tasks of analysing which methods and approaches have been used for FND or FNS. This makes it difficult to identify the progress that has been made in these tasks. In this chapter, we attempt to address this by carrying out a SLR in the area of FND and FNS. By doing so, we aim to identify the research gaps and outline future open issues and challenges.

Table 2.1 : A comparison of existing survey articles on fog computing to identify whether they focus on FND and FNS.

Articles	**Year of publication**	**Do they focus on FND?**	**Do they focus on FNS?**	**Area of focus**
Mouradian et al. [25]	2018	No	No	Architecture and algorithms of fog system
Sabireen and Neelanarayanan [26]	2021	No	No	Fog system algorithms
Yi et al. [27]	2015	No	No	Fog computing
Bellavista et al. [28]	2018	No	No	Fog system solutions
Lan et al. [29]	2019	No	No	Fog programming
Kaur and Aron [30]	2020	No	No	Load balancing
Yousefpour et al. [31]	2019	Yes	No	Fog computing
Nayeri et al. [32]	2021	No	No	Application placement
Islam et al. [33]	2021	No	No	Context-aware scheduling
Bakhshi et al. [34]	2019	No	No	Dependability in fog computing

Articles	**Year of publication**	**Do they focus on FND?**	**Do they focus on FNS?**	**Area of focus**
Hurbungs et al. [35]	2021	Yes	No	Partitioning and offloading tasks, sustainable energy consumption, edge analytics, edge security, edge node and data discovery, edge quality of service.
Pandeeswari and Padmavath [36]	2020	No	No	Fog computing architecture
Caiza et al. [37]	2020	No	No	Architecture, security, latency, energy consumption
Perera et al. [38]	2017	No	No	Fog computing
Khalid et al. [39]	2019	No	No	Privacy preserving and access control
Puliafito et al. [40]	2019	No	No	Application domain
Sadique et al. [41]	2020	No	No	Trust management
Parveen et al. [42]	2020	No	No	Fog computing challenges

Articles	**Year of publication**	**Do they focus on FND?**	**Do they focus on FNS?**	**Area of focus**
Singh et al. [43]	2021	No	No	Fog system development, fog metrics, fog platforms, fog frameworks
Naha et al. [4]	2018	No	No	Fog infrastructures, fog platforms, fog applications
Ogundoyin and Kamil [44]	2021	No	Yes	Optimization algorithms in fog computing
Ghobaei-Arani et al. [16]	2019	No	No	Resource management
Zhang et al. [24]	2018	No	No	Security and trust

2.3 Key requirements to consider for FND and FNS

As discussed in the previous section, the discovery and later selection of the right fog node are precursors that need to be satisfied to achieve the desired outcomes of a fog service. To meet these precursors, there are various requirements that need to be considered during FND and FNS, as follows:

FND1 - Location-Awareness: To ensure that the tasks from the edge devices are processed in a timely manner, one of the crucial aspects to consider for FND is the geographic distance between the fog node and the edge devices as this directly impacts latency, response time, and bandwidth usage [45]. Thus, location-awareness should be one of the key decision criteria for FND to ensure that the QoS metrics are satisfied.

FND2 - Context-awareness: Context is a multi-attribute requirement that captures the identity of the fog nodes, their capability, profile and the time at which they are available. Capturing these attributes of a context for FND is very important to filter out the nodes that do not match the requirements to prevent them from being considered further [46].

FND3 - Edge-device characteristics: Edge devices may deal with time-critical applications but have limited time and energy left. In such a case, if they form a fog service, the edge devices need to ensure that it is possible to return the required processed task in light of the available network performance, bandwidth and latency constraints. So, these constraints must be considered during the process of FND [47].

FND4 - Intelligent matching: The context-specific nature of a fog node should be matched with the type of task/s required by the edge devices. This is to ensure that only those fog nodes that have the computing capacity to meet the edge device requirements are considered further in the selection stage. This requires balancing the different constraints of the edge device requirements to the context-specific nature of the fog nodes [47].

FND5 - Proven to perform in a real-world setting: The FND approach should be proven to work in a real-world environment either through simulation experiments or real-world experiments. This is important to ensure that the output recommended by the model can be trusted.

Similarly, the requirements to be considered during FNS are as follows:

FNS1 - Informed selection: Similar to the discovery process, there may be more than one fog node that has the capability to meet the requirements of the edge device.

The recommended fog node should be one that balances the different constraints to ensure that the QoS requirements related to network performance, available bandwidth, latency, and energy consumption are met [47].

FNS2 - Trustworthy: An edge device selects a fog node based on the trust that its tasks will be processed and delivered in the required time frame. In other words, it selects a fog node in the belief that it will deliver on its promised outcomes [48]. So, a key requirement to consider for FNS is the past capability of a fog node in meeting its expectations and utilizing it in the future selection decisions.

FNS3 - Timely output: The fog node recommended from the FNS should be such that it has the capacity and capability to give timely output to ensure that edge devices are able to deal with time-critical applications despite their limited energy.

FNS4 - Proven to perform in a real-world setting: The FNS approach should be proven to work in a real-world environment either through simulation experiments or real-world experiments. This is important to ensure that the output recommended by the model can be trusted.

In the following sections, we determine whether the requirements detailed in FND1-FND5 and FNS1-FNS4 are used in the workings of the existing approaches for fog discovery and fog selection. In the next section, we discuss the process adopted to identify the related papers from the literature.

2.4 Process adopted to identify papers for SLR

SLR is a helpful method to understand and summarize the solutions proposed in an area by performing a critical appraisal of the current work. It also assists in identifying the open issues that remain unsolved in that area [49]. This SLR was conducted between 2019-2023. Undertaking an SLR consists of several steps as follows:

Step 1: Searching the literature according to the scope of the search process.

Step 2: Applying inclusion and exclusion criteria for selection.

Step 3: Study selection process.

Step 4: Data extraction and synthesis.

Figure 2.1 represents the Selection process of the SLR.

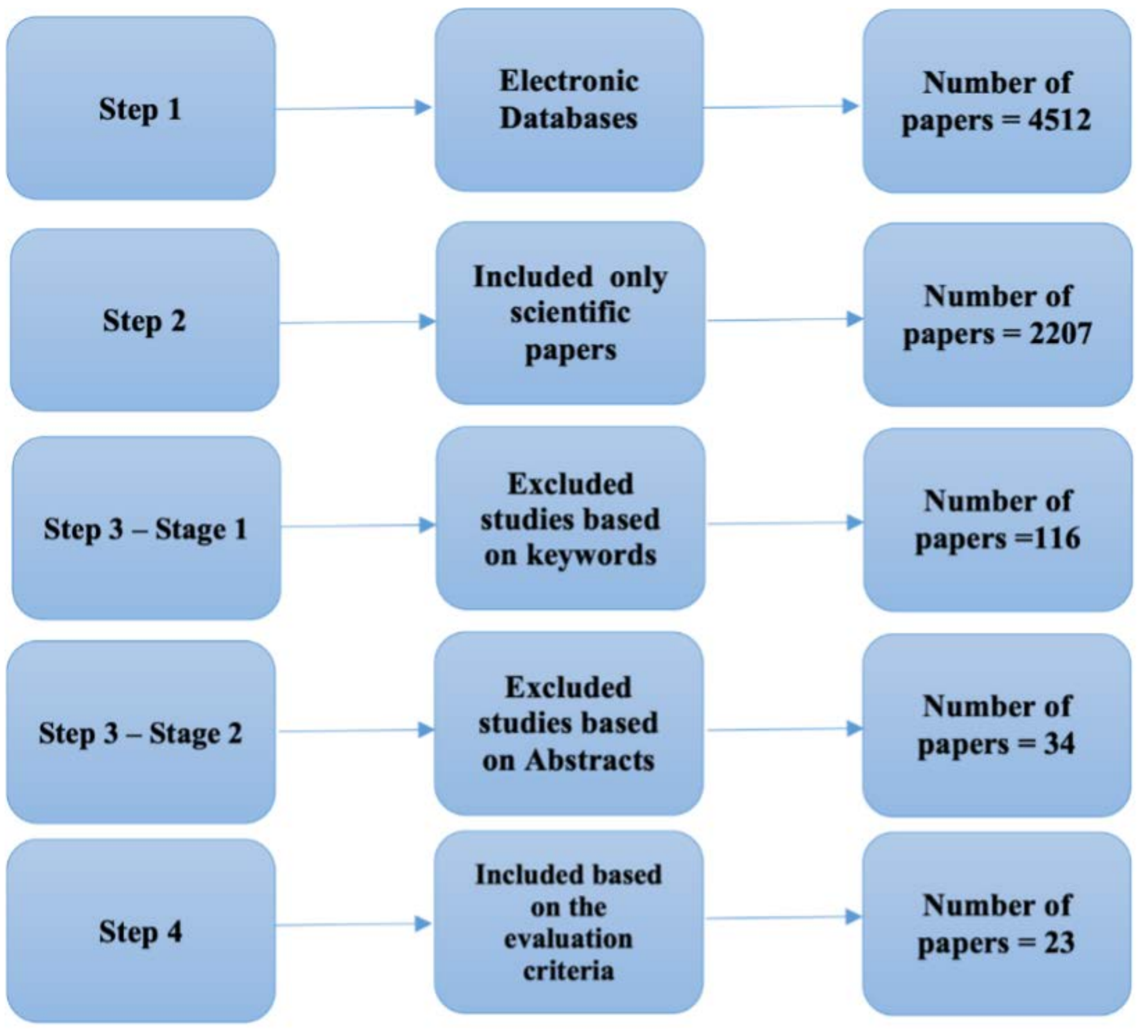


Figure 2.1 : Schematic representation of the selection process of the SLR

2.4.1 Step 1: Searching the literature according to the scope of the search process

In this step, the databases and keywords used to search for papers in the literature are identified. The following well-known electronic databases were used to find relevant papers:

- IEEE Xplore (http://www.ieeexplore.ieee.org)
- ACM Digital Library (https://dl.acm.org)
- Springer (https://link.springer.com/), and
- Science Direct (Elsevier) (https://www.sciencedirect.com/)

We defined two sets of keywords to search for papers related to FND and FNS in the literature. The keywords to search for node discovery are (fog AND computing AND discovery AND service AND node) whereas the keywords to search for node selection are (fog AND computing AND node AND service AND selection). We performed a search using these keywords and retrieved a total of 4512 papers. The search results were saved in a database which is available at here

2.4.2 Step 2: Applying inclusion and exclusion criteria for selection

In this step, we formalized the inclusion and exclusion criteria to filter the retrieved papers. The first criterion for inclusion was that the studies should be conducted from 2014 (since the introduction of fog computing) onwards. The second criterion for inclusion was that they should only be scientific papers such as journal or conference papers. Non-academic papers such as short papers, tutorials, newsletters, and magazines were excluded. Duplicated papers and survey papers were also removed. The third criterion for inclusion was that the studies should be in the English language. A total of 2207 papers remained after applying the inclusion and exclusion criteria.

2.4.3 Step 3: Study Selection Process

In this step, the decision to include an article in the literature review was made in two stages. In the first stage, a decision for inclusion was made after reading the title and the keywords of each paper. In the second stage, a decision for inclusion was made based on the abstract of each paper. In the first stage, the title and keywords of the 2207 papers were thoroughly reviewed. Papers which were not relevant to the defined criteria or the aim of this study were excluded. However, in this stage, it was not clear from some titles and keywords as to whether the papers were relevant or not. So, these papers were not excluded and were assessed further in the next stage. A total of 116 papers remained at the end of this stage. In Stage 2, the abstracts of these papers were read to determine their relevance to this study. If the abstract of a paper was found to be relevant, the paper was selected, otherwise it was excluded. At the end of this stage, 34 papers were deemed to be relevant to this study.

2.4.4 Step 4: Data extraction and synthesis

In this step, the 34 papers were evaluated using three quality criteria questions to ensure that there was no bias in the selection process and that they were relevant for inclusion in the SLR. The three quality criteria questions are:

- QE1: Does the paper deal with fog service or node discovery?
- QE2: Does the paper deal with fog service or node selection?
- QE3: Has the approach proposed in the paper been evaluated or tested?

If the answer was 'yes' to at least two of the three criteria questions, the paper was included in the SLR. From the 34 papers, only 23 papers satisfied the selection requirement as shown in Table 2.2. From the 23 shortlisted articles, as shown in Table 2.3, 13 articles focused on approaches to discover fog nodes and 10 articles focused on FNS. In Section 2.5, we discuss the studies related to fog discovery and in Section 2.6, we discuss the studies related to fog selection.

Table 2.2 : Evaluation of the papers against the quality criteria questions

Article	**Title**	**QE1**	**QE2**	**QE3**
Soo et al. [50]	Proactive Service Discovery in Fog Computing Using Mobile Ad Hoc Social Network in Proximity	Yes	No	Yes
Venanzi et al. [51]	MQTT-Driven sustainable node discovery for internet of things-fog environments	Yes	No	Yes
Venanzi et al. [52]	MQTT-Driven Node Discovery for Integrated IoT-Fog Settings Revisited: The Impact of Advertiser Dynamicity	Yes	No	Yes
Rejiba et al. [53]	F2C-Aware: Enabling Discovery in Wi-Fi-Powered Fog-to-Cloud (F2C) Systems	Yes	No	No
Rejiba et al. [54]	Analyzing the Deployment Challenges of Beacon Stuffing as a Discovery Enabler in Fog-to-Cloud Systems	Yes	No	Yes
Rejiba et al. [55]	A Beacon-assisted direction-aware scanning scheme for 802.11-based discovery in Fog-to-Cloud systems	Yes	No	Yes

Article	Title	QE1	QE2	QE3
Rejiba et al. [56]	Towards a context-aware Wi-Fi-based Fog Node discovery scheme using cellular footprints	Yes	No	Yes
Gedeon et al. [57]	Sunstone: Navigating the Way Through the Fog	Yes	No	Yes
Venanzi et al. [58]	Fog-Driven Context-Aware Architecture for Node Discovery and Energy Saving Strategy for Internet of Things Environments	Yes	No	Yes
Santos et al. [59]	Towards Dynamic Fog Resource Provisioning for Smart City Applications	Yes	No	Yes
Skiadopoulos et al. [60]	Multiple and replicated random walkers analysis for service discovery in fog computing IoT environments	Yes	No	Yes
Karagiannis et al. [12]	Addressing the node discovery problem in fog computing	Yes	No	Yes
Tomar and Matam [61]	Optimal Query-Processing-Node Discovery in IoT-Fog Computing Environment	Yes	No	Yes
Pešić et al. [62]	Bluetooth Low Energy Microlocation Asset Tracking (BLEMAT) in a Context-Aware Fog Computing System	No	No	Yes

Article	**Title**	**QE1**	**QE2**	**QE3**
Henze et al. [63]	Fog Horizons – A Theoretical Concept to Enable Dynamic Fog Architectures	No	No	No
Mahmud et al. [64]	Latency-Aware Application Module Management for Fog Computing Environments	No	No	No
Nair and Somasundaram [65]	Overload prediction and avoidance for maintaining optimal working condition in a fog node	No	Yes	Yes
Jabri et al. [66]	Vehicular fog gateways selection on the internet of vehicles: A fuzzy logic with ant colony optimization based approach	No	Yes	Yes
Mishra et al. [67]	An adaptive model for resource selection and allocation in fog computing environment	No	Yes	Yes
Rejiba et al [68]	Towards user-centric, switching cost-aware fog node selection strategies	No	Yes	Yes
Sami and Mourad [69]	Dynamic On-Demand Fog Formation Offering On-the-Fly IoT Service Deployment	No	No	Yes
Pan et al. [70]	A Novel Fog Node Aggregation Approach for Users in Fog Computing Environment	No	Yes	Yes

Article	Title	QE1	QE2	QE3
Li et al. [71]	Fog Node Selection for Low Latency Communication and Anomaly Detection in Fog Networks	No	Yes	Yes
Irtisam et al. [72]	Pathfinder: A Fog Assisted Vision-Based System for Optimal Path Selection of Service Robots	No	No	Yes
Mostafa [45]	Cooperative Fog Communications using A Multi-Level Load Balancing	No	Yes	Yes
Lera et al. [73]	Analyzing the Applicability of a Multi-Criteria Decision Method in Fog Computing Placement Problem	No	No	Yes
Nguyen et al. [74]	A Market-Based Framework for Multi-Resource Allocation in Fog Computing	No	No	Yes
Iyer et al. [75]	On the strategies for Risk Aware Cloud and Fog Broker Arbitrage Mechanisms	No	Yes	Yes
Yang et al. [76]	Adverse Selection via Matching in Co-operative Fog Computing	No	No	No
Rahman et al. [77]	Efficient Edge Nodes Reconfiguration and Selection for the Internet of Things	No	Yes	Yes
Baranwal and Vidyarthi [78]	FONS: a fog orchestrator node selection model to improve application placement in fog computing	No	Yes	Yes

Article	Title	QE1	QE2	QE3
Yang et al. [79]	Multi-attribute selection of maritime heterogenous networks based on SDN and fog computing architecture	No	No	Yes
Velasquez et al. [80]	A Rank-based Mechanism for Service Placement in the Fog	No	No	Yes

Table 2.3 : Categorization of papers as either fog discovery or fog selection

Research article	Year	Title of the article	Category
Soo et al. [50]	2016	Proactive Service Discovery in Fog Computing Using Mobile Ad Hoc Social Network in Proximity	Fog discovery
Venanzi et al. [51]	2018	MQTT-Driven sustainable node discovery for internet of things-fog environments	Fog discovery
Venanzi et al. [52]	2018	MQTT-Driven Node Discovery for Integrated IoT-Fog Settings Revisited: The Impact of Advertiser Dynamicity	Fog discovery
Rejiba et al. [53]	2018	F2C-Aware: Enabling Discovery in Wi-Fi-Powered Fog-to-Cloud (F2C) Systems	Fog discovery

Research article	**Year**	**Title of the article**	**Category**
Rejiba et al. [54]	2018	Analyzing the Deployment Challenges of Beacon Stuffing as a Discovery Enabler in Fog-to-Cloud Systems	Fog discovery
Rejiba et al. [55]	2018	A Beacon-assisted direction-aware scanning scheme for 802.11-based discovery in Fog-to-Cloud systems	Fog discovery
Rejiba et al. [56]	2018	Towards a context-aware Wi-Fi-based Fog Node discovery scheme using cellular footprints	Fog discovery
Gedeon et al. [57]	2020	Sunstone: Navigating the Way Through the Fog	Fog discovery
Venanzi et al. [58]	2019	Fog-Driven Context-Aware Architecture for Node Discovery and Energy Saving Strategy for Internet of Things Environments	Fog discovery
Santos et al. [59]	2018	Towards Dynamic Fog Resource Provisioning for Smart City Applications	Fog discovery
Skiadopoulos et al. [60]	2019	Multiple and replicated random walkers analysis for service discovery in fog computing IoT environments	Fog discovery
Karagiannis et al. [12]	2020	Addressing the node discovery problem in fog computing	Fog discovery

Research article	**Year**	**Title of the article**	**Category**
Tomar and Matam [61]	2018	Optimal Query-Processing-Node Discovery in IoT-Fog Computing Environment	Fog discovery
Nair and Somasundaram [65]	2019	Overload prediction and avoidance for maintaining optimal working condition in a fog node	Fog selection
Jabri et al. [66]	2019	Vehicular fog gateways selection on the internet of vehicles: A fuzzy logic with ant colony optimization based approach	Fog selection
Mishra et al. [67]	2019	An adaptive model for resource selection and allocation in fog computing environment	Fog selection
Rejiba et al [68]	2020	Towards user-centric, switching cost-aware fog node selection strategies	Fog selection
Pan et al. [70]	2020	A Novel Fog Node Aggregation Approach for Users in Fog Computing Environment	Fog selection
Li et al. [71]	2019	Fog Node Selection for Low Latency Communication and Anomaly Detection in Fog Networks	Fog selection

Research article	**Year**	**Title of the article**	**Category**
Mostafa [45]	2019	Cooperative Fog Communications using A Multi-Level Load Balancing	Fog selection
Iyer et al. [75]	2020	On the strategies for Risk Aware Cloud and Fog Broker Arbitrage Mechanisms	Fog selection
Rahman et al. [77]	2019	Efficient Edge Nodes Reconfiguration and Selection for the Internet of Things	Fog selection
Baranwal and Vidyarthi [78]	2021	FONS: a fog orchestrator node selection model to improve application placement in fog computing	Fog selection

2.5 Discussion of FND approaches in addressing requirements FND1-FND5

Most of the existing research on FND considers that fog nodes are located close to the edge device. So, they assume that the edge devices have already discovered the fog nodes and thus focus on the next phase of their operation, i.e., processing the communication between the devices. A few studies focus on the actual discovery of fog nodes and the approach they utilize to achieve this can be categorized as follows and as shown in Figure 2.2:

1. Broadcasting based on an 802.11 Wi-Fi beacon [53][54][55],
2. Message Queuing Telemetry Transport (MQTT) which uses publish/subscribe [51][52][58]
3. Wi-Fi-based FND [56]

4. DNS and BGP community-based FND [57]

5. Peer-to-Peer-based FND[50][59]

6. Static attribute table-based FND [61]

7. Exploration methods [60][12]

2.5.1 Broadcasting based on an 802.11 Wi-Fi beacon

The approaches under this category enable edge devices to find appropriate fog services using their signals. For example, Rejiba et al. [53] [54] presented the F2C (fog-to-cloud) system for discovering fog services. The proposed approach embeds the 802.11 beacon technique to discover fog services that are close to the vicinity of the edge devices using Wi-Fi technology. The F2C architecture consists of 3 entities, agents, area, and leader. The agent could be is the fog services, the area is a group of agents, and the leader is the fog service which is a high-capacity agent who manages the area. The fog agent must know if the leader is present in the boundary of the fog service device to connect to them. The discovery process works when the leader of the fog service area's boundary announces its presence by sending an 802.11 beacon message. The agent at the edge level listens to the leader's announcement and detects its presence before joining and connecting to the fog service. In another work [55], the authors proposed a solution for broadcasting, advertising and scanning the presence of the leader to reduce energy consumption. A novel beacon-assisted direction-aware scanning scheme (BDSS) which aims to control the scanning process of edge devices is presented. Edge devices should scan the area to find the leader to connect to the system. If the device moves in the same area, the scanning process will be unnecessary and will lead to a wastage of the devices' energy. To overcome this, the authors propose to save energy by enabling the scanning process when the device reaches a certain distance threshold. When the surrounding environment

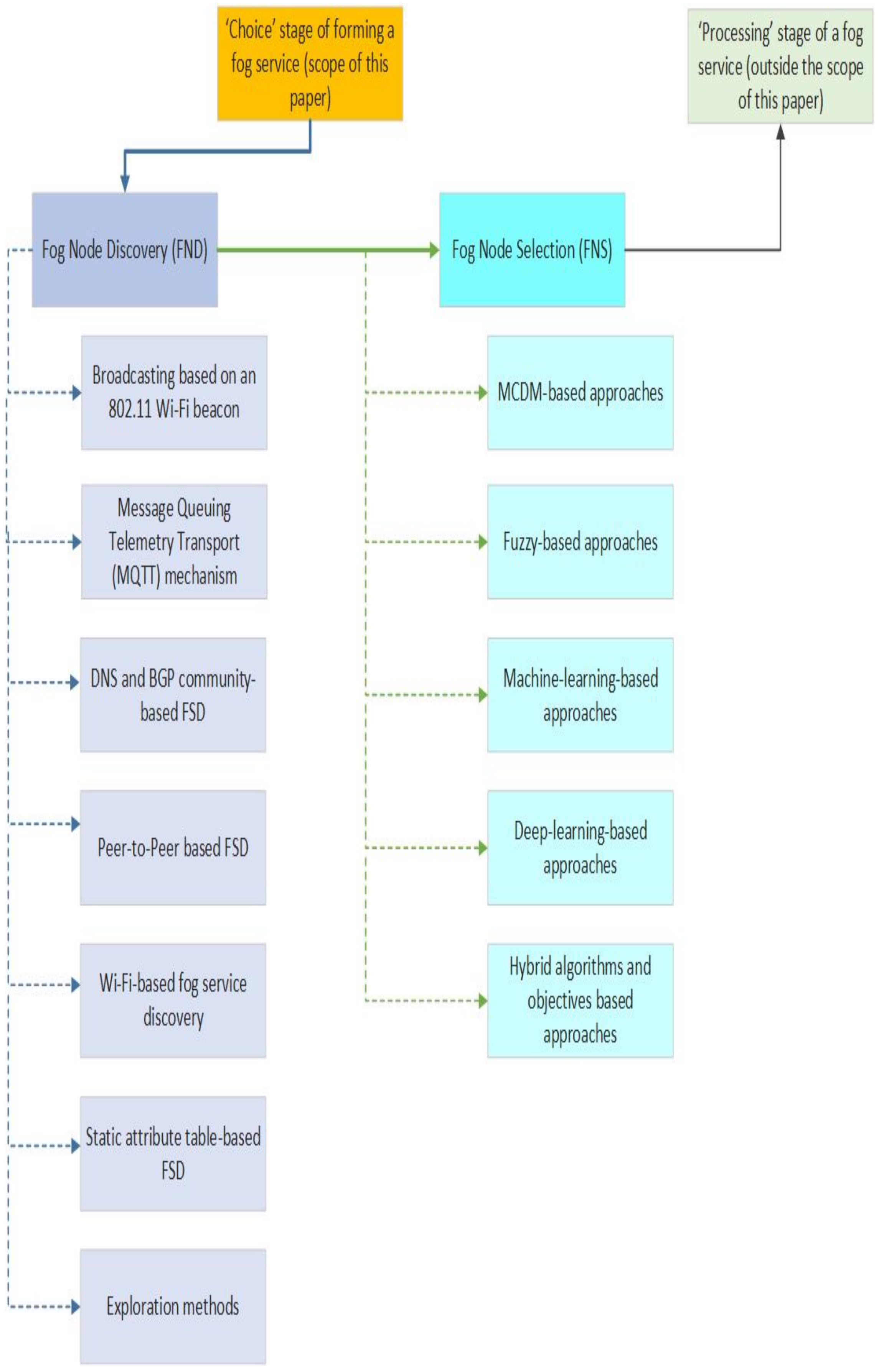

'Choice' stage of forming a fog service (scope of this paper)
'Processing' stage of a fog service (outside the scope of this paper)
Fog Node Discovery (FND)
Fog Node Selection (FNS)
Broadcasting based on an 802.11 Wi-Fi beacon
Message Queuing Telemetry Transport (MQTT) mechanism
DNS and BGP community-based FSD
Peer-to-Peer based FSD
Wi-Fi-based fog service discovery
Static attribute table-based FSD
Exploration methods
MCDM-based approaches
Fuzzy-based approaches
Machine-learning-based approaches
Deep-learning-based approaches
Hybrid algorithms and objectives based approaches

Figure 2.2 : Categorization of the existing literature based on their adopted techniques for FND and FNS

of the device is changed and the device reaches a certain distance threshold, the scanning process is performed. The proposed discovery approach focuses on reducing the energy consumption of edge devices based on the discovery of a wireless network signal range using the 802.11 beacon technique. While the proposed approaches detect and connect to the fog services using Wi-Fi and the 802.11 beacon technique, they fail to consider the location-awareness FND1 and context-awareness FND2 requirements, such as but not limited to user location, user identity, etc. in the discovery process. Other characteristics of edge devices such as time and energy constraints (as defined in FND3) are not considered in deciding in an intelligent manner which fog nodes are the most appropriate to meet the requirements of the edge devices (requirement defined in FND4). While the authors demonstrate the application of the proposed model using a Mininet-WiFi emulator [81] (requirement defined in FND5), not all of them are applied on a real-world test bed.

2.5.2 Message Queuing Telemetry Transport (MQTT) protocol

Approaches in this category use the MQTT protocol to discover edge devices with the help of the fog node as an MQTT broker. For example, Venanzi et al. [51] [52] proposed a Bluetooth-based MQTT-driven node discovery solution in an IoT-fog environment. The proposed approach is termed Power Efficient Node Discovery (PEND) and aims to ensure sustainability, energy efficiency, discoverability, and reliability while engaging in FND. The goal of this approach is to reduce the energy consumed by the edge device during the discovery process and to ensure IoT devices are 100% discoverable. A fog node as the MQTT broker between the IoT devices and the centralized cloud server is implemented. The edge device (IoT device) could be a Bluetooth Low Energy Scanner (BLE-S) or Bluetooth Low Energy Advertiser (BLE-A), with the former being a subscriber and the latter being a publisher. The MQTT broker monitors the BLE-A's path and controls the BLE-S's Bluetooth interface by implementing a signalling scheme based on the location of the BLE-A. The BLE-S is in a fixed location and scans its range continuously to detect the Bluetooth devices in the vicinity. BLE-A is a mobile device that may or may not appear in the scanning time frame of the BLE-S. BLE-A sends its location to the MQTT broker (in the fog layer) by publishing the MQTT message on a specific topic. The MQTT broker knows the location of each BLE-S and when it receives a MQTT message, it calculates the distance between the specific BLE-A and all the BLE-Ss in the environment. The MQTT broker then sends a wake-up message to the nearest BLE-S whose location is below a certain threshold. The BLE-S starts scanning its range only when a new BLE-A comes into range. The proposed approach achieves 100% discoverability by synchronizing the advertisement and scanning frames and helps to save the BLE, CPU, and per-application battery consumption. This is because the Bluetooth scanner interface is only activated whenever a BLE-A service is in the vicinity. The authors, in their further work, proposed an energy-saving technique by deciding when to switch BLE interfaces on or off based on the expected frequency of the approaching services [58]. For power consumption purposes, the BLE-S is kept off until the fog node alert and it wakes up only when the BLE-A is in the BLE range. Although the proposed approaches have been applied on a real-world platform (satisfying requirement FND5), they do not consider the context-aware (requirement FND2), location-aware (requirement FND1) and energy-constraint requirements (requirement FND3) of the fog and edge nodes and match them with the type of task/s that need to be met for the edge devices (requirement FND4).

2.5.3 Wi-Fi-based FND

Another method to discover fog services is to use a wireless connection or Wi-Fi technology. For example, Rejiba et al. [56] developed an approach to discover fog services while reducing energy consumption and using a context-aware scanning mechanism. The proposed approach predicts the fog service's presence in the vicinity based on the cellular footprint of the mobile device. After the fog service is detected, the stored surrounding information related to the cellular context of a mobile device is used as input to the machine learning algorithms to predict the presence or absence of the fog service in a particular location. Three machine learning algorithms, namely K-nearest neighbors, decision tree, and hidden Markov model (HMM) are evaluated to identify the best performance of the discovery process by keeping energy consumption to a minimum, thereby meeting requirements FND3 and FND4. However, the authors' proposed approach only takes into account a few parameters of context-awareness which relate to the mobile phone's historical cellular footprints such as the cellular tower ID and signal strength. It fails to take into account other important context-awareness parameters (in requirement FND2) such as the identity, capability, and location of fog services (in requirement FND1), etc. Although considering these parameters increases the energy consumption of the devices, it also enables a more precise search and leads to the discovery of appropriate fog nodes in the vicinity of the edge device/s. The authors conducted experiments over a small-scale fog service deployment, but further experiments need to be conducted to see if the results differ if applied to large-scale cellular data collection experiments to satisfy requirement FND5.

2.5.4 DNS and BGP community-based FND

This technique uses the Border Gateway Protocol (BGP) to discover fog services and the Domain Name System (DNS) is an auxiliary mechanism which is used as a decentralized database for fog sites to validate and enrich the discovery. Gedeon et al. [57] proposed an approach called Sunstone for the joint discovery and orchestration of fog services (fog computing resources). Sunstone combines three discovery mechanisms, namely (i) snooping of trace-route packets, (ii) DNS Naming Authority

Pointer (NAPTR) record information, and (iii) BGP community string advertisement. The actual discovery work takes place at the cloud side in a Kubernetes environment. The first mechanism is trace-route snooping which uses snooping to discover those fog services that are between the user and the cloud. First, trace-route determines all the packet paths from the user to the cloud and finds all the hops in the paths before Snooper detects the trace-route packets from the cloud to the user. Fog service snoopers receive a trace-route packet reply with information about the fog service. The second mechanism, DNS NAPTR record information uses DNS-based discovery to translate all IP addresses from the raw trace-route results to the respective autonomous system (AS) number using raw BGP update messages. Using PTR records, the network is resolved before using the NAPTR records. This enriches the network with information about the intermediate hops. The third mechanism is the BGP community string advertisement. It exchanges routing information between autonomous systems using the BGP as the standard protocol. BGP uses tables that contain a CIDR prefix and which use fog sites which are announced by a BGP peer on the Internet and community string. All BGP peers when sending the tag also send the CIDR prefix. This is to ensure that the information is made globally available to all fog sites. These tags are used to discover off-the-path fogs during the discovery process. This discovery process identifies the type of discovered fog, along with its distance from the cloud and the API endpoint to reach the discovered fog site. The proposed approach is evaluated in a real-world test bed, meeting requirement FND5. While the results show that this approach helps to reduce end-to-end latencies by matching to the edge device characteristics (meeting requirement FND3), the context-aware and location-aware characteristics of fog devices for discovery need to be considered to meet requirements FND1 and FND2. This will then result in the proposed approach finding the appropriate fog node according to the edge device requirements, thereby meeting requirement FND4.

2.5.5 Peer-to-Peer-based FND

Another category of approaches uses the hash table mechanism to find fog nodes in a distributed peer-to-peer (P2P) manner. Soo et al. [50] proposed a framework for FND based on the proximity of a mobile ad hoc social network (MASN) in a decentralized P2P manner. The mobile edge devices are connected to each other through a MASN and they monitor and gather information from MASN peers in the distributed hash table. When the edge device moves, the application from MASN gathers information about the fog nodes that are close to them and uses this information to access them. The GPS locations of the edge devices are used to find the nearest fog nodes. To reduce latency during the communication, the authors proposed using fog services that are within the location proximity, meeting requirement FND1. However, they do not consider an intelligent mechanism to carry out the discovery that incorporates other specifics such as context-aware requirements. So, requirements FND2 and FND4 are not met in [50]. Requirement FND5 is met since the proposed fog discovery approach was validated through simulations. However, the effectiveness of the proposed approach was not validated against other approaches. Santos et al. [59] presented a novel resource discovery service based on P2P distributed hash tables (DHTs) to enable automated resource discovery functionalities for IoT services. Information on all cloud or fog services is stored in the DHT. By storing this information, it is possible to know exactly which computing resources have been allocated to a particular instance because all the necessary provisioning information is available through the DHT. Protocols such as Kademlia and Pastry are shown to be appropriate candidates to provide flexible discovery process solutions during simulations in an OverSim simulator [82]. While the proposed approach assists in fog service discovery, it does not consider the location of fog nodes during that process. Furthermore, it does not consider context-aware characteristics when discovering appropriate fog nodes. Thus, requirements FND1, FND2 and

FND4 are not met in [59].

2.5.6 Static attribute table-based FND

Tomar and Matam [61] proposed an approach that discovers fog nodes based on a centralized static attributes table (SAT) which is located in the broker. SAT contains all the static information of edge devices which are stored and updated by the broker. The proposed approach consists of two layers, namely the lower and upper layers. The lower layer contains the source services which are edge devices or sensor services. The upper layer contains fog services and the broker. The fog service is a processing service which is responsible for finding the optimal source service for the assigned tasks. The broker is centralized storage to store the static attributes of the end devices in the SAT, which is used to find the optimal source services. The static attributes of the end devices consist of location, the identification data of end devices and the distance between end devices to all the fog nodes. The proposed approach discovers and finds fog nodes using various criteria, such as available processing capability, amount of free memory, and the distance between the fog node and the end device to ensure minimum delay in its processing. When the discovery process is initialized, the fog nodes query the SAT and the list of all source services is obtained. The processing service with minimum average distance and sufficient storage is selected as the optimal service for processing. However, the authors do not consider requirement FND2, which is considering the context-aware characteristics when discovering fog services. Furthermore, they fail to present the actual mechanism for carrying out the search based on the location requirement to meet requirement FND1. The applicability of the proposed approach is conducted over a small network of services that vary in size from 20 to 100. While this shows a reduction in energy consumption, this may increase when the number of fog nodes and the network size is increased. This proposed approach does not address requirements FND2 and FND4 while discovering fog nodes.

2.5.7 Exploration methods

Approaches in this category discover fog nodes by exploring information from other nodes which includes their neighbours. Karagiannis et al. [12] proposed an approach to discover new fog nodes and integrate them with other nodes in the system while considering characteristics such as fault tolerance, resource heterogeneity, proximity awareness, and scalability. To build fault tolerance, during the discovery process, the authors propose two mechanisms to discover other nodes that can take charge of the required computation when the current node fails. In the first mechanism, a node can make a request to join the network from a pre-existing fog node and it stores information about the nearby neighbors. In the second mechanism, a fog node may request to join the network from a pre-existing fog node, and it stores information on not only its neighbouring services but also those of the contact node. These neighbours are used when the contact node fails and the node in question needs to depend on others to complete the tasks. Using simulations, the authors show that the fog nodes improve the fault tolerance of a fog computing system significantly by storing information about the additional nodes of its neighbors during the discovery phase. However other aspects, such as improving resource heterogeneity, proximity awareness, and scalability of the nodes, are not considered. The authors evaluated the proposed approach using simulations over a small-scale network with 50-100 nodes. However, the feasibility of the results over large-scale networks needs to be seen. Skiadopoulos et al. [60] proposed a random walker mechanism to discover multiple fog nodes. The proposed approach uses a walker to visit one of its neighbor nodes to increase the coverage of the nodes which are found. While these approaches discover new fog nodes, they adopt a random approach rather than using an intelligent one according to pre-defined criteria such as FND1, FND2, and FND3 (to meet requirement FND4). Considering these criteria will assist them to consider the location-aware and context-aware requirements to meet FND1 and FND2 of the fog nodes while minimizing energy consumption to meet FND3. Table 2.4 compares the 13 approaches that we discussed under the category of FND and determines the gaps in meeting requirements FND1-FND5. In the next section, we discuss the approaches that come under the category of FNS and compare them against requirements FNS1-FNS4.

Table 2.4 : Comparison of the existing studies on FND against the requirements FND1-FND5

Research articles	**FND Requirements**				
	FND1	**FND2**	**FND3**	**FND4**	**FND5**
Soo et al. [50]	Yes	No	Yes	No	Simulation
Venanzi et al. [51]	No	No	Yes	No	Real network
Venanzi et al. [52]	No	No	Yes	No	Real network
Rejiba et al. [53]	No	No	Yes	No	None
Rejiba et al. [54]	No	No	Yes	No	Real network
Rejiba et al. [55]	No	No	Yes	No	Simulation
Rejiba et al. [56]	No	Yes	Yes	Yes	Real network
Gedeon et al. [57]	No	No	Yes	No	Real network
Venanzi et al. [58]	Yes	No	Yes	No	Real network
Santos et al. [59]	No	No	Yes	No	Simulation
Skiadopoulos et al. [60]	No	No	No	No	Simulation
Karagiannis et al. [12]	No	No	No	No	Simulation
Tomar and Matam [61]	Yes	No	Yes	No	Real network

2.6 Discussion of FNS approaches in addressing requirements FNS1-FNS4

FNS is an essential process in fog computing as it enables the best fog node to be selected to meet the required tasks of the edge device/s. Selecting the appropriate fog node also decreases overall end-to-end latency and energy consumption. From the literature, we found ten research articles that discuss an approach for FNS. We categorise them according to their working style in the following categories as shown in Figure 2.2:

1. MCDM-based FNS [65][67][78]
2. Fuzzy-based FNS [66]
3. Machine-learning-based FNS [71]
4. Deep-learning-based FNS [45]
5. Hybrid algorithms and objectives-based approaches [68][70][75][77].

2.6.1 MCDM-based FNS

Approaches in this category utilize multi-criteria decision-making (MCDM) techniques to select the best alternative services from the different options. For example, Nair et al. [65] proposed an approach which is a combination of the Analytic Hierarchy Process (AHP) and Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) to select the most appropriate VM to migrate the fog service that is predicted to overload. Mishra et al. [67] proposed an adaptive multi-criteria decision-making (A-MCDM) model to rank fog service resources in dynamic, distributed and scalable environments. The performance of A-MCDM is analyzed using mean absolute error (MAE), Spearman's rank correlation, response time (in seconds) and precision measure when it is compared with other MCDM methods such as the Simplified PROMETHEE-II and PROMETHEE-II. The experiment results of A-MCDM show that it helps in reducing the response time compared to other approaches. Baranwal and Vidyarthi [78] proposed the fog orchestrator nodes (FONs) approach to select nearby fog services that can act as an orchestrator. The authors defined the fog gateway node (FGN) as an orchestrator and the fog computational node (FCN) as a normal fog service that executes the applications. The aim of the approach is to select suitable FGNs as the FON. TOPSIS, which is one of the MCDM techniques, is used to select the suitable FGN as FON. Various aspects such as energy or battery power, computational power or capacity, number of FCNs connected to FGN, and the distance of the application from FGN are used in the selection process of the FON. The FON is responsible for collecting application requirements and offerings of the FCNs. It uses a placement algorithm to place an IoT application on a suitable FCN that acts as a temporary central hub between the connected FCNs and the applications. The evaluation of the FON selection model shows that it selects the most suitable FGN as the FON for the execution of the placement algorithm, thereby improving the overall performance of the system. While MCDM methods consider the different selection aspects, none of them consider the trustworthiness of the services (requirement FNS2) as one of the aspects during the selection process. Other approaches such as [67] and [?] failed to investigate the use of intelligent approaches (requirement FNS1) for fog service selection and determine if the output is recommended in a timely manner (FNS3). The case with [65] and [78] is similar. Nair et al. [65] and Baranwal and Vidyarthi [78] simulated their results using Xen Hypervisor and MATLAB respectively, however they have not been implemented on a real-world data set (requirement FNS4).

2.6.2 Fuzzy-based FNS

In this category, fog services are selected using the fuzzy logic technique. Jabri et al. [66] presented a gateway selection module that selects a suitable fog service for vehicles to access multi-access edge computing servers and clouds to reduce communication costs (e.g. bandwidth use, delay) and avoid network congestion. Fuzzy logic and multi-objective optimization, which are ant colony optimization techniques, are used to select the optimal gateways from the available ones based on different parameters such as the number of neighbors, LTE link, IEEE802.11p link quality, and length of stay. The authors evaluated the results of the study by simulations using NS3 [83] with the FuzzyLite library [84]. These approaches however do not consider the trustworthiness of the fog services in making an informed selection (thereby not meeting requirements FNS1 and FNS2). Thus, it needs to be determined if the performance of the proposed model will differ when a different trust parameter is used in the fuzzy selection of optimal gateways.

2.6.3 Machine-learning-based FNS

Li et al. [71] proposed an unsupervised algorithm that uses k-means clustering with Euclidean distance to group similar nodes into a cluster and determine its head. Clustering is applied by considering the distance of a fog node with the other nodes of the cluster and choosing the one which is closest to all of them. The simulated experiment results show that the unsupervised k-means algorithm can reduce the latency communication of the network. The authors validated the proposed model by considering 100 fog nodes. The effectiveness of the algorithm in giving timely output when there is an increased number of services available needs to be determined. Furthermore, the authors do not consider the trustworthiness of the fog node (requirement FNS2) when deciding whether to choose it or not by considering the constraints that it needs to satisfy.

2.6.4 Deep-learning-based FNS

Mostafa [45] proposed a cooperative fog system to select the optimal fog services that meet the user's preferences such as delay, cost, and privacy. The proposed model is termed the fog resource selection (FResS) algorithm which is a prediction model that uses the user's log files and past execution history to inform the system about incoming tasks. An artificial neural network is used with the execution history to predict the best fog resources that suit the user's requirements. The FResS algorithm contains an execution log, which maintains the performance data of the fog services. When a new job enters the fog system for execution, its run-time and required resources are predicted with the help of these execution logs which, in turn, assist in selecting the best fog service. The authors simulate the model in the GridSim simulator [85] and the results show a very high level of accuracy and one that has low overheads. Moreover, the experiments showed that resource utilization had an inverse effect on the cost, response time, and bandwidth usage. However, the model does not consider the trustworthiness of the fog services (requirement FNS2) during the recommendation process.

2.6.5 Hybrid algorithms and objectives-based approaches

Other methods have been used to select the optimal fog nodes. For example, Rejiba et al. [68] modelled the FNS problem as a multi-armed bandit problem with delay minimization. In fog computing, due to user mobility, the availability of fog services is dynamically changed which leads to a high switching cost from one fog service to the other. To reduce this high switching cost, the authors proposed a block-based fog service selection scheme which reduces the occurrence of switching by keeping the same fog service selected during a block of timeslots. An adaptive greedy approach was used to select the fog service that has the minimum average delay. If this is not possible, then the user is allowed to perform an exploration step to find fog services offering lower delays. The experiments showed that a block-based fog service selection scheme reduced the switching costs, and the proposed adaptive greedy approach improves the overall system performance. However, the authors failed to investigate an intelligent algorithm (requirement FNS1) to carry out the trust-based selection for fog services. Iyer et al. [75] proposed the Cloud and Fog Broker Arbitrage mechanism to select an appropriate cloud service or fog service. The proposed model uses different parameters such as the user's risk nature, price, reputation of fog services and cloud services for the decision process which is done by the broker which can either be in the cloud or in the fog. The results show that apart from the trust value, parameters such as delay, price and risk can affect the decision to choose a cloud or fog service. The trust value is calculated based on the user's experience with fog service providers (FSPs) and cloud service providers (CSPs). The authors conducted simulation experiments (requirement FNS4) that consist of a small number of services, however it is not applied in a distributed fog environment to evaluate its effectiveness in the trust-based selection process. Furthermore, this proposed approach does not address requirements FNS1 and FNS3. Pan et al. [70] proposed an approach to select suitable fog services that minimizes the latency in transmitting data and the rental cost for providing resources. They proposed the novel FNS algorithm based on Simulated Annealing (FSSA) to find enough fog nodes to meet the deployed requirements of a single service while minimizing the overheads of end-users. This aim is achieved by firstly modelling the problem of selecting fog nodes as a knapsack problem before using the FSSA algorithm for service selection. The output of the FSSA algorithm is compared with other FNS approaches such as random greedy algorithms in which it is shown to perform better with reduced overheads for the end-user while achieving the deployment of service requirements. Rahman et al. [77] proposed three methods, namely Shortest Estimated Latency First (SELF), Shortest Estimated Buffer First (SEBF) and random selection, to select fog nodes. The SELF scheme is used to select the fog node that has the shortest latency to transfer data from the end user to the fog node. The end user is responsible for maintaining the latency of the transmitting data from the user to the fog nodes. The end user (such as the sensor) calculates the latency between itself and the other fog nodes in its coverage. The SEBF scheme aims to select the fog node for the new upcoming request from the end user which has the minimum number of tasks buffered in the queue. The random selection method selects the fog node randomly. The three methods are evaluated and the results show that the SELF method helps to decrease energy consumption significantly while the SEBF method helps to reduce the risk of losing data. However, SELF will lead to packet loss and SEBF will increase energy consumption in some cases. Also, the random selection method increases the energy consumption and packet loss compared to the other methods, SELF and LCBF. The proposed approach achieved efficient performance in relation to fog nodes by reducing energy consumption and packet loss and increasing the hit ratio at the fog node (meeting requirement FNS3). However, they do not consider the trustworthiness aspect of the service in the selection process (requirement FNS2).

Table 2.5 compares the 10 approaches that are discussed under the category of FNS and determine the gaps against requirements FNS1-FNS4. In the next section, we critically analyse the existing approaches and present the open gaps and areas of future work.

2.7 Open issues and future directions

In this section, we summarise the gaps in the existing approaches on FND and FNS and determine areas of future work to address them. We also explain in Figure 2.3 how the proposed areas of future work assist in achieving requirements FND1-FND5 and FNS1-FNS4 as previously discussed.

Table 2.5 : Comparison of the existing studies on FNS against the requirements FNS1-FNS4

Research articles	**FNS Requirements**			
	FNS1	**FNS2**	**FNS3**	**FNS4**
Nair et al. [65]	No	No	No	Simulation
Jabri et al. [66]	Yes	No	No	Simulation
Mishra et al. [67]	No	No	Yes	Real network
Rejiba et al. [68]	No	No	Yes	Simulation
Pan et al. [70]	No	No	Yes	Real network
Li et al. [71]	Yes	No	Yes	Simulation
Mostafa [45]	Yes	No	Yes	Simulation
Iyer et al. [75]	No	Yes	No	No
Rahman et al. [77]	No	No	Yes	Simulation
Baranwal and Vidyarthi [78]	No	No	No	Simulation

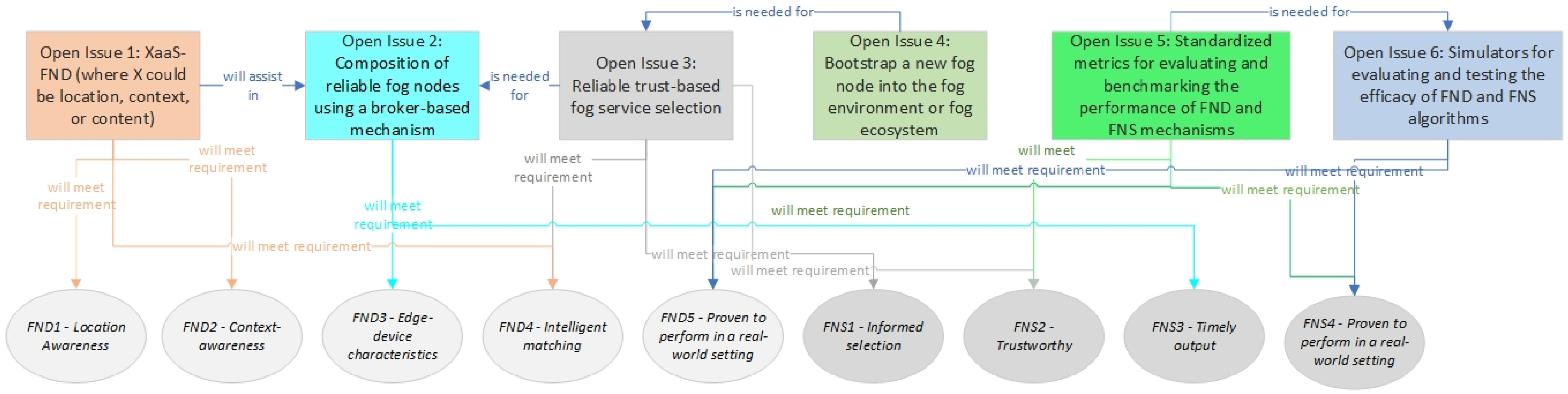

Open Issue 1: XaaS-FND (where X could be location, context, or content)
will assist in
Open Issue 2: Composition of reliable fog nodes using a broker-based mechanism
is needed for
Open Issue 3: Reliable trust-based fog service selection
is needed for
Open Issue 4: Bootstrap a new fog node into the fog environment or fog ecosystem
Open Issue 5: Standardized metrics for evaluating and benchmarking the performance of FND and FNS mechanisms
is needed for
Open Issue 6: Simulators for evaluating and testing the efficacy of FND and FNS algorithms
will meet requirement
will meet requirement
will meet requirement
will meet requirement
will meet requirement
will meet requirement
will meet requirement
will meet requirement
will meet requirement
will meet requirement
will meet requirement
will meet requirement
will meet requirement
FND1 - Location Awareness
FND2 - Context-awareness
FND3 - Edge-device characteristics
FND4 - Intelligent matching
FND5 - Proven to perform in a real-world setting
FNS1 - Informed selection
FNS2 - Trustworthy
FNS3 - Timely output
FNS4 - Proven to perform in a real-world setting

Figure 2.3 : Describing how the proposed areas of future work assist in meeting requirements FND1-FND5 and FNS1-FNS4

2.7.1 XaaS-FND (where X could be location, context, or content)

Although researchers have proposed different methods for FND, they fail to consider the context-aware, location-aware, content-aware mechanism characteristics through which they should be discovered. The use of context-aware mechanisms has also been explored in Web services and cloud selection areas to improve QoS discovery [86][46], but it is not considered a requirement of all fog discovery approaches. Nejad et al. [87] discussed context-aware fog computing systems but they did not use it for fog node discovery. However, as emphasized earlier, context awareness in fog computing assists in the correct use of the requirements of the user that will assist in providing information on the service that has the best ability to provide these requirements [11]. Thus, it should be an important criterion to consider in FND. Furthermore, while location-aware mechanisms have been studied widely in pervasive computing environments [88][89][90] and content-aware mechanisms have been discussed in Web services [91], they have not been applied to the different approaches for FND. One possible option to address this gap is to approach this as XaaS-FND, in which goal-specific services are developed that assist the edge devices in evaluating the available fog nodes according to the location, context and content of their requirements (meeting requirements FND1, FND2 and FND4). These services can be integrated and federated to develop a fog service composition model, as explained next.

2.7.2 Composition of reliable fog nodes using a broker-based mechanism

Most of the work on FNS is focused on selecting a single fog node. There is no work on fog service composition that could be a combination of multiple fog nodes that need to work together to solve and deliver the requirements of a single edge device. This is a major challenge that requires researchers to develop an architecture of the fog registry consortium module that can host information and the characteristics of fog nodes. Such a consortium module can act as an intermediary between the edge nodes and the fog nodes and as a broker for various activities such as, but not limited to, fog discovery, fog selection, etc. Having a hub and spoke model of a main central fog registry and various distributed fog registries will ensure that the specific capabilities of a fog node and its trustworthiness in committing to the promised expectations as required by the edge device (meeting requirement FND3) can be captured and used in making an informed selection. It will also assist in ensuring the concurrency of synchronizing data between distributed fog registries and distribute them across the globe in a uniform manner according to their geographical boundaries. A key aspect to consider for fog node composition is interoperability among individual fog nodes. Mouradian et al. [25] discussed the issue of interoperability and analysed whether the existing literature considered this or not in fog computing. The authors concluded that as part of a federated system, an application can be executed spread over different fog nodes. However, from an architectural perspective, this implies the need for appropriate signalling and control interfaces, as well as appropriate data interfaces to enable interoperability at different architectural modules. While a few studies such as [92][93][94] discuss the need to have interoperability in forming a fog node, more research is needed to incorporate the FND and FNS requirements mentioned earlier along with how to describe a fog service, how to manage it etc.

2.7.3 Reliable trust-based fog node selection

Forming a reliable trust-based FNS has two requirements. The first is to ensure that the trustworthiness of the fog nodes is taken into consideration when deciding if they are the best in meeting the edge devices's requirements. The second is to ensure that the trust value of a fog node is not easily manipulated, nor does it contain bias. A number of approaches have been proposed for trust-based service selection in the cloud and in Web services [95][96][97][98][99][100]. However, in the area of trust-based FNS, the application of these approaches is still in its infancy. Furthermore, the complexity of this issue is compounded by the fact that in the completely distributed environment of fog computing, there is no single repository that stores the trust values of fog nodes. Thus, the trust values of a service can be manipulated, for example negative feedback can be given on good services by fog nodes or ballot-stuffing can be used to give positive feedback to bad services from fog nodes. There is no work in the existing literature to counter these mechanisms and ensure the reliability of the underlying trust values. Intelligent algorithms need to be developed to assist in achieving these aims and make fog-based selection more reliable and meet requirements FND4, FNS1 and FNS2. Researchers can look at mechanisms such as blockchain to ensure the reliability of trust values.

2.7.4 Bootstrap a new fog node into the fog environment or fog ecosystem

A key issue with reputation systems is that they are unable to rank new service providers objectively during the process of service selection. This disadvantages new fog nodes as the lack of previous ratings renders them ineligible for trust-based ranking. This issue is called the cold-start problem which is an important and challenging issue for a reputation or trust system. In Web services, many studies propose solutions to the cold-start problem. For example, Tibermacine et al. [101] proposed a reputation mechanism based on the initial QoS attributes of the new Web service and its similarity with other Web services that have extensive feedback records. They used regression models to estimate the new reputation value of the new Web service from the known QoS attributes. Another approach in the online environment [102] is clustering the new user to similar users. However, this research assumes that a user can trust other users within its cluster. This can be done if the trust values of existing users are available. Moreover, the recommendation system suffers from the cold-start problem for a new user or item where the system does not have any information about the user's or item's preferences to make good recommendations. Lika et al. [103] proposed a classification method to find users who are similar to a new user using the C4.5 Tree. They classified the new user to a category of people based on their demographic data. They applied the nearest neighbor algorithm to find the new user in the category. After this, the rating of the user for n items is predicted based on the ratings of the neighbors using a weighted sum of the ratings made by the neighborhood users. An open research issue in using the trust values of fog nodes in its selection is to address this gap of bootstrapping in a cold-start scenario. This is necessary to address the previous open issue and meet requirements FND4, FNS1 and FNS2. Another open research issue is to develop distributed mechanisms to ensure concurrency between the distributed fog registries within the consortium to ensure that the most updated trust values are reflected across them.

2.7.5 Standardized metrics for evaluating and benchmarking the performance of FND and FNS mechanisms

From the analysis detailed in Section 2.5 and 2.6, it can be seen that several authors have used various evaluation approaches such as simulation or real-world data to demonstrate the performance of their models. However, there is no single source of truth or benchmark mechanism that can be used by different approaches for evaluation. This needs to be both in terms of the metrics used for the evaluation of different algorithms and for benchmarking their performance. In the distributed fog computing literature, the platforms are not consistent which results in a lack of a standard data set for evaluation, causing bias in the result. Future research should look at developing such metrics and benchmarks that will assist in a standardized comparison of the different approaches. This will assist in addressing the next open issue and requirements FND5 and FNS4.

2.7.6 Simulators for evaluating and testing the efficacy of FND and FNS algorithms

The FND and FNS approaches should be proven to work in a real-world environment either through simulation experiments or real-world experiments. In this regard, researchers have proved the applicability of their proposed approaches in a simulator. For example, Soo et al. [50] evaluated DTN routing and application protocols using the Opportunistic Networking Environment (ONE) simulator [104]. The proposed simulator allows users to create scenarios based on different synthetic movement models and real-world traces and offers a framework for implementing routing and application protocols. Rejiba et al. [55] used the Mininet-WiFi emulator [81] to emulate the proposed approach of the discovery of a wireless network signal range using the 802.11 beacon technique to solve the problem of unnecessary scans in the Wi-Fi-based discovery process which causes energy consumption penalties. Promising emulation results in terms of energy consumption, total time of scanning, and discovery rate were obtained as a result of using the simulation. Santos et al. [59] used the OverSim simulator [82] which is an open-source flexible overlay network simulation framework based on OMNeT++ to support P2P protocols. The simulator was used to simulate the proposed P2P system based on DHTs to enable automated resource discovery functionalities for IoT services. Jabri et al. [66] used NS3 [83] with the Fuzzylite library [84] to simulate how to select the optimal gateways and connected fog nodes in the vehicular fog computing environment. Mostafa [45] used the GridSim simulation [85] to simulate the discrete-event grid simulation toolkit. The toolkit supports the modeling and simulation of heterogeneous grid resources (both time and space-shared), users and application models.

Researchers have also built their own simulators using different programming languages to evaluate their work. For example, Skiadopoulos et al. [60] simulated the work on Python 3.6.3, using the SciPy and NumPy libraries [105]. Karagiannis et al. [12] and Rahman et al. [77] built a simulator using JAVA while Rejiba et al. [68] used Python and Baranwal and Vidyarthi [78] used MATLAB. However, none of the existing research develops a simulator to evaluate the efficacy and effectiveness of FNS and FND techniques by capturing their specific requirements. As supported by [106], the fog computing simulation tool needs further extension and more attention is needed to maintain software quality. This is an open gap that needs to be addressed in the literature to satisfy requirements FND5 and FNS4.

2.8 Conclusion

Fog computing as a platform has been used to solve the different limitations of cloud computing. By bringing the processing, storage, and networking to the edge of the network, fog computing assists in decreasing latency, network bandwidth, response time, communication costs and saves the energy used by network-constrained resources. However, to achieve this, the efficient discovery and selection of fog nodes according to the requirements of the edge devices is key. In this chapter, we provided details of a systematic literature review of the proposed FND and FNS approaches in the literature to identify their style of working and determine the gaps for future work. Our analysis identified only 23 articles in the literature that discuss FND and FNS. We critically analyzed and compared these articles against the requirements and identified the gaps to be addressed. In our future work, we aim to develop a conjoint framework for fog node/s discovery and selection that addresses the aforementioned gaps. Having a trustworthy and dynamic model for FND and FNS will also assist in the envisaged Osmotic computing approach [107] while deciding which services can be executed at the edge rather than at a data centre.

Chapter 3

Problem Definition

3.1 Introduction

The first chapter highlighted the importance of FND and FNS in the context of fog computing. In the previous chapter, we presented an extensive review of the existing literature. Based on the literature review, we noted in Chapter 2 that none of the existing proposals offer an intelligent methodology for fog node discovery and selection. Additionally, in the previous chapter, we identified five shortcomings in the existing literature that need to be addressed to propose a complete methodology for FND and FNS. The shortcomings in the existing literature are described in Section 2.7. In this chapter, we formally define and present the problem that we address in this thesis.

This chapter is organised as follows: in Section 3.2, we propose a set of definitions of those key terms and concepts that will be used while defining the problem in Section 3.3. Section 3.4 presents the research issues. In Section 3.5, we outline the research questions. In Section 3.6, we present the research objectives. Section 3.7 presents the research approach to problem solving. Finally, Section 3.8 concludes the chapter.

3.2 Key Concepts

3.2.1 Fog computing

Fog computing is defined as a distributed computing paradigm that extends cloud computing capabilities to the edge of the network, providing a decentralized and hierarchical architecture for deploying computing resources closer to the data source. By utilizing edge devices, access points, and fog nodes, fog computing reduces network latency, enhances real-time processing, and enables efficient data management [25].

3.2.2 Fog node

A *fog node* is defined as a distributed computing device that is located in the fog layer as an intermediate layer in a fog computing architecture. Fog nodes can be virtual like virtual machines or physical such as any device with sufficient capacity, computing power, and energy power, such as routers, switches, servers, and access points. The processing, analysis and management of data are done by these virtualized or non-virtualized devices at the edge of the network where they are often deployed in close proximity to end-users and devices to reduce latency and save bandwidth for real-time applications [14] [25].

3.2.3 Fog service

We define a *fog service* as a computing resource that is provided by a fog node provider to fog consumers through the internet. The fog service is defined as an association between an edge device and a fog node in which the edge device uses the fog node's computing resources to process its task in a low-latency and high-bandwidth manner.

3.2.4 Fog node provider

We define a *fog node provider* as any organization or company that provides fog computing services to end-users. These providers are responsible for operating, managing, and maintaining the reliability, availability, and security of the fog nodes and services they offer.

3.2.5 Fog consumer

We define a *fog consumer* as any end entity that connects to a fog node and uses the fog service. The entity could be an individual user, application or edge device such as an IoT device [14].

3.2.6 IoT device

An *IoT device* is defined as a physical object or a digital representation of a physical object that is capable of being uniquely identified and is capable of sending or receiving data over the internet or other network without human intervention [108].

3.2.7 Fog Registry

We define a *fog registry (FR)* as a fog node that has large storage and high processing and computing capacities. The FR maintains information about all fog services in the network and controls fog nodes in the vicinity.

3.2.8 Distributed Fog Registry

We define a *distributed fog registry (DFR)* as a collection of fog nodes that collectively hold information about all other fog nodes in a distributed manner. The distributed fog registry works with each other in a peer-to-peer manner.

3.2.9 Central Fog Registry

We define a *central fog registry (CFR)* as a fog node or a collection of fog nodes that controls and coordinates all the distributed fog registries.

3.2.10 Fog Registry Consortium

We define a *fog registry consortium (FRC)* as a finite collection of fog registries (FR) that manages and controls fog nodes and maintains their information. Conceptually, FRC may be defined as follows: FRC = {FR1, FR2, FR3, FR4, ... FRn}. Each consortium is set up to serve a specific objective and may have specific criteria which fog nodes have to meet to become a part of the consortium. In this research, we regard the FRC as an intermediary between the fog consumer and fog nodes that acts as a broker for various activities such as, but not limited to, fog discovery, fog selection, etc.

3.2.11 Fog Node Discovery

We define *Fog node discovery (FND)* as a process of finding the relevant fog node that satisfies the end user's requirements. Fog node discovery is a process that allows end users to locate the most suitable fog nodes to execute their applications or services.

3.2.12 Fog Node Selection

We define *Fog node selection (FNS)* as the process of choosing the best-suited fog node/s from those available based on the end users' required criteria to perform their job.

3.2.13 Context-aware computing

Context-aware computing is defined as the ability of an application or a system to discover and adapt the situational context of the user in the environment (i.e., location, time, identity of people, nearby devices, objects and environmental factors) [11].

3.2.14 Trust

In the context of fog computing and our research, *trust* is defined as the belief that a fog consumer (trusting agent) holds in the willingness and capability of a fog node or service provider (trusted agent) to deliver a mutually agreed-upon service within a specific context and time frame [48].

3.2.15 Trust value

Trust value is defined as the degree of trust or confidence that is assigned to a specific entity within the fog computing environment. It is typically represented as a numerical or probabilistic value, indicating the level of trust placed in the entity. The trust value is often calculated based on various factors such as historical data, QoS data, reputation, perceived security measures, reliability metrics, and feedback from other users or monitoring systems [48].

3.2.16 Trusted fog node

We define a *trusted fog node* as a fog node that has been verified to be reliable and capable of performing its intended functions.

3.2.17 Cold-start problem

We define the *cold-start problem* as a challenging scenario in which a system encounters difficulties in making accurate predictions or providing effective recommendations for new or previously unseen entities. It arises when there is insufficient or limited data available about these entities, making it difficult to understand their characteristics, preferences, or behaviors. In the context of fog computing, the cold-start problem occurs when new fog nodes join the network and there is a lack sufficient information or historical data from which to assess their quality of service, reliability, or trustworthiness.

3.2.18 Bootstrapping

In the context of a fog reputation system, *bootstrapping* refers to the process of initializing or assigning trust values to newly joined fog nodes that have no prior reputation history within the system. When a fog node joins the network for the first time, it lacks a reputation score or trust value, which is crucial for making informed decisions about its integration and interaction within the fog computing environment. Bootstrapping mechanisms aim to overcome this challenge by assigning initial trust values to these new fog nodes based on various criteria or metrics. These criteria may include the fog node's QoS attributes, such as latency, reliability, bandwidth, or energy efficiency, as well as other relevant factors like its location, resource availability, or previous interactions with other entities. By assigning initial trust values through bootstrapping, the fog reputation system can begin to assess and monitor the trustworthiness and reliability of new fog nodes, facilitating their integration and decision-making processes within the fog computing network.

3.3 Problem Definition

Through the comprehensive examination of the literature in Chapter 2, it becomes apparent that the prevailing studies do not address the discovery process in a context-aware or intelligent manner. In the existing literature, context-aware FND is not studied in a scalable manner. It also does not present a framework for intelligent and reliable trust-based assessment in a distributed fog environment or explore a trust-based FNS that depends on distributed registries. Finally, the existing literature doesn not discuss an intelligent approach to select the optimal fog node from several discovered fog nodes.

Within the scope of this research, our focus lies in the intelligent discovery and selection of fog nodes with the aim of improving performance by reducing latency, response time, and bandwidth usage. Specifically, the fog consumer initiates the discovery process to identify fog nodes with the computing capacity that satisfies the requirements of the fog consumer. Following this, the selection stage commences when the edge device evaluates the trustworthiness of the identified fog nodes and selects the most suitable one that meets its task requirements.

In other words, the fog consumer selects a fog node based on the belief that it will successfully process and deliver the required outcomes within the specified time frame. Therefore, the proposed intelligent system is expected to employ an optimal approach to discover the most suitable fog nodes and support fog consumers in selecting the most trustworthy one.

3.4 Research Issues

- **Research Issue 1:**

  The fog environment lacks a consortium for fog nodes to host important fog node information and a distributed mechanism to ensure DFR concurrency.

- **Research Issue 2:**

  A context-aware FND mechanism using intelligent methods does not exist. In the scope of this research, the word context refers to the identity and location of the fog consumer. In the future, other research may choose to enrich context-aware discovery by adding additional context-related parameters. In the existing literature, no study has been conducted that examines scalable and Intelligent context-aware FND.

- **Research Issue 3:**

  Existing work lacks a framework for intelligent and reliable trust-based assessment in distributed fog environments, failing to explore the potential of trust-based FNS that leverages distributed registries. Additionally, the selection mechanism for fog nodes currently does not incorporate trust-based criteria, underscoring the need for innovation in this area.

- **Research Issue 4:**

  In a fog environment, mechanisms for predicting the trust value of new fog nodes are absent. Existing literature does not examine how to intelligently predict a fog node's trust value with the aim of bootstrapping it in the fog ecosystem.

3.5 Research Questions

The research gaps and issues highlighted earlier have led to the formulation of the principal research question for this project, which is: How can the fog consumer intelligently discover and select a fog node in a context-aware and reliable manner? This main question can be further divided into five sub-questions as follows:

- **Research Question 1:**
  How to register fog node information in the FRC and ensure concurrency among the DFRs?
- **Research Question 2:**
  How can fog consumers discover fog nodes in a context-aware manner?
- **Research Question 3:**
  How to predict the trust value of a fog node to make reliable fog-based selection decisions?
- **Research Question 4:**
  How to predict the trust value of a new fog node for trustworthy and reliable fog-based selection?
- **Research Question 5:**
  How to evaluate and validate the proposed framework including the proposed solutions to address Research Question 1 to Research Question?

3.6 Research Objectives

Based on the research questions in the previous section, we define five research objectives that will be achieved in this thesis. The objectives of this research are as follows:

- **Objective 1:**

  Develop the architecture of the FRC that can host the important information of fog nodes. Furthermore, develop distributed mechanisms to ensure concurrency between the DFRs within the consortium.

- **Objective 2:**

  Develop an intelligent approach to discover fog nodes in a context-aware manner.

- **Objective 3:**

  Develop an intelligent approach to predict the trust value of fog nodes to help the user select a reliable fog node based on the trust value.

- **Objective 4:**

  Develop an intelligent approach to bootstrap new fog nodes into a fog ecosystem to make a reliable fog-based selection.

- **Objective 5:**

  Build a prototype system to evaluate and refine the framework and the approaches developed to achieve sub-aims (1) – (4).

3.7 The research approach to problem solving

In this thesis, the design science research methodology is chosen to solve the research questions proposed in Section 3.5. The design science research methodology is the most suitable research methodology for this thesis. Peffers et al. [109] defined the design science research methodology as the process to create and evaluate IT artifacts that intend to solve problems. The main process of the design science research methodology is to design artifacts to solve observed problems, to make research contributions, to evaluate the designs, and to communicate the results to appropriate audiences [109]. This thesis focuses on developing a new methodology for intelligently discovering and selecting the optimal fog node. This new methodology will address the defined problems of FND and FNS. Thus, this thesis follows the design science research methodology approach to make the methodology more scientific and rigorous. The details of the chosen research method are presented in the next section.

3.7.1 The design science research methodology

In this thesis, a design science research methodology [109] approach is chosen as the research method for the proposed solution development. The process of this research method is depicted in Figure 3.1. The process comprises six phases discussed in the following sub-sections, as defined in [109]:

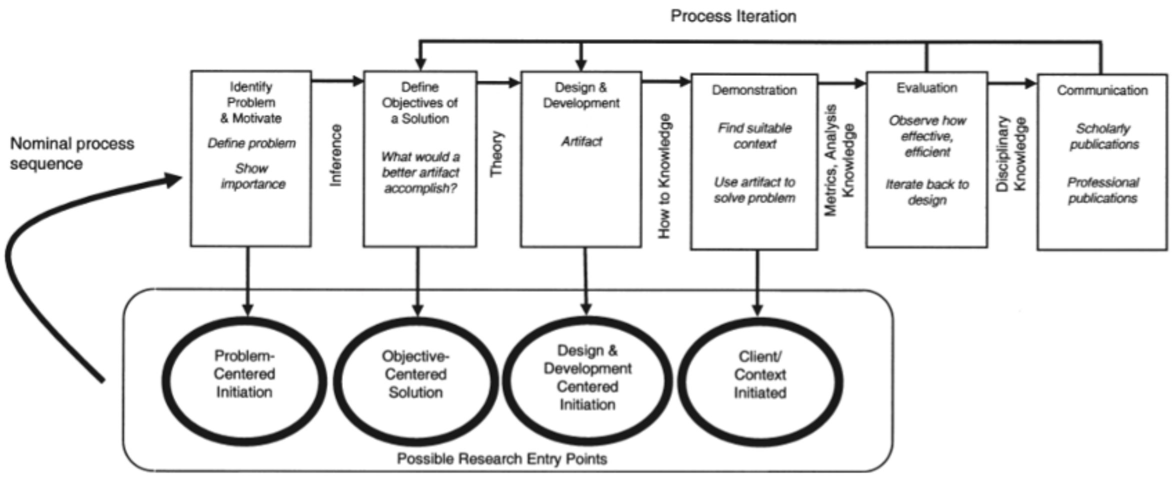

Process Iteration
Nominal process sequence
Identify Problem & Motivate
Define problem
Show importance
Inference
Define Objectives of a Solution
What would a better artifact accomplish?
Theory
Design & Development
Artifact
How to Knowledge
Demonstration
Find suitable context
Use artifact to solve problem
Metrics, Analysis Knowledge
Evaluation
Observe how effective, efficient
Iterate back to design
Disciplinary Knowledge
Communication
Scholarly publications
Professional publications
Problem-Centered Initiation
Objective-Centered Solution
Design & Development Centered Initiation
Client/ Context Initiated
Possible Research Entry Points

Figure 3.1 : A design science research methodology process model

3.7.1.1 Problem identification and motivation:

The first phase of the design science research methodology approach is to identify the research problems. This step will help to design the artifact and solve observed problems. In this thesis, we conducted an extensive study of the existing literature, as presented in Chapter 2, to identify the research problems and gaps. Subsequently, these research problems are thoroughly outlined and elucidated in Chapter 3.

3.7.1.2 Define the objectives for a solution:

During this phase, the research objectives of a solution are defined. Chapter 4 provides an extensive overview of the solution for all five issues identified within the scope of this study and outlines how the research questions will be solved focusing on developing an intelligent framework on top of fog computing for context-aware FND and trust-based FNS.

3.7.1.3 Design and development:

During this phase, the focus is on designing the research solution and the proposed prototype. Chapters 5, 6, 7, and 8 delve into the intricate process of designing and developing a prototype for FND and FNS. Extensive efforts are dedicated to engineering prototype systems, ensuring their effectiveness and functionality. Furthermore, several case studies are carefully developed to serve as valuable testing grounds for evaluating the proposed prototype. These case studies enable a comprehensive assessment of the performance and feasibility of the developed solution.

3.7.1.4 Demonstration:

In this phase, the objective is to showcase how the prototype effectively addresses the research problems. Chapters 5, 6, 7, and 8 detail the experiments which were conducted to validate the proposed prototype to address the identified problems.

Through extensive experimentation and analysis, the efficacy and functionality of the proposed prototype is demonstrated, providing empirical evidence of its effectiveness in resolving the research problems.

3.7.1.5 Evaluation:

During this phase, the focus is on assessing the proposed prototype and evaluating its performance to determine how effectively it supports the efficiency of the overall solution. Once the prototype has been tested, the evaluation and validation of our proposed approach is undertaken in the evaluation step which is elaborated upon in Chapters 5, 6, 7, and 8.

3.7.1.6 Communication:

In this phase, the focus is on communicating the result of the proposed prototype to researchers and the professional community. The results of the proposed prototype are published in journals and conference proceedings.

3.8 Conclusion

In this chapter, we defined important key terms and concepts that hold significant relevance to this thesis. These definitions ensure a shared understanding of the terminology used throughout the thesis, enabling effective communication and comprehension. Moreover, we identified the main research problem that serves as the focal point of our investigation. By addressing this problem, we aim to contribute to the field of fog computing and provide valuable insights into FND and FNS methodologies. To guide our research process, we formulated research questions and established research objectives. These questions and objectives act as guiding principles, shaping the trajectory of our inquiry and driving us towards developing innovative solutions. In line with our research objectives, we adopted a design science research methodology, which provides a structured framework for creating and evaluating practical artifacts. This methodology ensures that our research outcomes are not only theoretically grounded but also capable of addressing real-world challenges in the domain of FND and FNS. In the next chapter, we present an overview of our proposed solution to the research problem outlined in this chapter. We detail our innovative approach, the FRC framework, providing insights into its key features and contributions.

Chapter 4

Solution Overview

4.1 Introduction

In this chapter, we examine the methodological approach that is utilized to address the gaps identified during the literature review. Additionally, we overview the solutions proposed for each of the five research issues that are discussed in Chapter 3. The chapter details the solutions for all five issues presented in this thesis, along with an explanation of how the research questions are addressed. In Section 4.2, we provide a general overview of the proposed solution. Sections 4.3 through 4.7 present an overview of the solutions for research questions 1 to 5, respectively. Lastly, Section 4.8 concludes the chapter.

4.2 Overview of the solutions

In this section, we present an overview of all the solutions to achieve the five objectives listed in Chapter 3. The objective of this research is to develop an intelligent framework on top of fog computing for context-aware FND and trust-based FNS. In our framework, we propose a novel concept of the FRC. We define the FRC as an intermediary between the fog consumer and fog nodes that acts as a broker for various activities such as, but not limited to, fog discovery, fog selection, etc. In our research, we propose and use the notion of FRC for FND and FNS. In the future, the role of FRC can be extended beyond this to carry out other activities. The definition and the role of FRC is explained in Chapter 5. The FRC is a collection of fog registries and consists of two types of fog registries as follows: (see Figure 4.1).

1. The central fog registry (CFR) (further details in Chapter 5 in Section 5.3).
2. The distributed fog registry (DFR) (further details in Chapter 5 in Section 5.3). The DFR has the following modules:
    (a) Fog repository (further details in Chapter 5 in Section 5.3)
    (b) Fog node discovery engine (FNDE) (further details in Chapter 6)
    (c) Fog node selection engine (FNSE) which includes:
        i. Trust evaluation engine (TEE) (further details in Chapter 7)
        ii. Bootstrapping engine (BE) (further details in Chapter 8)

The following steps describe the overall process of the research (solution steps) and the function of each component in a methodological manner:

Step 1: Any given fog node registers (or publishes) its service information such as fog node ID, fog node name, fog node description, fog node location, and QoS values to the DFR. This information is stored in the fog repository.

Step 2: The DFR pushes fog node data to the CFR. This activity is event-based and is carried out when there are updates to the underlying information (adding new information, updating the information etc.).

Step 3: The CFR broadcasts this data to all the DFRs in the FRC.

Step 4: As a result of Step 3, all the DFRs have the same up-to-date data on all fog nodes, which maintains the consistency of the data and the concurrency between all the DFRs in the FRC.

Step 5: The FRC manages the connection between the fog consumer and one of the DFRs based on the fog consumer's geographical location. Depending on the physical location of the fog consumer, they intelligently connect to the nearest fog registry. Further details of steps 1 to 5 can be found in Chapter 5.

Step 6: The fog consumer initiates the discovery process by sending their context-aware data (identity and location) to the DFR. These parameters are used to carry out context-aware FND.

Step 7: The FNDE in the DFR verifies the fog consumer's credentials (authentication) and then collects context-aware data, this being the physical location of the fog consumer. After this, the FNSE intelligently determines the nearest fog node based on a similarity measure. Further details of steps 6 and 7 can be found in Chapter 6.

Step 8: Based on step 7, the FNDE nominates the most optimal fog nodes to the fog consumer based on location-based context awareness.

Step 9: The fog consumer selects one of the candidate fog nodes or seeks help from the FNSE to select a reliable fog node. In our research, we use the notion of trustworthiness to select a fog node.

Step 10: The FNSE uses the TEE in the DFRs to predict the trust value of the candidate fog nodes based on the QoS factors of fog nodes. In our research, we regard the aggregated QoS value of a given fog node as its trustworthiness value. Chapter 7 provides further information on the working of the TEE.

Step 11: If the fog node is new and has no QoS data, the BE will predict the trust value of the new cold-start fog node. Chapter 8 provides further information about the working of the BE.

Step 12: The FNSE ranks the candidate fog nodes based on their trust values in ascending order.

Step 13: The fog consumer makes the final decision and chooses one of the candidate fog nodes.

Figure 4.1 illustrates the framework of the FRC with the proposed FND and FNS approach and Figure 4.2 illustrates the solution steps.

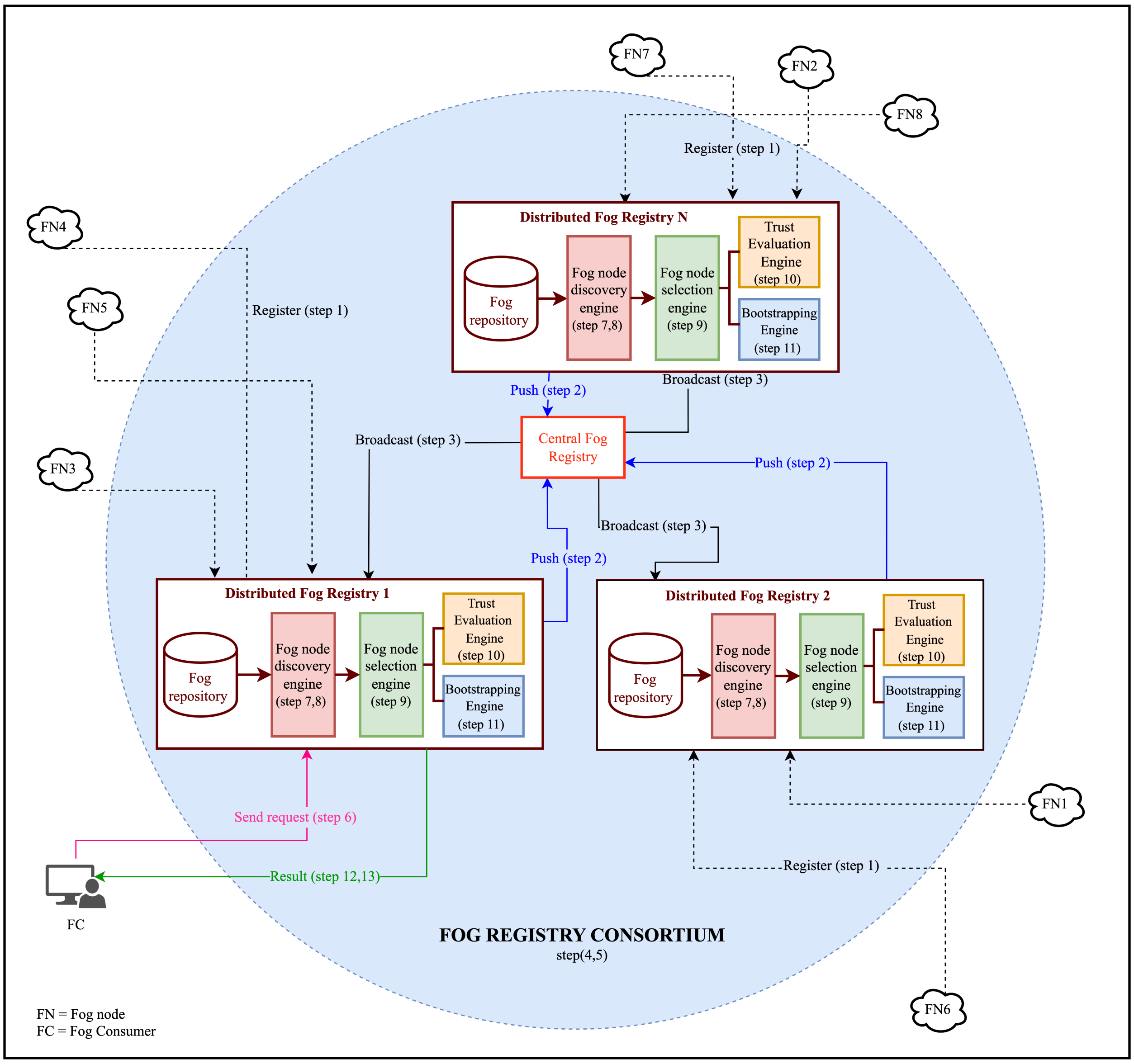
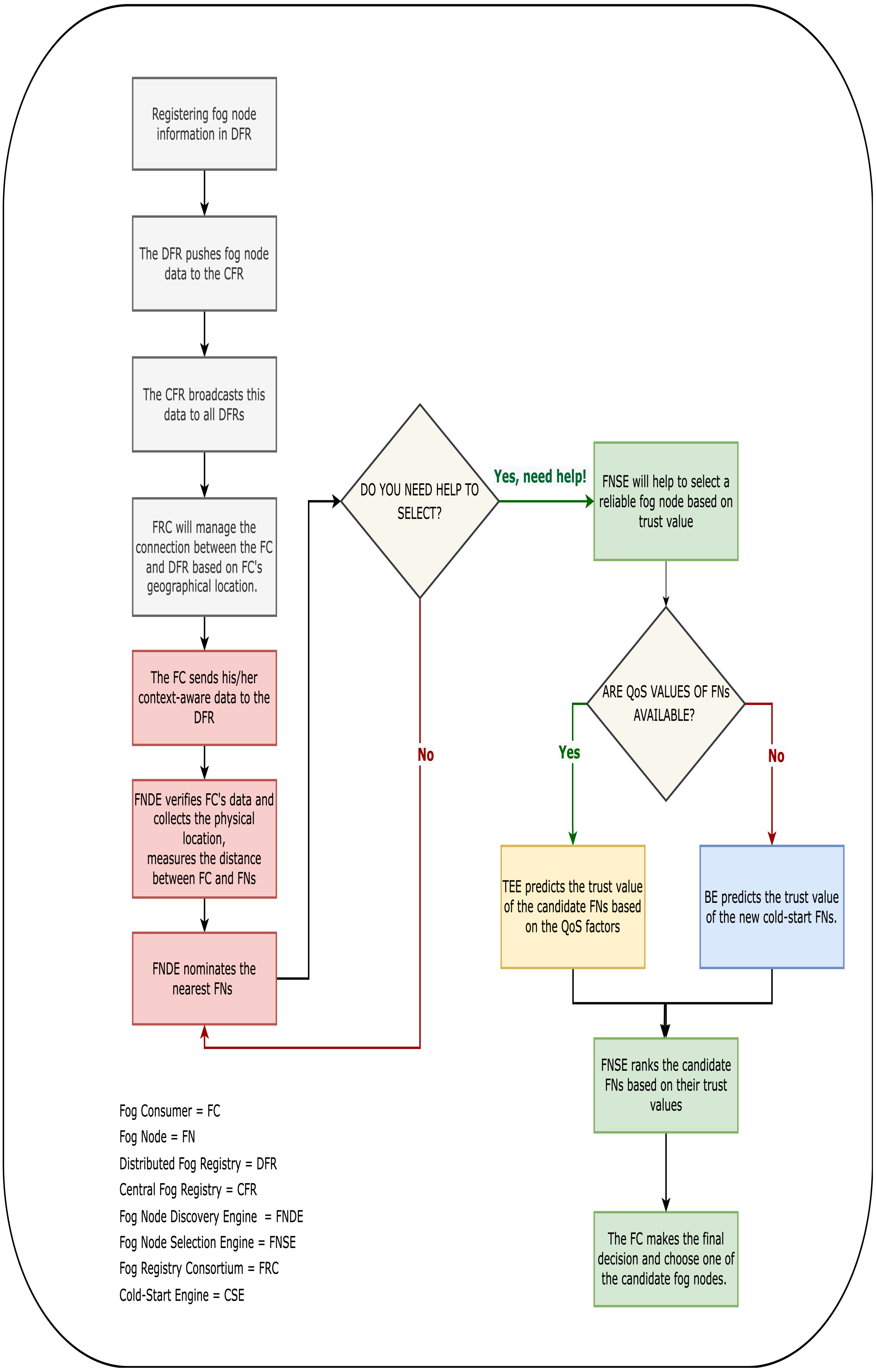

FN7
FN2
FN8
Register (step 1)
FN4
FN5
Register (step 1)
FN3
Distributed Fog Registry N
Fog repository
Fog node discovery engine (step 7,8)
Fog node selection engine (step 9)
Trust Evaluation Engine (step 10)
Bootstrapping Engine (step 11)
Push (step 2)
Broadcast (step 3)
Broadcast (step 3)
Central Fog Registry
Push (step 2)
Broadcast (step 3)
Push (step 2)
Distributed Fog Registry 1
Fog repository
Fog node discovery engine (step 7,8)
Fog node selection engine (step 9)
Trust Evaluation Engine (step 10)
Bootstrapping Engine (step 11)
Distributed Fog Registry 2
Fog repository
Fog node discovery engine (step 7,8)
Fog node selection engine (step 9)
Trust Evaluation Engine (step 10)
Bootstrapping Engine (step 11)
Send request (step 6)
FN1
Result (step 12,13)
Register (step 1)
FC
FOG REGISTRY CONSORTIUM
step(4,5)
FN6
FN = Fog node
FC = Fog Consumer

Figure 4.1 : Conceptual overview of the proposed solution

4.3 Overview of the solution to achieve objective 1

In this section, we overview the solution to achieve objective 1: ***"Develop the architecture of the fog registry consortium framework that can host the***



Registering fog node information in DFR
The DFR pushes fog node data to the CFR
The CFR broadcasts this data to all DFRs
FRC will manage the connection between the FC and DFR based on FC's geographical location.
The FC sends his/her context-aware data to the DFR
FNDE verifies FC's data and collects the physical location, measures the distance between FC and FNs
FNDE nominates the nearest FNs
DO YOU NEED HELP TO SELECT?
Yes, need help!
No
FNSE will help to select a reliable fog node based on trust value
ARE QoS VALUES OF FNs AVAILABLE?
Yes
No
TEE predicts the trust value of the candidate FNs based on the QoS factors
BE predicts the trust value of the new cold-start FNs.
FNSE ranks the candidate FNs based on their trust values
The FC makes the final decision and choose one of the candidate fog nodes.
Fog Consumer = FC
Fog Node = FN
Distributed Fog Registry = DFR
Central Fog Registry = CFR
Fog Node Discovery Engine = FNDE
Fog Node Selection Engine = FNSE
Fog Registry Consortium = FRC
Cold-Start Engine = CSE

Figure 4.2 : The workflow of the solution steps

important information of fog nodes. Furthermore, develop distributed mechanisms to ensure concurrency between the distributed fog registries within the consortium.” In this section, we present an overview of the architecture and working of the FRC that hosts important information on fog nodes and the distributed mechanism to ensure concurrency between the DFRs. In this research, we propose the notion of FRC which is a finite collection of fog registries (FR) that maintains information on the fog nodes. Conceptually, FRC may be defined as follows: FRC = {FR1, FR2, FR3, FR4, . . . FRn}. The FRC is responsible for managing the membership of the FRC, ensuring the integrity of the FRC process, maintaining concurrency in the FRC, distributing the DFRs, and controlling the connection between the fog consumer and the DFR in their geographical limitation.

Figure 4.3 shows the architecture of the FRC.

The step-by-step working of the FRC framework is as follows: (further information in Section 5.5)

Step 1: Register the context-aware fog node data on the DFR when the fog node joins the network (further information in Section 5.5.1).

Step 2: Synchronize the fog node data between the DFRs in the FRC to ensure the concurrency of the information between them (further information in Section 5.5.2).

Step 3: Connecting the fog consumer with the fog registry in FRC (further information in Section 5.5.3).

4.4 Overview of the solution to achieve objective 2

In this section, we present an overview of the solution to achieve objective 2: ***”Develop an intelligent approach to discover fog nodes in a context-aware manner”***. To solve the issue of FND, we propose the FNDE within the

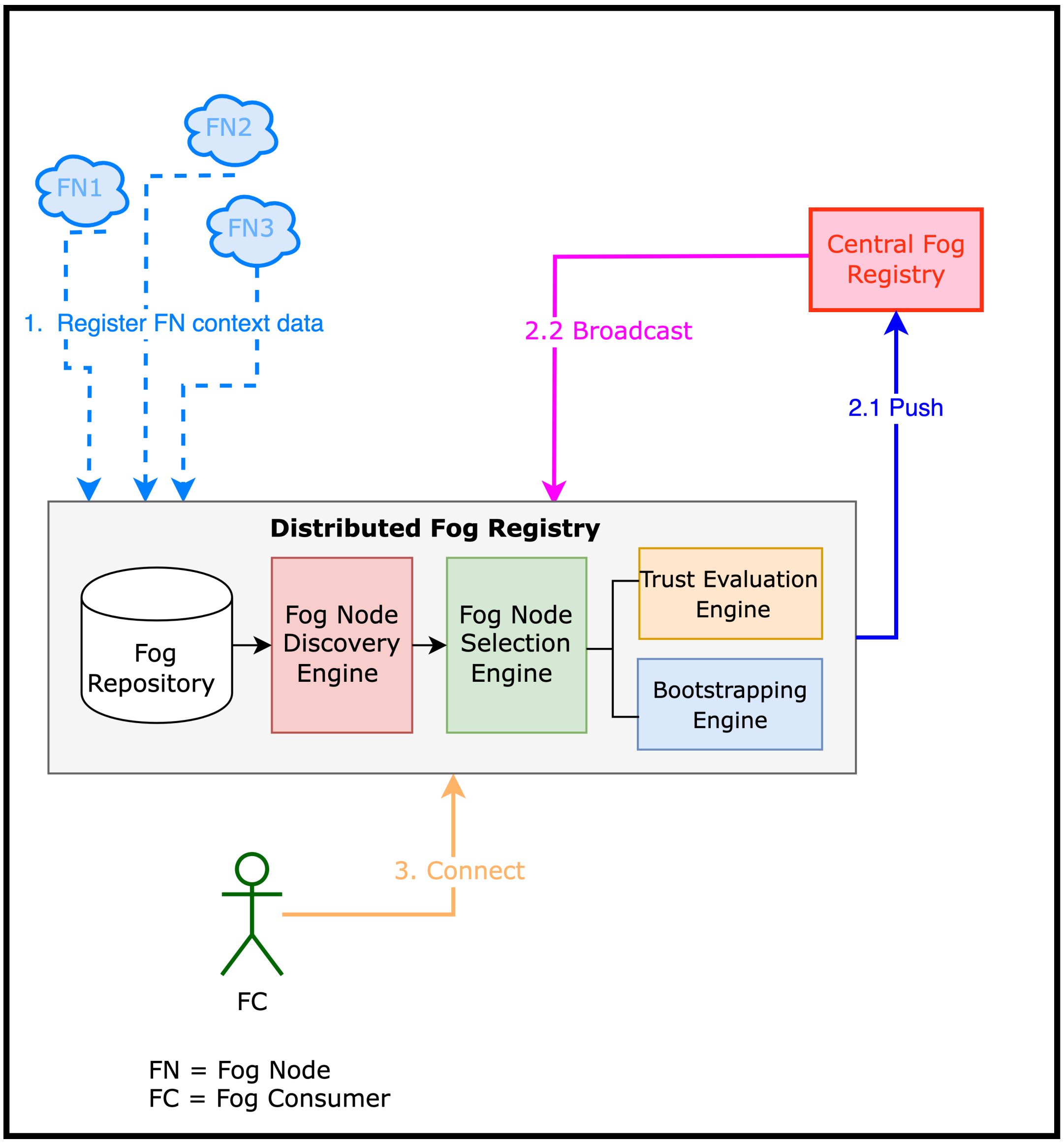


Figure 4.3 : Architecture of the FRC

DFR as an intelligent and distributed FND mechanism which enables a fog consumer to intelligently discover fog nodes in a context-aware manner. The working of the mechanism is encapsulated in the FNDE in the DFR as explained in Chapter 6.

The methodological stepwise working of the FNDE is as follows: (further explanation is given in Chapter 6).

Step 1: Check the identity authentication of the fog consumer (further information in Section 6.2.1).

Step 2: Collect the context-aware data of the fog consumer (further information in Section 6.2.2).

Step 3: Discover the nearest fog nodes based on context-aware parameters by implementing the nearest neighbor algorithms (KNN, Kd-tree, or brute force) (further information in Section 6.2.3).

Step 4: Provide a list of the nearest fog nodes (further information in Section 6.2.4).

Figure 4.4 illustrates the working of FNDE.

4.5 Overview of the solution to achieve objective 3

In this section, we present an overview of the solution to achieve objective 3: ***"Develop an intelligent approach to predict the trust value of fog nodes in order to help the fog consumer select a reliable fog node based on the trust value".*** To solve the issue of FNS, we propose an intelligent trust-based mechanism to assist fog consumer select a reliable fog node from the discovered fog nodes. To do this, we propose the FNSE in the DFR. The FNSE is an intelligent and reliable FNS mechanism in a distributed fog environment. FNSE helps fog consumers choose the most reliable fog node based on the trust value. The FNSE

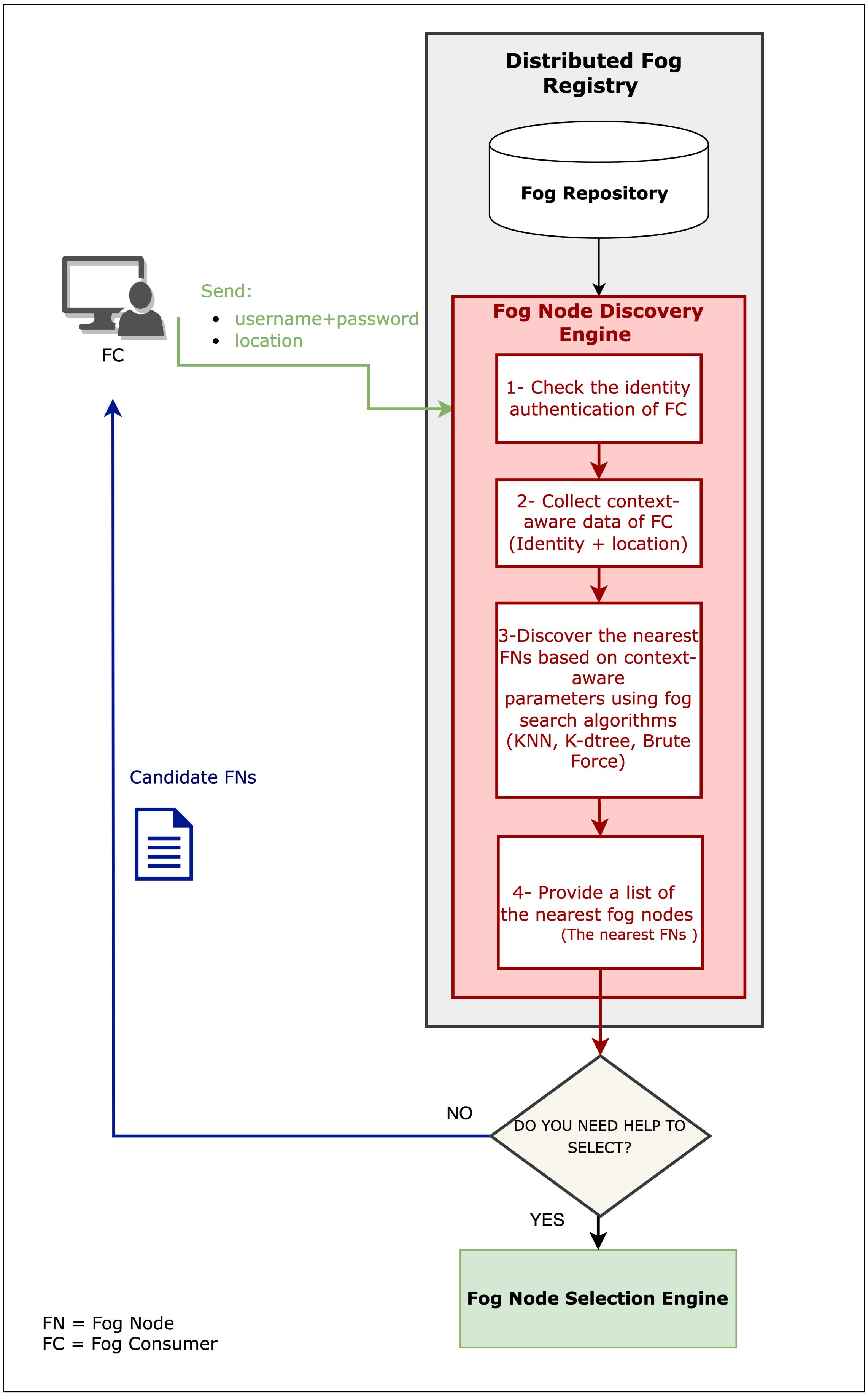

Distributed Fog Registry
Fog Repository
Fog Node Discovery Engine
1- Check the identity authentication of FC
2- Collect context-aware data of FC (Identity + location)
3-Discover the nearest FNs based on context-aware parameters using fog search algorithms (KNN, K-dtree, Brute Force)
4- Provide a list of the nearest fog nodes (The nearest FNs)
FC
Send:
• username+password
• location
Candidate FNs
DO YOU NEED HELP TO SELECT?
NO
YES
Fog Node Selection Engine
FN = Fog Node
FC = Fog Consumer

Figure 4.4 : Overview of the FNDE workflow

is an important component within the concept of the FRC as explained in detail in Chapter 5. The FNSE includes TEE and BE. The TEE performs a trust-driven evaluation of fog nodes and makes recommendations for selection (more details in Chapter 7). The TEE includes an intelligent mechanism for predicting the trust values of fog nodes to make a reliable selection decision. However, the BE performs a trust-driven evaluation of new fog nodes and makes recommendations for selection (more details in Chapter 8).

The methodological stepwise working of the FNSE framework is as follows: (further explanation is given in Chapter 7)

Step 1: The FNSE checks the historical information or any previous interaction history of the QoS factors of the fog node.

Step 2: If the fog node has the historical information of the QoS factors provided to this fog node, then FNSE will go to Steps 3 and 4. If a fog node is new and has only recently joined the network, it does not have any previous interaction history and suffers from the cold-start problem, in which case the FNSE will move to Step 5.

Step 3: Collect QoS data for each fog node with which they have previously interacted. The QoS factors are response time, availability, throughput, successability, reliability, and rank for service quality. The QoS data are stored in the fog repository in the DFR along with context date of fog node. When the QoS data are added to the repository, the FRC should be synchronized and updated as described in Section 5.5. The DFR uses the push synchronization mechanism to send the data to the CFR. Then, the CFR broadcasts this update to all DFRs.

Step 4: The TEE in FNSE intelligently predicts the trust value of the fog node. The TEE predicts the trust value of each fog node based on the value of the QoS factors. The fuzzy logic system, logistic regression and deep neural network (DNN) are used to predict the trust value of the fog nodes (further information is given in Section 7.2).

Step 5: If a fog node is new and has only recently joined the network, it does not have any previous interaction history and suffers from the cold-start problem, in which case the BE in FNSE proceeds to predict the trust value of the new fog node. The fuzzy logic system, logistic regression and DNN are used to bootstrap the reputation of the new fog node and predict the trust value of new cold-start fog node. (Further information is given in Section 8.2).

Step 6: After assigning the trust value to each fog node, the FNSE ranks the fog nodes in ascending order based on their trust values.

Step 7: The final ranked result is displayed to the fog consumers to assist them select one of the candidate fog nodes. This allows the fog consumer to make an informed decision in selecting the most reliable fog node for their requirements.

Figure 4.5 describes the working of the FNSE framework.

4.6 Overview of the solution to achieve objective 4

In this section, we present an overview of the solution to achieve objective 4: ***"Develop an intelligent approach to bootstrap new fog nodes into a fog ecosystem".*** In this section, we present the solution for the cold-start problem of fog nodes. We propose an intelligent approach to bootstrap a new fog node provider into the fog environment or fog ecosystem. Our selection approach, the FNSE, is based on the trust value of the fog node based on the QoS factors. If there are no QoS values, this means the FNSE is unable to make a meaningful selection of fog nodes. When a fog node lacks previous QoS data, has recently joined the network and it

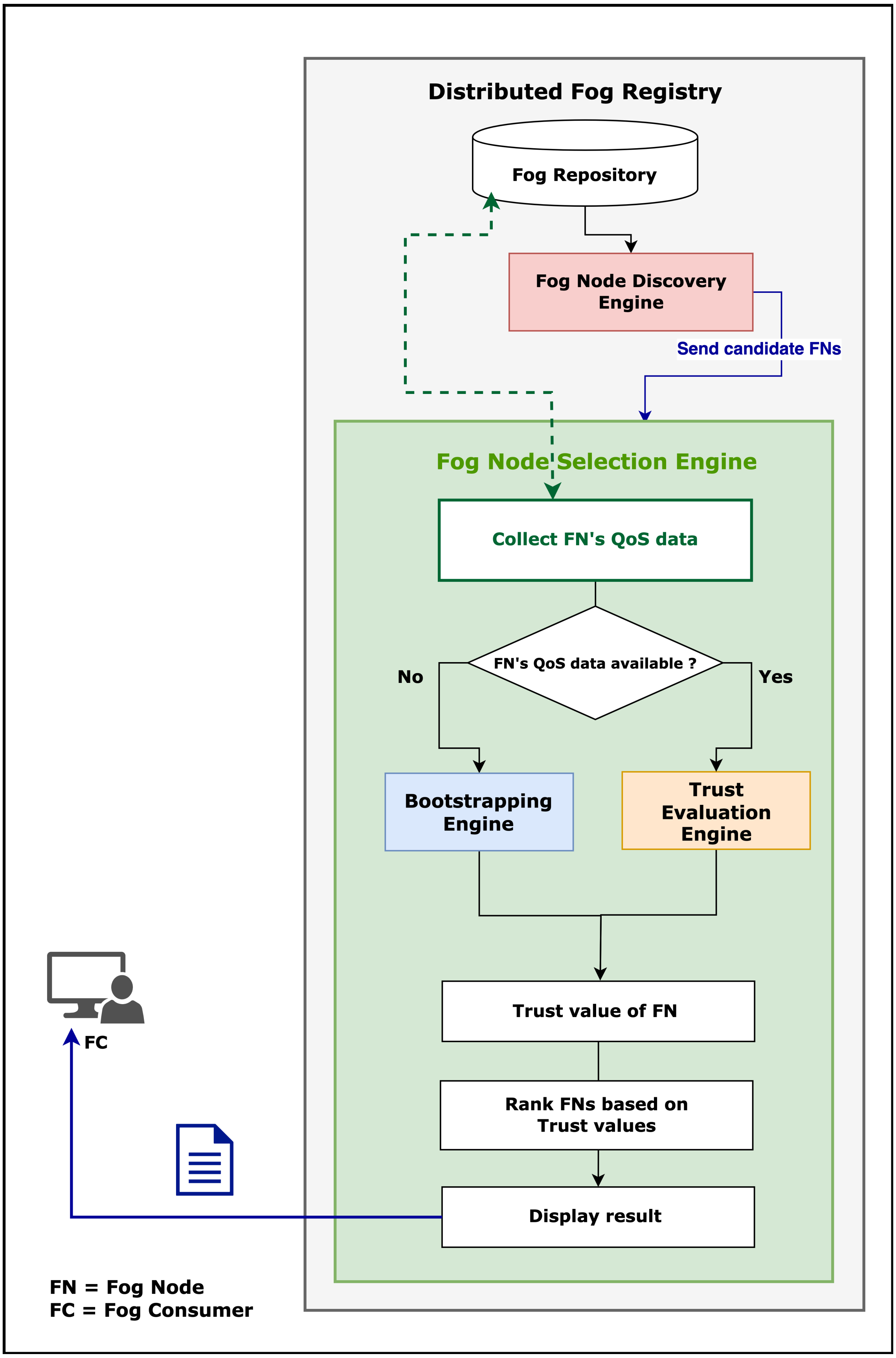


Figure 4.5 : Overview of the working of the FNSE framework

does not have any previous interaction history, the BE takes charge of performing a trust-driven assessment of the new fog nodes and recommending a suitable and reliable fog node for selection. The BE specifically addresses the cold-start phase that new fog nodes experience when joining the network.

The BE is an important component of FNSE in the DFR. The BE comprises two modules, the *QoS prediction module* to intelligently predict the QoS value of a new fog node, and the *reputation prediction module* to predict the trust value and make a successful and reliable selection, as explained in Chapter 8.

The methodological stepwise working of the BE in the FNSE is as follows: (further explanation in Section 8.2)

Step 1: **QoS prediction for new fog nodes by the QoS prediction module:** The QoS prediction module in BE is responsible for predicting the QoS value of the new cold-start fog node which includes:

(a) **Clustering:** Cluster fog nodes based on contextual attributes such as geographical information using K-means algorithm.

(b) **Closest Cluster:** When a new fog node joins the network, the QoS prediction module determines the closest cluster based on its geographical location. The K-nearest neighbors (KNN) algorithm is used to measure similarity and identify similar fog nodes (nearest neighbors).

(c) **QoS Prediction:** Predict (initialize) the QoS value to the new fog node based on the QoS data of the closest fog nodes (neighbors) by calculating the average QoS values of the 10 nearest neighbors.

Step 2: **Trust value prediction using reputation prediction module:** The reputation prediction module intelligently predicts the trust value of the new fog node based on its predicted QoS values. We applied various AI-based reputation prediction techniques such as fuzzy logic, logistic regression, and DNN.

All the steps involved in Step 1 will be carried out offline when the new fog node comes to the area. So, when the fog consumer wants to know the trust value of the new fog node, the BE will measure the similarity between the new fog node and all the nearest fog nodes in its cluster. Similar nearest fog nodes should provide similar QoS features.

Figure 4.6 illustrates the BE function.

4.7 Overview of the solution to achieve objective 5

In this section, we present an overview of the solution for objective 5: ***"Build a prototype system for the evaluation and validation of the framework and the techniques developed in sub-aims (1) – (4)"***

To validate the proposed framework, a simulation of the fog environment is developed. An existing fog simulator, OMNeT++, is used to set up the fog simulation environment.

4.7.1 The validation process for the solution to achieve objective 2

Step 1: Initialization process: The initialization process comprises the following steps:

(a) Set up the framework in OMNeT++.

(b) Specify the number of fog nodes, starting with 100 nodes. A random latitude and longitude value are assigned for each fog node.

(c) Determine the number of fog registries, starting with 5 registries.

(d) Determine the number of iterations = n times. The value of the parameters for (b) and (c) vary from one iteration to the next.

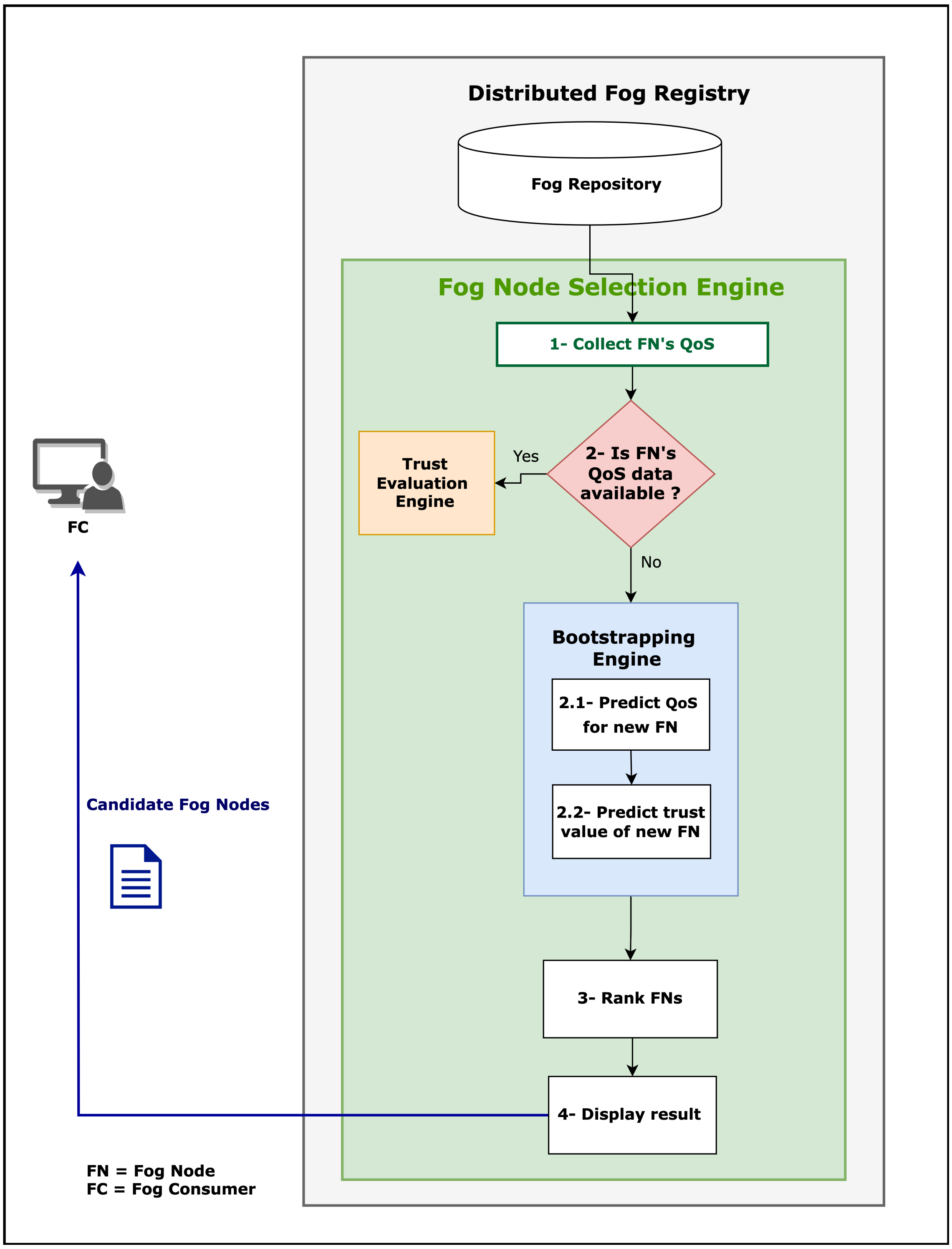


Figure 4.6 : Overview of the working of the FNSE and BE

Step 2: Implement three nearest neighbor algorithms (KNN, K-d tree, brute force) for FND in OMNeT++.

Step 3: The system administrator selects a random fog node (e.g. fog node A as a fog consumer) from the available 500 fog nodes and asks it to carry out context-aware FND. The system administrator knows the closest context-aware fog nodes for fog node A; however, this information is unknown to fog node A.

Step 4: Use well-known and accepted metrices such as precision, recall and F1 score which are defined as:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.2}$$

$$\text{F1 Score} = \frac{2 \cdot (\text{Recall} \cdot \text{Precision})}{\text{Recall} + \text{Precision}} \tag{4.3}$$

Then, we compute and compare the accuracy of the three methods.

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + TN + FP} \tag{4.4}$$

where:

- TP = True Positive: means the discovered fog node is relevant to the actual output.

- TN = True Negative: means the undiscovered fog node is not relevant to the actual output.

- FP = False Positive: means the discovered fog node is irrelevant to the actual output.

- FN = False Negative: means the undiscovered fog node is relevant to the actual output.

Step 5: Repeat steps 3 and 4 n times

4.7.2 The validation process for the solution to achieve objective 3 and objective 4

Step 1: Initialization process: as discussed in Section 4.7.1, this step specifies the number of fog nodes, the fog registries, and number of iterations. Figures 5.1 and 5.2 illustrates the setup of the OMNeT++ simulator, which includes 100 fog nodes, one fog consumer, 5 DFRs and one CFR.

Step 2: Implement the fuzzy logic-based approach, logistic regression-based approach, and DNN-based approach in the TEE in FNSE in the DFR module for objective 3. Implement the BE with fuzzy logic, logistic regression, and DNN in FNSE in the DFR module for objective 4.

Step 3: The system administrator selects random fog nodes X from the available fog nodes and asks TEE to predict the trust value of these fog nodes and asks FNSE to carry out the FNS process which is called predicted results. The system administrator knows the trust value of fog nodes X and the list of fog nodes that should be selected which is called the actual results; however, this information is unknown to nodes X in order to compare the predicted results against the actual results.

Step 4: Use well-known and accepted metrices such as precision, recall and F1 score which are defined in equations 4.1, 4.2, 4.3.

Then, we compute and compare the accuracy as defined in equation 4.4 of the three methods.

where a:

- TP = True Positive: means the predicted output is relevant, and it is correctly identified as relevant.

- TN = True Negative: means the predicted output is not relevant, and it is correctly identified as not relevant.

- FP = False Positive: means the predicted output is not relevant, but it is mistakenly identified as relevant.

- FN = False Negative: means the predicted output is relevant, but it is mistakenly identified as not relevant.

Step 5: Repeat steps 3 and 4 n times

4.8 Conclusion

In conclusion, this chapter has provided a comprehensive overview of the solutions proposed to achieve each objective outlined in Chapter 3. The chapter commenced by discussing the selected research methodology, namely the design science research methodology, which serves as the guiding framework for this research. This methodology emphasizes the practical development and evaluation of innovative solutions. Then, we presented an overview of all the solutions to achieve objectives 1, 2, 3, 4, and 5, detailed in Chapter 3, in Sections 4.3, 4.4, 4.5, 4.6 and 4.7 respectively. The next chapter provides more comprehensive details on the development of the FRC framework's architecture and discusses the FRC components, activities, design, implementation, and experimental evaluation.

Chapter 5

Developing the architecture of the fog registry consortium (FRC)

5.1 Introduction

In this chapter, we explain in detail our solution for developing the architecture of the FRC and its pivotal role. The FRC serves as the backbone of the proposed framework, responsible for managing crucial information regarding fog nodes and ensuring seamless concurrency between the DFRs within the consortium. In our proposed framework, we propose the novel concept of the FRC as defined in Section 3.2.10. In this chapter, we present the significance of the FRC in Section 5.2. Then, we explain the architecture of FRC in Section 5.3. We elaborate in detail the activities of FRC in Section 5.4. The working steps of the FRC are explained in Section 5.5. Section 5.6 presents the implementation and prototype setup of the FRC. Finally, Section 5.7 concludes the chapter.

It is worth noting that portions of this chapter have been previously published in the Internet of Things journal [110], further validating the relevance and novelty of our research.

5.2 The significance of FRC

The FRC plays a critical role in the efficient operation of the fog ecosystem. It manages and stores information about fog nodes within the network. This includes essential details such as fog node ID, fog node name, fog node description, fog node location, and QoS parameters. By maintaining this information in a structured manner, the consortium enables efficient management and coordination of fog resources. The FRC facilitates the dynamic membership of fog registries. In a fog environment, the membership of the consortium is transient and subject to frequent changes as fog nodes join or leave the network. This dynamic nature allows for flexible scalability and adaptability, ensuring that the consortium can accommodate the evolving fog computing landscape. The FRC ensures that the integrity of the DFR process is of the utmost importance. The consortium establishes mechanisms and checks to maintain the accuracy and consistency of the stored information. This ensures that fog nodes' attributes are up-to-date and reliable, enabling effective decision-making processes within the fog computing infrastructure. In a DFR environment, where multiple registries exist across different locations, the consortium ensures the synchronization of data between these registries. This synchronization mechanism helps in maintaining consistent and coherent information across the DFR, enabling seamless communication and collaboration among fog registries. The consortium also plays a role in controlling the connection between fog consumers and the DFR based on their geographical limitations. This ensures that fog consumers are connected to the most relevant and suitable DFR within their proximity, promoting efficient resource utilization and QoS optimization. The following section elaborates the main components of the FRC architecture and their respective functionalities within the fog computing environment.

5.3 The architecture of FRC

In Chapter 4, we defined the FRC as an intermediary between the fog consumer and fog nodes that acts as a broker for various activities such as fog discovery and fog selection. FRC is defined as a finite collection of fog registries (FR) that maintains information on the fog nodes. Conceptually, as mentioned in Section 3.2.10, FRC is defined as follows: FRC = {FR1, FR2, FR3, FR4, . . . FRn}. In this section, we present the structural design of our proposed framework, the FRC, which consists of two types of fog registries as follows:

1. **The main central fog registry (CFR):** The CFR controls and coordinates all the DFRs in the consortium, ensures an equally and uniformly spread of DFRs, manages the membership of the DFR in the FRC, ensures the integrity of the FRC, and ensures concurrency by synchronizing fog node data between all other DFRs to keep them up to date.

2. **The distributed fog registries (DFR):** In contrast to the main CFR, there are multiple geographically dispersed registries that synchronize with the main CFR. The DFRs are located in different remote regions and are uniformly spread across the globe. Each DFR stores information on all fog nodes in the network. All DFRs have the same data on all fog nodes and are synchronized. The DFRs are the first point of contact for fog consumers. The DFR further has the following modules:

   (a) **Fog repository:** Fog repository stores essential information about all the fog nodes in the fog ecosystem. Within the fog repository, we store specific details related to each fog node which include fog node ID which is a unique identifier assigned to each fog node, fog node name which is a specific name assigned to the fog node, fog node description which is detailed information about the fog service functionalities and operations, fog node location which is the physical location which contains the longitude and latitude where the fog node is located, and the QoS of the fog node which is the QoS values associated with the fog node.

   (b) **Fog Node Discovery Engine (FNDE):** The FNDE, as discussed in Chapter 6, is a component within the DFR that is responsible for identifying and locating fog nodes within the fog ecosystem intelligently and in a context-aware manner. It employs several techniques to perform the discovery process efficiently.

(c) **Fog Node Selection Engine (FNSE):** The FNSE, elaborated in Chapter 7, is a third component in the DFR. It facilitates the intelligent selection and assignment of trusted fog nodes to handle specific computational tasks or services. The FNSE consists of two key sub-components:

   i. **Trust Evaluation Engine (TEE):** The TEE, as described in Chapter 7, is the first component within the FNSE. The TEE is a trust-driven evaluation of fog nodes and makes recommendations for selection. It is responsible for predicting the trustworthiness and reliability of fog nodes in the fog ecosystem. The TEE predicts the trust value for each fog node intelligently based on the QoS of the fog node. These trust values play a crucial role in the decision-making process of selecting reliable fog nodes for other computational tasks.

   ii. **Bootstrapping engine (BE):** The BE, explained in Chapter 8, is another component within the FNSE. It addresses the scenario where a fog node is newly added to the fog ecosystem or restarted after a period of inactivity. It is responsible for intelligently predicting the trust value of the new cold-start fog node that recently joined the fog ecosystem and suffers from an unknown QoS. This is a fundamental role in trust-based fog node selection.

Figure 4.3 depicts the architecture of the FRC.

5.4 The activities of FRC

In this section, we highlight the significant responsibilities that lie within the realm of the FRC. The FRC is entrusted with the following key activities:

1. Managing the membership of the FRC (as discussed in Section 5.4.1): The FRC oversees the dynamic composition of its membership, ensuring that fog registries are appropriately included or excluded from the consortium.

2. Ensuring the integrity of the FRC process (as detailed in Section 5.4.2): The FRC implements robust measures to guarantee the trustworthiness and reliability of its operations, thereby upholding the integrity of the entire process.

3. Ensuring concurrency in the FRC by synchronizing data between DFRs (as elucidated in Section 5.4.3): The FRC facilitates the seamless coordination and synchronization of data across DFRs, enabling consistent and coherent information exchange.

4. Distributing the DFRs across the globe in a uniform manner: The FRC strategically allocates the DFRs to various locations worldwide, ensuring equitable distribution and optimizing network performance.

5. Controlling the connection between fog consumers and the DFR within their geographical limitation: The FRC governs the connectivity between fog consumers and the DFR, effectively managing access based on geographical constraints and limitations.

5.4.1 Membership of the FRC

The membership of the FRs within the FRC could either be static or dynamic. In a true fog environment, the membership of the FRC is transient and dynamic. However, the membership of the FRC could also be static with proper checks and balances in place to ensure the integrity of the process activity carried out by the FRC. In our research, we use a static membership function for the FRC. Below we describe how fog nodes can become members of the FRC and also how the integrity of the activity carried out by the FRC can be maintained (particularly under static membership). The job of the FRC is to store an exhaustive list of all the fog nodes in the fog network with their bespoke attributes such as but not limited to fog node ID, fog node name, fog node location etc. It is critical to note that this list of fog nodes stored in the FRC is not static but an evolving list where new fog nodes may join the fog network and existing ones drop out of the fog network. The membership criteria for the FRC is grounded on the factors that enable the FR to maintain and update the list of the fog nodes in a dynamic manner. As such, we use the following membership functions for joining our FRC:

1. Storage capacity: The FR should have very large storage space to maintain the extensive and expansive list of fog nodes with their bespoke attributes (ideally SSD storage to minimize retrieval time).

2. Processing capacity: The FR should have fast processing capacity (ideally GPUs; however non-GPU processors such as the i7 processor are sufficient as well). This is again to minimize retrieval times.

3. Reputation value: The FR should be a trustworthy fog node as evidenced by its reputation value.

In this research, we propose the above three factors for selecting the FRs, however in practical implementation, the above factors can be different. The CFR initiates the selection process. It is critical to note that while the CFR initiates the process, the activities within the selection process are carried out in a manner that is visible to all the participants.

5.4.2 Ensuring the integrity of the FRC process

To ensure that no single entity or group of entities within the FRC has control of the entire activities in the FRC to the extent that they are able to manipulate its activities in an unfair manner, we propose the notion of checkpointing and benchmarking. Checkpointing is carried out at regular periodically recurring intervals of time (n). The working of the checkpointing and benchmarking is as follows:

1. During the checkpointing process, each member of the FRC is asked to submit a one-way hash of its repository, using a dynamically generated key. The CFR (explained below) generates this key dynamically and every member of the FRC is required to submit a one-way hash outcome to the FRC which is shared with all members of the consortium.

2. During the benchmarking process, led by the CFR, the hash outcomes of all the 'N' DFRs are compared to determine if there are any erroneous DFRs. These erroneous DFRs, whose hash outcomes do not match those of the majority ones, are asked to rectify the content of their respective registries so that there is no discrepancy in the one-way hash outcomes. If the CFR finds a DFR whose one-way hash outcome repeatedly does not match those of the majority, then that fog node may be expelled from the FRC.

It is critical to note that both the checkpointing and the benchmarking process is carried out in a manner that is visible to all members of the consortium. A consortium-visible blackboard is used to systematically display all the activities of the checkpointing process (such as a call for one-way hash functions and the submission of one-way hash functions) and also all the activities of the benchmarking process (outcome of the comparison process, identification of non-compliant one-way hash functions, rectification of the repositories of non-compliant nodes and expulsion of repeating non-complaint nodes).

5.4.3 Ensuring concurrency between DFRs in the FRC

Each DFR stores information about all fog nodes in the fog network. The FRC achieves consistency and integrity of information by ensuring that the information is stored and synchronized in all the DFRs. In this research, we use the push synchronization mechanism to synchronize fog node data between the DFRs in the FRC. It is an event-based process and is carried out when new information is added or when information is updated. The redundant data in the DFRs helps detect any malicious node if data is changed. Also, if one of the DFRs fails or dies, the data can be recovered from another registry. In this research, we propose the notion of DFRs as an enabler to fog node discovery and fog node selection. The notion of the DFR also gives rise to other research issues, such as consortium formation, consortium evolution etc. Developing intelligent methods for the formation of the fog consortium and fog dissolution are very important research issues. However, they are outside the scope of this research.

5.5 The working steps of FRC

In this section, we present the step-by-step working of the FRC framework.

5.5.1 Step 1: Register the context-aware fog node data on the DFR when the fog node joins the fog network

When a fog node joins the fog network, the context-aware data of the fog node is published on the DFR closest to its geographical location. The context fog node data includes the identity and the current physical location of the fog node (fog node name, fog node ID, fog node location). These data are stored in the fog repository of the DFR which means data are registered in one of the DFRs. The location of the fog node in this research is static and must be determined in its geographical range. Fog nodes may be located at various locations such as shopping malls, schools, hospitals, libraries, train stations, etc.

5.5.2 Step 2: Synchronize the fog node data between the DFRs in the FRC to ensure the concurrency of the information between them

When the data of the fog node is registered, whether data is new or modified in one of the DFRs, the DFR will push this new data or changed data to the CFR using the push synchronization mechanism. The data source, which is the DFR, notifies the data sink which is the CFR of the new or changed data. The push is an event-based process which starts when the new data arrive or data are changed. This event-based push mechanism ensures data consistency between all DFRs. Then, the CFR broadcasts the new data or the changed data to all the other DFRs.

5.5.3 Step 3: Connecting the fog consumer with the fog registry in FRC

When the fog consumer wants to discover a fog node, they need to connect to the one of distributed fog registries. They send their credentials to the FRC which will find the nearest DFR in the fog consumer's geographical boundary and responds. Automatically, current physical location information such as the latitude and longitude values of the fog consumer is collected by the DFR to discover the closest fog nodes.

Figure 4.3 presents the architecture and workflow of the FRC framework.

5.6 FRC implementation and prototype setup

The FRC framework is implemented in a simulation environment using the OMNeT++ platform [111]. OMNeT++ is an open-source simulation tool programmed in the C++ language [111]. For our implementation, we utilized OMNeT++ version 5.6.2. OMNeT++ is selected because it is one of the network simulators that supports our proposed model features such as location-awareness, low latency and scalability. Moreover, OMNeT++ has been widely used in academia and its flexibility enables it to identify the network behaviour based on our scenario. It is capable of handling fog networks and devices in a distributed environment. OMNeT++ consists of three key components: the network, the NED language, and the configuration file. In the network component, we define four modules: CFR, DFR, FN, and FC, as illustrated in Figure 5.1. Each module represents a distinct entity within the FRC architecture and interacts with other modules to simulate the behaviour of the fog computing system. The NED language is employed to create the network topology, specifying the connections and interactions between the different modules. Using the NED language, we define the relationships and communication protocols among the CFR, DFR, FN, and FC modules, allowing for the simulation of their interactions within the FRC environment. To run the network simulation, we utilized the configuration file named omnetpp.ini. This file contains the necessary parameters and settings to configure the simulation environment in OMNeT++. It specifies the simulation duration, network parameters, module configurations, and other simulation-specific details required for executing the FRC simulation.

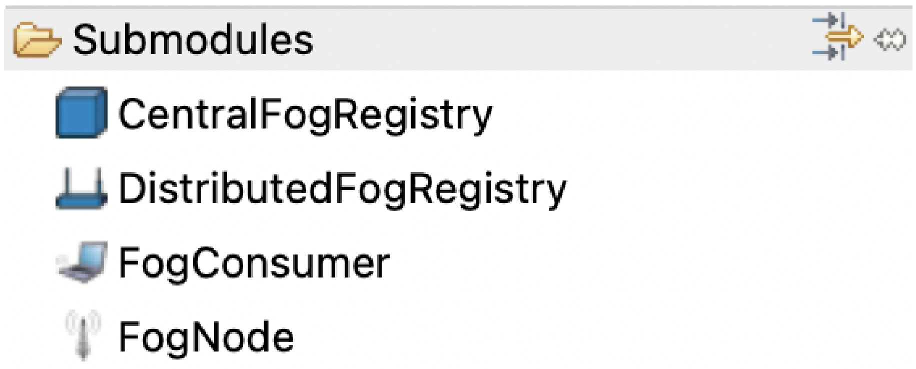


Figure 5.1 : Simulation modules

Firstly, we established a wireless network and created four modules to simulate the FRC environment. Figure 5.2 depicts the simulated network comprising these four modules. The first two modules are the CFR and the DFR, both of which contain the fog repository. The fog repository stores comprehensive information about all registered fog nodes, which is pushed from the DFR. Table 5.1 provides a detailed schema of the fog repository used in both the CFR and DFR.

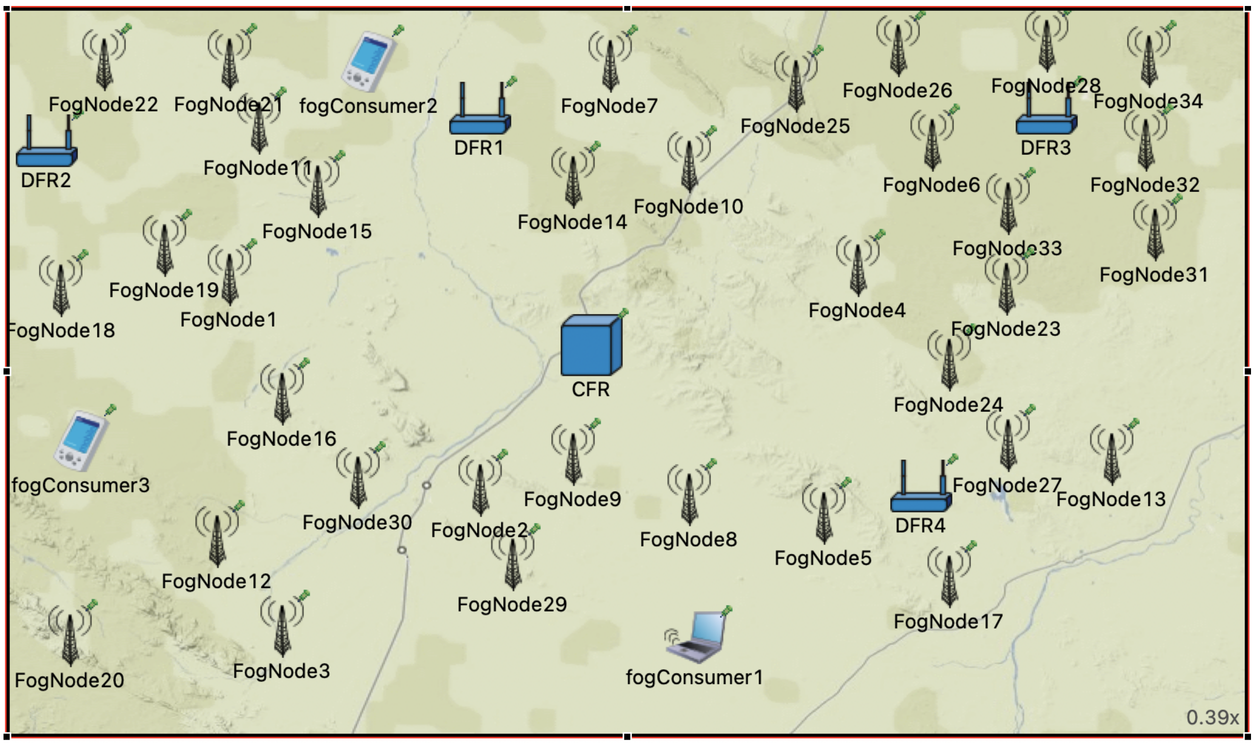


Figure 5.2 : Simulation of the FRC framework

The third module represents the fog nodes within the network. Each fog node is characterized by four parameters: identification, name, and physical location (latitude and longitude). The network is designed to accommodate up to 2000 fog nodes, each associated with its specific physical location. For simulation purposes, we utilized the latitude and longitude coordinates of edge servers from the EUA dataset [112]. Table 5.2 presents a snapshot of the fog node dataset used in the simulation.

The fourth module represents the fog consumers within the network. Each fog consumer is represented by the fog consumer ID, fog consumer IP, and physical location (latitude and longitude). To simulate fog consumers, we employed the end

Table 5.1 : The fog repository schema of CFR and DFR

FN ID	**FN NAME**	**LATITUDE**	**LONGITUDE**
1000	Fort Hill Wharf DARWIN	-12.471947	130.845073
10000	Cnr Castlereagh and Lethbri PENRITH	-33.756158	150.698182
10000002	Optus 50m Lattice Tower 71 Eastward Road Utakarra	-28.77766	114.63426

user dataset from the EUA dataset [112]. Table 5.3 provides a glimpse of the fog consumer dataset used in the simulation. Communication between these modules is facilitated through channels, allowing for message exchange and interaction.

Secondly, within the simulation environment, we implemented FNDE and FNSE, which lastly comprises TEE and BE. These engines are incorporated into the DFR. For further details about the FNDE, FNSE, TEE, and BE, refer to Chapter 6, 7, and 8, respectively.

By simulating these modules and engines, we can analyze the behavior and performance of the FRC in a controlled environment. This simulation allows us to evaluate the effectiveness and efficiency of the FNDE, FNSE (including TEE and BE), and their impact on the overall fog computing system.

In this research, we utilized a comprehensive dataset obtained from EUA (Edge

Table 5.2 : A snapshot of the fog node dataset

SITE_ID	NAME	LATITUDE	LONGITUDE
1000	Fort Hill Wharf DARWIN	-12.471947	130.845073
10000	Cnr Castlereagh and Lethbri PENRITH	-33.756158	150.698182
10000002	Optus 50m Lattice Tower 71 Eastward Road Utakarra	-28.77766	114.63426
10000003	6 Knuckey Street Darwin	-12.464597	130.840708
10000004	Cape Wickham Links Clubhouse KING ISLAND	-39.5964	143.9339

Table 5.3 : A snapshot of the fog consumer dataset

FC_ID	IP	LATITUDE	LONGITUDE
1	1.120.2.1	-37.8833	145.3333
2	1.120.0.1	-30.5083	151.6712
3	1.120.163.1	-21.0405	149.1849
4	1.122.32.1	-31.9344	115.8716
5	1.123.11.1	-34.8333	138.6333

User Allocation) datasets [112] which comprises real-world data collected from publicly available sources of edge computing. The dataset specifically includes the geographical locations of both edge servers and end-users within the Australian region. Due to the lack of fog node datasets and because there is no specific publicly available standardized dataset exclusively focused on fog services, we used the EUA Edge Computing Dataset, since fog computing and edge computing share similarities in terms of processing data at the network edge.

The EUA dataset contains precise geographical Information regarding all cellular base stations in Australia, which are considered as edge servers. In total, the edge server dataset comprises 95,562 entries, each representing a fog node within the fog computing network in our research. The attributes associated with fog nodes include SiteID, Name, Latitude, and Longitude, as outlined in Table 5.2. Moreover, the end user dataset from the EUA datasets consists of 4,748 entries, representing end-users who are considered to be fog consumers in our research. For fog consumers (end users), we extracted the necessary attributes, namely fog consumer ID, IP address, Latitude, and Longitude, as specified in Table 5.3. All the data pertaining to fog nodes and fog consumers were stored within the fog repository in the DFRs.

To validate the effectiveness and functionality of the FRC, we conducted experiments that involved the FND process explained in Chapter 6 and the FNS process explained in Chapters 7 and 8. These experiments aim to assess the FRC's ability to accurately discover appropriate fog nodes and select reliable ones based on predefined criteria.

Furthermore, we performed a series of experiments on the simulated network, considering various scenarios with different numbers of fog nodes and fog consumers. These experiments were conducted to gain insights into the behavior, performance, and scalability of the FRC with the CFR, and DFRs under different conditions.

For more detailed information on the working of the FND process and experiments, refer to Chapter 6. Furthermore, for more information on the working of the FNS process and experiments, refer to Chapter 7 and 8. Both FND and FNS are based on FRC. By conducting these experimental evaluations, we are able to assess and validate the functionality, efficiency, and effectiveness of the FRC in managing fog nodes and facilitating communication between fog consumers and fog nodes within the simulated network.

5.7 Conclusion

In conclusion, Chapter 5 effectively addresses the first research question by proposing the development of the FRC. The chapter begins by introducing the novel concept of the FRC, establishing its significance within the context of fog computing. By defining the FRC and its key components, a comprehensive understanding of its purpose and functionality is provided. The chapter further delves into the architecture and working steps of the FRC, presenting a detailed overview of how the consortium operates. This includes the intricate processes involved in managing and synchronizing data across DFRs, ensuring the integrity and consistency of information. The implementation of the FRC framework is explained, highlighting the use of the OMNeT++ platform for simulation purposes.

Chapter 6

Developing an intelligent approach to discover fog nodes in a context-aware manner

6.1 Introduction

The fog ecosystem is a dynamic and distributed environment where fog nodes can join or leave the network. Locating the best and most appropriate fog node is a primary concern of fog consumers. The FND is an important process in the choice stage of the fog node environment where the fog consumers can find the optimal fog nodes that meet their requirements. The FND process can be achieved by several techniques such as context-aware, location-aware, content-aware mechanism etc. to improve the accuracy and efficiency of the discovery process. In this research, the FND process aims to identify the best fog nodes based on their context awareness. As discussed in Chapter 3, context awareness refers to the location-based context awareness that involves using geographic location information to determine the proximity of fog nodes to the fog consumer. To ensure that the tasks from the fog consumer are processed in a timely manner, one of the crucial aspects to consider for FND is the geographic distance between the fog node and the fog consumer as this directly impacts latency, response time, and bandwidth usage for the fog consumers. Thus, location-based context awareness is one of the key decision criteria for FND to ensure that the QoS metrics are satisfied. So, discovering the nearest fog nodes based on location-based context awareness will reduce latency and improve network performance. In this chapter, we explain in detail our solution for developing an intelligent approach to discover fog nodes in a location-based context-aware manner.

In this chapter, we propose the FNDE within the DFR as an intelligent and distributed fog discovery mechanism which enables a fog consumer to intelligently discover fog nodes in a context-aware manner. The FNDE is a framework responsible for searching and discovering relevant fog nodes in a context-aware manner. In Section 6.2, we discuss the framework of the proposed solution FNDE, explain the FNDE working steps, and provide the algorithm of the FNDE. Section 6.3 presents the implementation of the FNDE framework. Furthermore, the extensive experiments are detailed in Section 6.4. Section 6.5 discusses the evaluation process and results. Finally, Section 6.6 concludes the chapter.

It is worth noting that the content of this chapter has been previously published in the Internet of Things journal [110], further validating the relevance and novelty of our research.

6.2 FNDE framework

We proposed the FNDE as a mechanism for the process of discovering fog nodes in a fog ecosystem. The development of FNDE is encapsulated in the DFR. Basically, after publishing and synchronizing the context-aware fog nodes' data in CFR and DFRs (Chapter 5), the process of discovery starts when the fog consumer intends to consume a service in the fog node. On successful authentication, the FNDE collects the context-aware data of the fog consumer to carry out the discovery process. In the scope of this research, the term context-aware refers to the identity and location of the fog consumer. The context-aware data of a fog consumer consists of fog consumer ID, fog consumer IP, fog consumer latitude, and fog consumer longitude. The FNDE applies the intelligent search algorithm based on the context-aware data of the fog consumer and fog nodes to find the fog nodes nearest to the current physical location of the fog consumer. The FNDE provides a list of available nearest fog nodes in order to select one of them. The following steps demonstrate the methodological working steps of the FNSE framework:

6.2.1 Step 1: Check the identity authentication of the fog consumer

When the fog consumer wants to connect to the fog node, the fog consumer will be prompted on their edge device (such as a mobile device) about the availability of the nearest fog node. The discovery of the fog node is based on the location-based context information of the fog consumer and also the fog node. The fog consumer initiates the FND process by providing their login credentials (usually comprising username and password; however, other authentication mechanisms such as finger printing may also be used). Developing reliable and foolproof authentication measures has been a longstanding research question. In this research, we do not intend to address this question and instead focus on developing reliable authentication mechanisms. On the other hand, we use proven authentication mechanisms such as the use of authentication service providers coupled with a single sign-on. Fog consumers authenticate their identity using an external third-party authentication service (such as Google or Microsoft). On successful authentication, they are redirected to the DFR in the consortium. The external party communicates the authenticated identity to the central fog node provider.

6.2.2 Step 2: Collect the context-aware data of the fog consumer

The FNDE autonomously collects the fog consumer's physical location (latitude and longitude values from the GPS receiver) to carry out FND.

6.2.3 Step 3: Discover the nearest fog nodes based on context-aware parameters

The FNDE applies the selected fog search algorithm, namely k-nearest neighbor (KNN) [113], k-d tree [114], or brute force [115]. These search algorithms are commonly and successfully used in nearest neighbor searching (further details in Section

6.3). In our research, we implement four variants of the fog discovery algorithms (with each variant comprising one algorithm). It intelligently matches the nearest fog nodes on context-aware parameters such as identity and location.

6.2.4 Step 4: Provide a list of the nearest fog nodes

The FNDE provides a list of the nearest fog nodes. The fog consumer is presented with these available fog nodes and they can select one of them or seek help to select the most trusted fog node using FNSE.

Figure 4.4 describes the proposed FNDE framework and the detailed working of the FNDE. Algorithm 1 explains the working of the FNDE. Table 6.1 explains the semantics of all the algorithm variables in this chapter.

Table 6.1 : The definition of the algorithm variables

Variable name	**Variable definition**
FN_Identity	Fog Node Identity
FN_Location	Fog Node Location
FC_Identity	Fog Consumer Identity
FC_Location	Fog Consumer Location
K	The number of nearest neighbors

6.3 FNDE implementation

To find the closest and optimal nearest fog nodes, the FRC should be implemented first. We implemented the framework of FRC in a simulation environment using the OMNeT++ platform [111]. All modules in the FRC which includes FC, FN, CFR, and DFR modules are implemented and simulated using the OMNeT++

Algorithm 1 FNDE mechanism

```
Require: FN_Identity, FN_Location, FC_Identity, FC_Location
Ensure: K fog nodes
1: The FC provides their FC identity login credentials (username and password).
2: If authentication is successful, then:
3: The FNDE in the DFR autonomously collects the FC location (latitude and
   longitude values from the GPS receiver).
4: The FNDE finds the fog nodes nearest to the FC using one of the fog search
   algorithms.
5: The FNDE provides a list of the k-nearest fog nodes.
```

platform [111] (details in Chapter 5). For the discovery process, we implemented the FNDE in the DFR module with four different nearest neighbors search algorithms. The first algorithm is KNN with Euclidean distance [113]. The second algorithm is KNN with Manhattan distance [113]. The third algorithm is the k-d tree [114]. The fourth algorithm is brute force with haversine distance [115]. Then, we compare the results of the four selected methods. All four methods are implemented using the C++ language using OMNeT++ [111]. After conducting an extensive literature review in the field of service science, we identified that the most widely adopted and successful methods for finding and locating nearest neighbors are brute force, KNN and K-d tree.

First, we selected the brute force algorithm as one of the nearest neighbors searching methods because of its flexibility and accuracy. In the literature, the brute force algorithm is commonly used for k-nearest neighbor searches [116]. The brute force algorithm is the most naïve and simple neighbor search algorithm. It is an easy algorithm to implement and debug. The brute force algorithm works with any distance metric such as Euclidean, Manhattan, haversine etc., hence it is flexible. In this research, we used the haversine formula to measure the shortest distance between the fog consumer and fog nodes. The brute force algorithm using haversine formula provides the optimal and exact nearest neighbor, not the approximated one and hence this lends itself to higher accuracy. Then, we selected the KNN algorithm because of its efficiency, flexibility, and accuracy. The KNN algorithm is a supervised machine learning algorithm. In the literature, the KNN algorithm is widely, commonly, and successfully being used to find the nearest locations. Researchers in [117] [118] [119] used the KNN algorithm for searching and retrieving the nearest neighbors. The KNN is a robust algorithm to handle the noisy training data and hence it is efficient. This algorithm can handle large data and works with different distance matrices such as Euclidean, Manhattan, Hamming, and Minkowski and hence it is flexible. In this research, we implement the KNN algorithm with Euclidean and Manhattan distance metrices to investigate which metric will most accurately provide the optimal nearest fog nodes to the fog consumer. Finally, the K-d tree algorithm is also selected due to its efficiency. The K-d tree algorithm is the most successfully and commonly used method of searching, finding, and discovering nearest neighbors in the literature [120] [118] [121]. The K-d tree is a simple binary tree structure. It can handle moderate dimensional spaces more efficiently. Regarding the efficiency of this algorithm, the K-d tree can achieve faster search times. Each of the selected algorithms is explained with its corresponding pseudo-code in the following subsections 6.3.1, 6.3.2, 6.3.3 and 6.3.4.

6.3.1 KNN with Euclidean distance

First, we implement the KNN algorithm with Euclidean distance, which is used to estimate locations. The Euclidean distance is the most widely used distance with the KNN algorithm and measures the straight-line distance between two points in the Euclidean space [122]. The fog consumer node sends the following information:

a fog consumer's identification includes fog consumer ID and fog consumer IP, and geographical coordinates including latitude and longitude. The KNN calculates the distance between the fog consumer's location and the fog nodes in the area using Euclidean distance. Euclidean distance is calculated as the square root of the sum of the squared differences between the two points [123] [124]. The Euclidean distance is given by Equation 6.1:

$$D_{Euclidean}(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{6.1}$$

The K-fog nodes that have the shortest distance are provided in the discovery list. Algorithm 2 [123] [125] outlines the working of the KNN algorithm in the FNDE framework.

6.3.2 KNN with Manhattan distance

Second, we implement the KNN algorithm with Manhattan distance, which estimates locations. Similar to the method detailed in 6.3.1, the fog consumer node sends the following information: a fog consumer's identification represented as fog consumer ID and fog consumer IP and geographical coordinates including latitude and longitude. The KNN calculates the distance between the fog consumer's location and the fog nodes in the vicinity using Manhattan distance. The Manhattan distance is calculated as the sum of the absolute differences between the two points [124]. The Manhattan distance is given by Equation 6.2:

$$D_{Manhattan}(x, y) = \sum_{i=1}^{n}|x_i - y_i| \tag{6.2}$$

The K-fog nodes that have the shortest distance are provided in the discovery list. The below Algorithm 3 [125] [124] outlines the working of KNN algorithm in the FNDE framework.

Algorithm 2 KNN algorithm with Euclidean distance

```
input  : k: The number of nearest points to find.
         x: The location of the FC (fog consumer).
         Y: A list of locations of FNs (fog nodes).
output: kPoints: A list of k points that are the k-nearest points to the FC.
Initialize an empty list, distances, to store the computed Euclidean distances.
foreach y in Y do
    Calculate the Euclidean distance between x and y
    Append the distance to distances
end
Sort distances in ascending order
Initialize an empty list, kPoints, to store the k nearest points
for i from 0 to k-1 do
    Find the index of the $i-th$ smallest distance in distances
    Append the corresponding point from Y to kPoints
end
return kPoints
```

The K-fog nodes that have the shortest distance are provided in the discovery list. Algorithm 3 [125] [124] outlines the working of the KNN algorithm in the FNDE framework.

Algorithm 3 KNN algorithm with Manhattan distance

```
input  : $k$: The number of nearest points to find.
         $x$: The location of the FC (fog consumer).
         $Y$: A list of locations of FNs (fog nodes).
output: $kPoints$: A list of $k$ points that are the $k$-nearest points to the FC.
Initialize an empty list, $distances$, to store the computed Manhattan distances.
foreach $y$ in $Y$ do
    Calculate the Manhattan distance between $x$ and $y$ Append the distance to $distances$
end
Sort $distances$ in ascending order
Initialize an empty list, $kPoints$, to store the $k$ nearest points for $i$ from 0 to $k-1$ do
    Find the index of the $i$-th smallest distance in $distances$ Append the corresponding point from $Y$ to $kPoints$
end
return $kPoints$
```

6.3.3 K-d tree

Third, we employ the K-d tree for the KNN algorithm with Euclidean distance for the nearest location estimation. The K-d tree algorithm is a data structure that efficiently organizes points in a multidimensional space [126]. In our implementation, we focus on estimating the nearest locations of fog nodes to fog consumers. The fog consumer transmits relevant information including the fog consumer's identification, consisting of the fog consumer ID and fog consumer IP, as well as the geographical coordinates represented by latitudes and longitudes. The K-d tree algorithm then calculates the distance between the fog consumer's location and the fog nodes in the vicinity using the Euclidean distance metric. Euclidean distance is commonly utilized in a nearest neighbor search. By leveraging the K-d tree algorithm, we efficiently identify the K-nearest fog nodes based on their shortest Euclidean distances. These K fog nodes are subsequently included in the discovery list, providing valuable information for further processing and decision-making. This approach enables a more effective and expedited identification of the nearest fog nodes, contributing to the overall efficiency and performance of the system. Algorithm 4 [126] [127] explains the working of k-d tree in the FNDE framework.

Algorithm 4 K-d tree

```
input  : kdtree: A data structure of a k-d tree containing points of FNs' locations.
         $x$: The FC's location.
         $K$: The number of nearest neighbors.
output: nearestNeighbors: A list of $K$ the nearest neighbors to $x$.

Initialize an empty priority queue, nearestNeighbors, to store the $K$ nearest neighbors sorted by their Euclidean distance to $x$.
Search(kdtree, $x$, nearestNeighbors)
Procedure(Search(node, $x$, nearestNeighbors)) if node is null then
    return
end
Calculate the Euclidean distance between $x$ and the node's point
if the size of nearestNeighbors is less than $K$ or the distance is smaller than the maximum distance in nearestNeighbors then
    Add the node's point to nearestNeighbors along with its distance
    if the size of nearestNeighbors exceeds $K$ then
        Remove the point with the maximum distance
    end
end
Determine the splitting axis based on the depth of the current node
if $x$'s coordinate on the selected axis is less than the node's point's coordinate then
    Search(node's left child, $x$, nearestNeighbors)
end
```

```
    else
        Search(node's right child, $x$, nearestNeighbors)
    end
    Calculate the squared distance between the splitting axis and $x$'s coordinate on the selected axis
    if the size of nearestNeighbors is less than $K$ or the squared distance is smaller than the maximum distance in nearestNeighbors then
        Search(opposite child of node, $x$, nearestNeighbors)
    end
    return the $K$ nearest neighbors from nearestNeighbors EndProcedure
```

6.3.4 Brute Force

Fourth, we implement the linear search using the brute force KNN algorithm by measuring the distance between two points using haversine distance (spherical surface). The haversine distance is the angular distance between two points on the surface of a sphere. The haversine formula provides a good approximation of the distance between two points of the Earth's surface, with a less than 1% error on average [128]. The haversine formula is given by Equation 6.3 [129]:

$$D_{\text{Haversine}} = R \times 2 \times \arcsin\left(\sqrt{a + c}\right) \tag{6.3}$$

The value a is given by equation 6.4:

$$a = \sin^2\left(\frac{\Delta\text{lat}}{2}\right) \tag{6.4}$$

Moreover, the value c is given by equation 6.5:

$$c = \cos(\text{lat}1) - \cos(\text{lat}2) \times \sin^2\left(\frac{\Delta\text{long}}{2}\right) \tag{6.5}$$

where R = earth's radius 6371 km, Δ lat = (lat2 − lat1), and Δ long = (long2 − long1).

The brute force method is the most basic and simple to find the exact nearest locations. Algorithm 5 [130] [126] explains the working of the brute force algorithm in the FNDE framework.

Algorithm 5 Brute Force

```
input  : $k$: The number of nearest points to find.
         $x$: The location of the FC (fog consumer).
         $Y$: A list of locations of FNs (fog nodes).
output: $kPoints$: A list of $k$ points that are the $k$ nearest points to the FC.

Initialize an empty list, $distances$, to store the computed haversine distances.
foreach $y$ in $Y$ do
    Compute the haversine distance between $x$ and $y$. Add the computed distance to $distances$.
end
Sort $distances$ in ascending order. Initialize an empty list, $kPoints$, to store the $k$ nearest points.
for $i$ from 0 to $k - 1$ do
    Find the index of the $i$-th smallest distance in $distances$. Add the corresponding point from $Y$ to $kPoints$.
end
return $kPoints$.
```

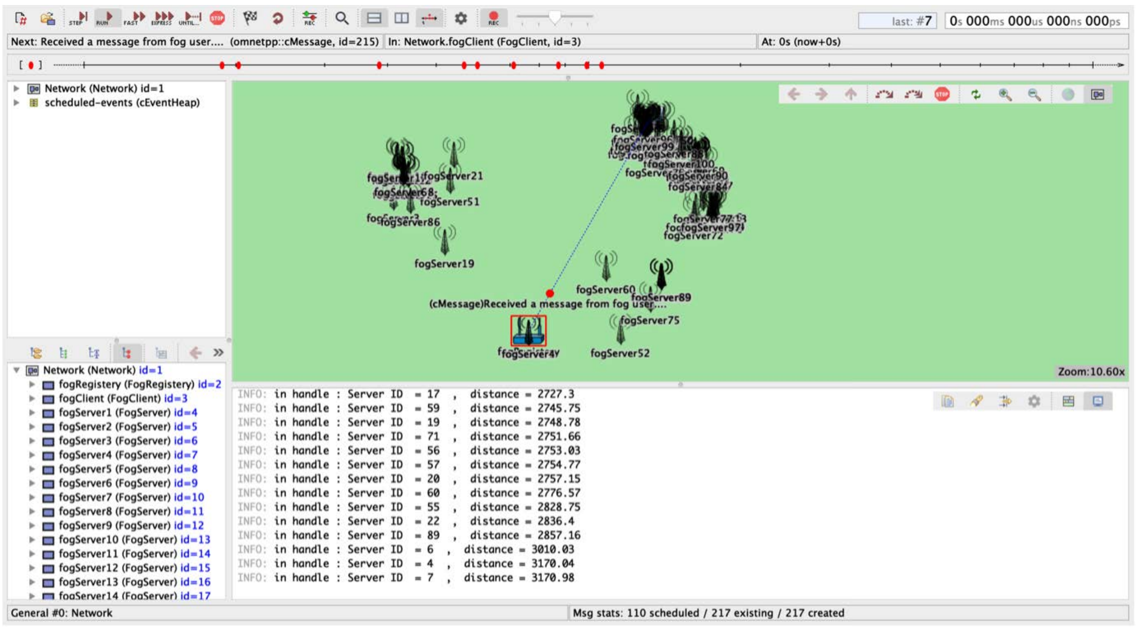
6.4 Experiments

The dataset used in this research is a set of EUA datasets [112] which includes the context data of fog consumers and fog nodes that we need in this research. There are 95562 edge servers which are considered to be fog nodes in this research. The attributes of fog nodes are SiteID, Name, Latitude, Longitude. There are also 4748 end users who are considered to be fog consumers in this research. In this research, for fog consumers (end users), we use the following attributes fog consumer ID, IP, Latitude, and Longitude (explained in detail in Chapter 5). We conducted extensive experiments on the simulated network with a different number of fog nodes and different fog consumers. The experiments include finding the 10-nearest fog nodes for five different fog consumers. We selected five different fog consumers as detailed in Table 5.3.

For each fog consumer, we find the 10-nearest fog nodes using the four proposed methods with a different number of fog nodes. We used 5 iterations. The first iteration has 100 fog nodes. The second iteration has 500 fog nodes. The third iteration has 1000 fog nodes. The fourth iteration has 1500 fog nodes. The fifth iteration has 2000 fog nodes. Then for each fog consumer, the 10-nearest fog nodes will be obtained using:

1. KNN with Euclidean distance
2. KNN with Manhattan distance
3. K-d tree
4. Brute force

When a fog node is added to the network, it communicates with the nearest DFR and sends a cMessage which includes the fog node's context date. In OMNeT++, a cMessage is a fundamental class that represents a message in the simulation. It is used to model the exchange of information or events between simulation modules or components [111]. Then, when the fog consumer module establishes the session to discover the nearest fog nodes, the latitude and longitude of the fog consumer's current location is sent to the DFR module as a cMessage. The FNDE in the DFR module will find the nearest fog nodes using the search algorithm and then sends the list to the fog consumer. We undertook the same processing for all iterations of 100, 500, 1000, 1500, and 2000 fog nodes for each fog consumer. Figures 6.1, 6.2, 6.3, 6.4, and 6.5 show the simulation of 100, 500, 1000, 1500, 2000 fog nodes respectively using OMNeT++. In Section 6.5, we evaluate the results of the nearest neighbors' fog nodes by comparing the results of the four methods. We validated the results by calculating the evaluation metrics of the four methods using well-known and accepted metrices, namely precision, recall, F1 score and accuracy to obtain the most accurate and optimal method of finding the nearest neighbors' fog nodes.



```
Next: Received a message from fog user....  (omnetpp::cMessage, id=215)  In: Network.fogClient (FogClient, id=3)  At: 0s (now+0s)
```

```
Network (Network) id=1
  fogRegistery (FogRegistery) id=2
  fogClient (FogClient) id=3
  fogServer1 (FogServer) id=4
  fogServer2 (FogServer) id=5
  fogServer3 (FogServer) id=6
  fogServer4 (FogServer) id=7
  fogServer5 (FogServer) id=8
  fogServer6 (FogServer) id=9
  fogServer7 (FogServer) id=10
  fogServer8 (FogServer) id=11
  fogServer9 (FogServer) id=12
  fogServer10 (FogServer) id=13
  fogServer11 (FogServer) id=14
  fogServer12 (FogServer) id=15
  fogServer13 (FogServer) id=16
  fogServer14 (FogServer) id=17
```

```
INFO: in handle : Server ID  = 17  ,  distance = 2727.3
INFO: in handle : Server ID  = 59  ,  distance = 2745.75
INFO: in handle : Server ID  = 19  ,  distance = 2748.78
INFO: in handle : Server ID  = 71  ,  distance = 2751.66
INFO: in handle : Server ID  = 56  ,  distance = 2753.03
INFO: in handle : Server ID  = 57  ,  distance = 2754.77
INFO: in handle : Server ID  = 20  ,  distance = 2757.15
INFO: in handle : Server ID  = 60  ,  distance = 2776.57
INFO: in handle : Server ID  = 55  ,  distance = 2828.75
INFO: in handle : Server ID  = 22  ,  distance = 2836.4
INFO: in handle : Server ID  = 89  ,  distance = 2857.16
INFO: in handle : Server ID  = 6  ,  distance = 3010.03
INFO: in handle : Server ID  = 4  ,  distance = 3170.04
INFO: in handle : Server ID  = 7  ,  distance = 3170.98
```

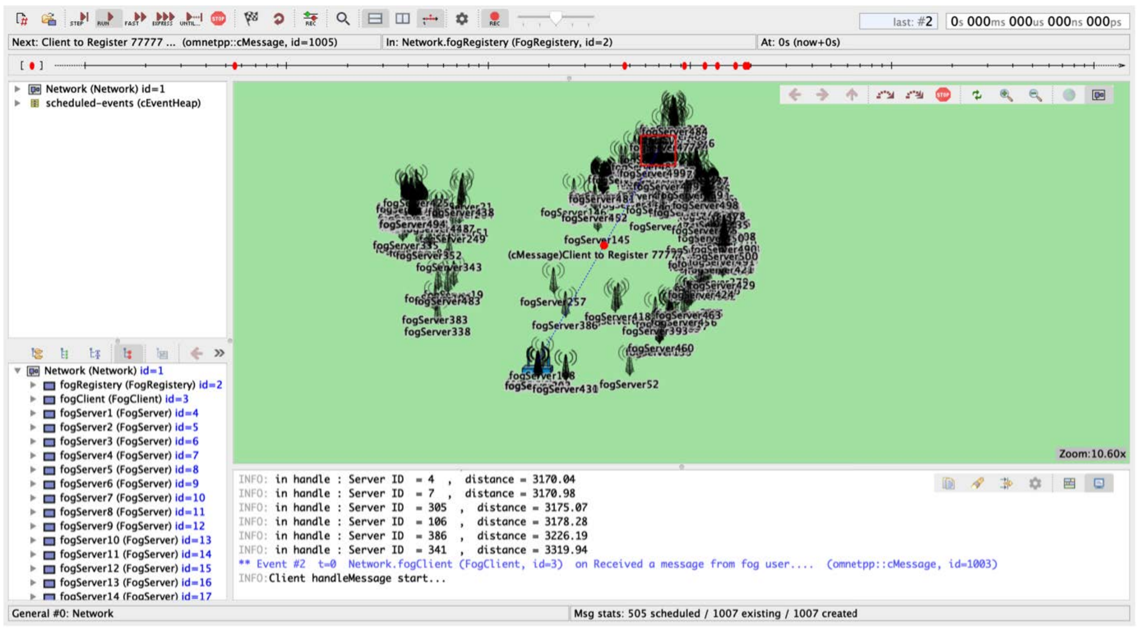
```
General #0: Network          Msg stats: 110 scheduled / 217 existing / 217 created
```

Figure 6.1 : Simulating 100 fog nodes using OMNeT++

6.5 Evaluation and discussion

In this section, we evaluate the results of the nearest neighbors' fog nodes by comparing the four methods. We validate the results by calculating the evaluation metrices of the four methods using precision, recall, F1 score and accuracy to obtain the optimal method of finding the accurate nearest neighbors fog nodes.

The validation process is as follows:

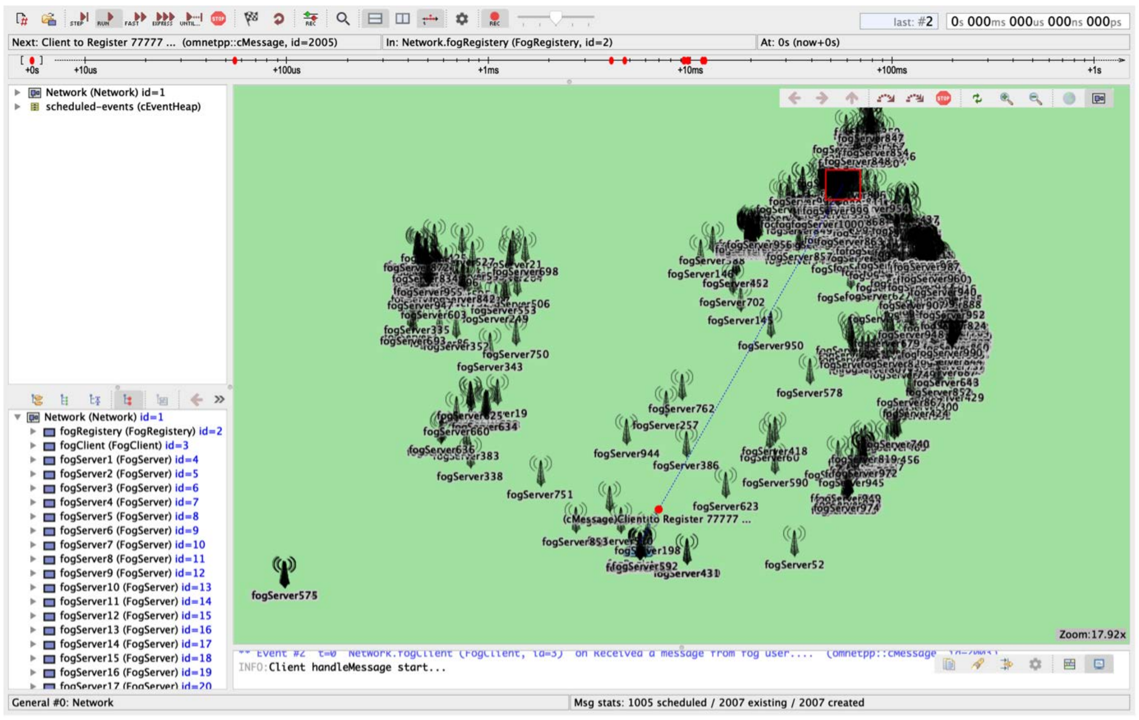


```
Next: Client to Register 77777 ...  (omnetpp::cMessage, id=1005)    In: Network.fogRegistery (FogRegistery, id=2)    At: 0s (now+0s)
```

```
INFO: in handle : Server ID  = 4   ,  distance = 3170.04
INFO: in handle : Server ID  = 7   ,  distance = 3170.98
INFO: in handle : Server ID  = 305  ,  distance = 3175.07
INFO: in handle : Server ID  = 106  ,  distance = 3178.28
INFO: in handle : Server ID  = 386  ,  distance = 3226.19
INFO: in handle : Server ID  = 341  ,  distance = 3319.94
** Event #2  t=0  Network.fogClient (FogClient, id=3)  on Received a message from fog user....  (omnetpp::cMessage, id=1003)
INFO:Client handleMessage start...
```

```
General #0: Network    Msg stats: 505 scheduled / 1007 existing / 1007 created
```

Figure 6.2 : Simulating 500 fog nodes using OMNeT++

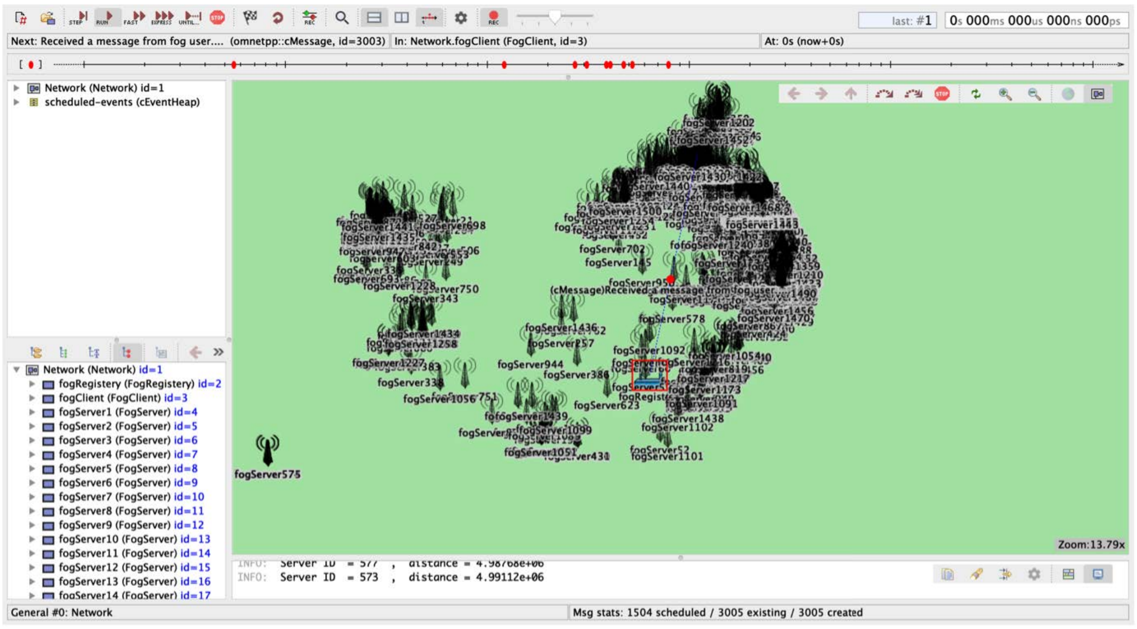


```
Next: Client to Register 77777 ...  (omnetpp::cMessage, id=2005)    In: Network.fogRegistery (FogRegistery, id=2)    At: 0s (now+0s)
```

```
INFO:Client handleMessage start...
```

```
General #0: Network    Msg stats: 1005 scheduled / 2007 existing / 2007 created
```

Figure 6.3 : Simulating 1000 fog nodes using OMNeT++


```
Next: Received a message from fog user....  (omnetpp::cMessage, id=3003)   In: Network.fogClient (FogClient, id=3)   At: 0s (now+0s)
```

```
Network (Network) id=1
scheduled-events (cEventHeap)
```

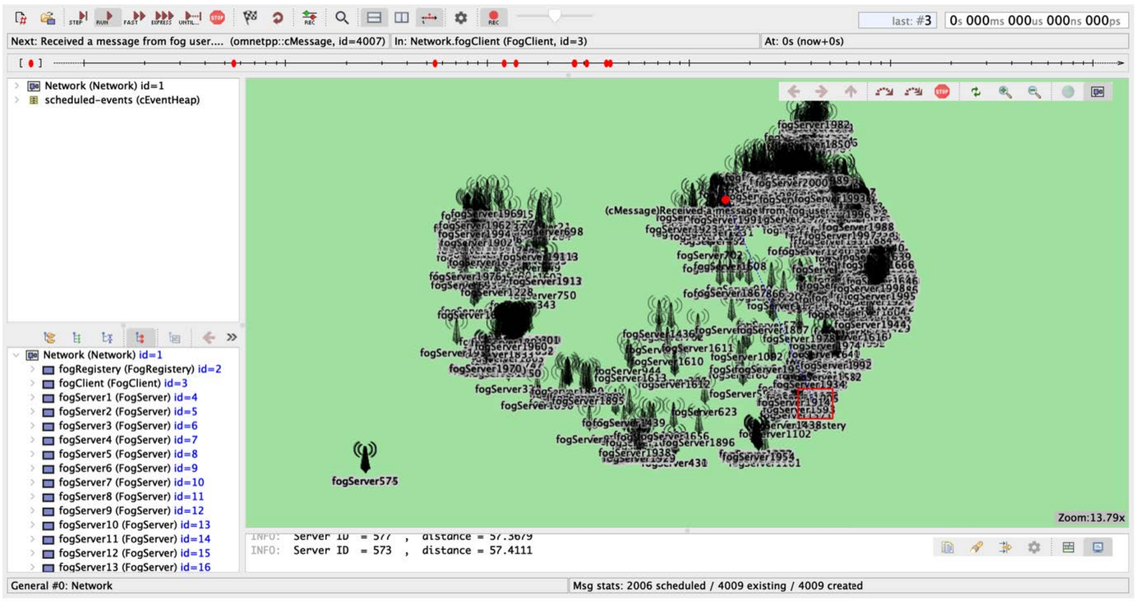
```
Network (Network) id=1
  fogRegistery (FogRegistery) id=2
  fogClient (FogClient) id=3
  fogServer1 (FogServer) id=4
  fogServer2 (FogServer) id=5
  fogServer3 (FogServer) id=6
  fogServer4 (FogServer) id=7
  fogServer5 (FogServer) id=8
  fogServer6 (FogServer) id=9
  fogServer7 (FogServer) id=10
  fogServer8 (FogServer) id=11
  fogServer9 (FogServer) id=12
  fogServer10 (FogServer) id=13
  fogServer11 (FogServer) id=14
  fogServer12 (FogServer) id=15
  fogServer13 (FogServer) id=16
  fogServer14 (FogServer) id=17
```

```
INFO:   Server ID  = 577  ,  distance = 4.98768e+06
INFO:   Server ID  = 573  ,  distance = 4.99112e+06
```

```
General #0: Network                Msg stats: 1504 scheduled / 3005 existing / 3005 created
```

last: #1
0s 000ms 000us 000ns 000ps
(cMessage)Received a message from fog user
fogServer575
Zoom:13.79x

Figure 6.4 : Simulating 1500 fog nodes using OMNeT++


```
Next: Received a message from fog user....  (omnetpp::cMessage, id=4007)   In: Network.fogClient (FogClient, id=3)   At: 0s (now+0s)
```

```
Network (Network) id=1
scheduled-events (cEventHeap)
```

```
Network (Network) id=1
  fogRegistery (FogRegistery) id=2
  fogClient (FogClient) id=3
  fogServer1 (FogServer) id=4
  fogServer2 (FogServer) id=5
  fogServer3 (FogServer) id=6
  fogServer4 (FogServer) id=7
  fogServer5 (FogServer) id=8
  fogServer6 (FogServer) id=9
  fogServer7 (FogServer) id=10
  fogServer8 (FogServer) id=11
  fogServer9 (FogServer) id=12
  fogServer10 (FogServer) id=13
  fogServer11 (FogServer) id=14
  fogServer12 (FogServer) id=15
  fogServer13 (FogServer) id=16
```

```
INFO:   Server ID  = 577  ,  distance = 57.3679
INFO:   Server ID  = 573  ,  distance = 57.4111
```

```
General #0: Network                Msg stats: 2006 scheduled / 4009 existing / 4009 created
```

last: #3
0s 000ms 000us 000ns 000ps
(cMessage)Received a message from fog user
fogServer575
Zoom:13.79x

Figure 6.5 : Simulating 2000 fog nodes using OMNeT++

Step 1: Initialization process: The initialization process comprises the following steps:

(a) Specify the number of fog nodes, starting with 100 nodes. The number of fog nodes in each iteration is increased by 500.

(b) Determine the number of fog registries, starting with 5 registries.

(c) Determine the number of iterations = n times. The value of the parameters for (b) and (c) vary from one iteration to the next.

Number of iterations = 5, the number of fog nodes in each iteration is increased by 500.

i. First iteration: fog nodes =100

ii. Second iteration: fog nodes =500

iii. Third iteration: fog nodes =1000

iv. Fourth iteration: fog nodes =1500

v. Fifth iteration: fog nodes =2000

Figure 5.2 Figure 6.6 illustrates the setup of the OMNeT++ simulator, which includes 100 fog nodes, one fog consumer, four distributed fog registries and one central fog registry.

Step 2: Implement the four nearest neighbor algorithms including the two KNNs, K-d tree, and brute force for FND in the DFR module.

Step 3: The system administrator selects a random fog consumer A from the available fog consumers and asks it to carry out context-aware FND. The system administrator knows the closest context-aware fog nodes for the selected fog consumer A; however, this information is unknown to node A. An overview of this steps is shown in Figure 6.6.

Step 4: Use well-known and accepted metrices such as precision, recall and F1 score as discussed in Section 4.7.1 to compare the three methods.

Step 5: Repeat steps 3 and 4 n times.

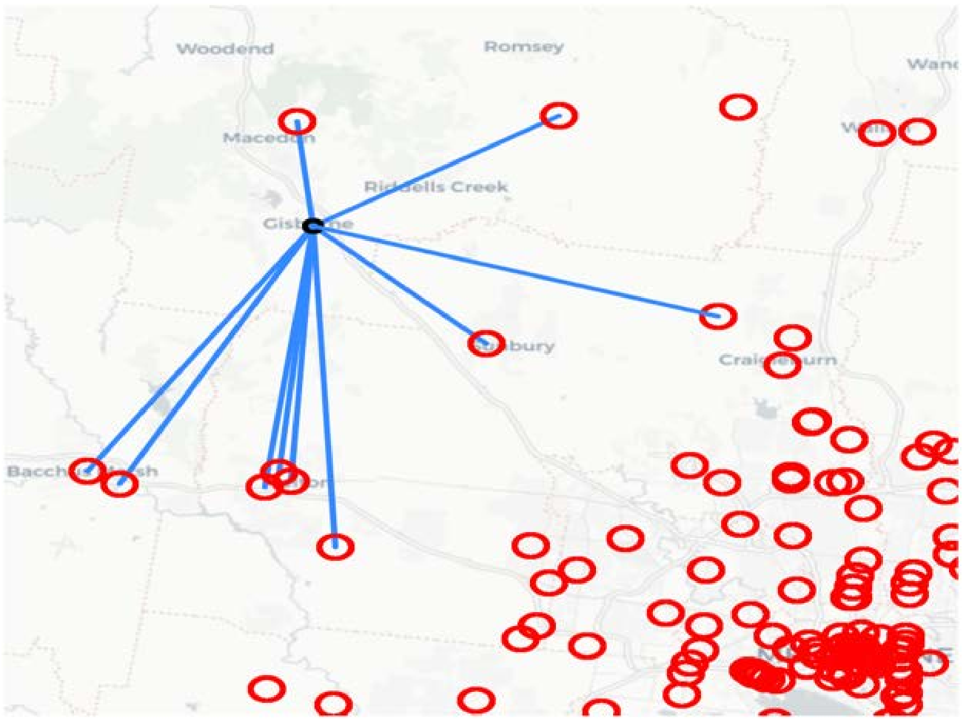


Figure 6.6 : Actual location of fog nodes

Figure 6.7, Figure 6.8, Figure 6.9, Figure 6.10 and Figure 6.11 illustrate the results of the four methods with a varying number of fog nodes.

The overall accuracy of the four methods is shown in Figure 6.12.

Figures 6.7 – 6.12 show that the brute force algorithm with haversine distance has the highest accuracy of all the methods, outperforming K-d tree and KNN and also obtains a value of 1 for precision, recall, and F1 score. Precision measures the proportion of true positive results of all the positive results. In the evaluation, the brute force method consistently achieves a precision of 1, indicating that all the fog nodes selected were true positive matches. This exceptional precision demonstrates the accuracy of the haversine distance in identifying the nearest fog nodes accurately. Recall, also known as sensitivity, measures the proportion of true positive

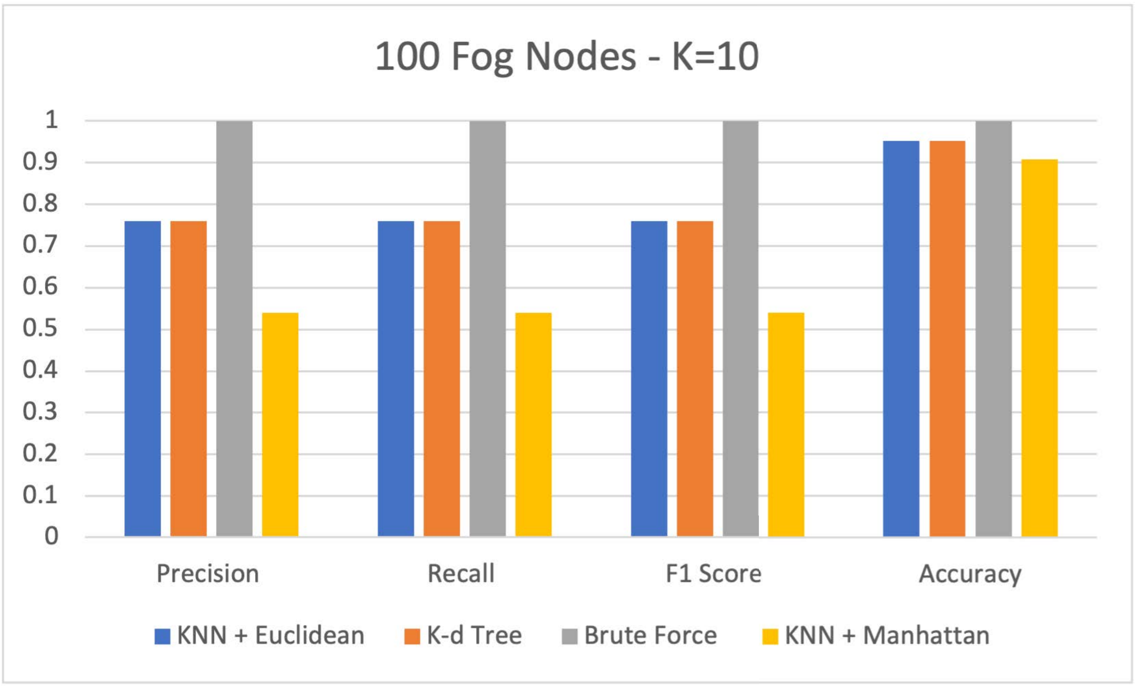


Figure 6.7 : Evaluation results of the network with 100 fog nodes

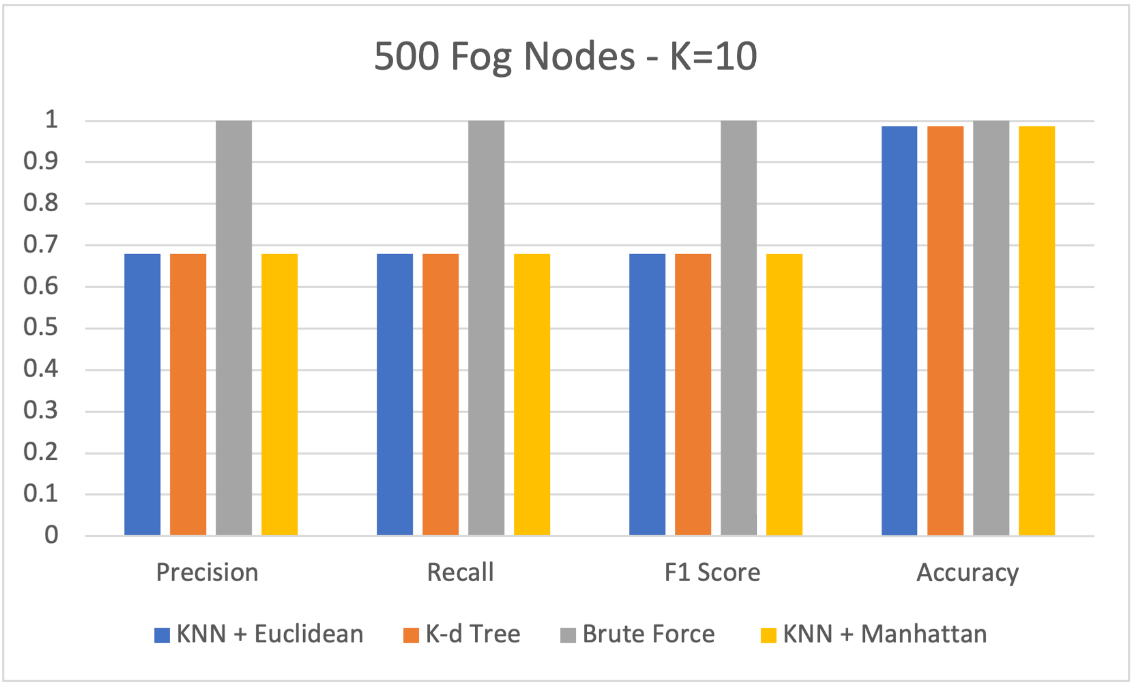


Figure 6.8 : Evaluation results of the network with 500 fog nodes

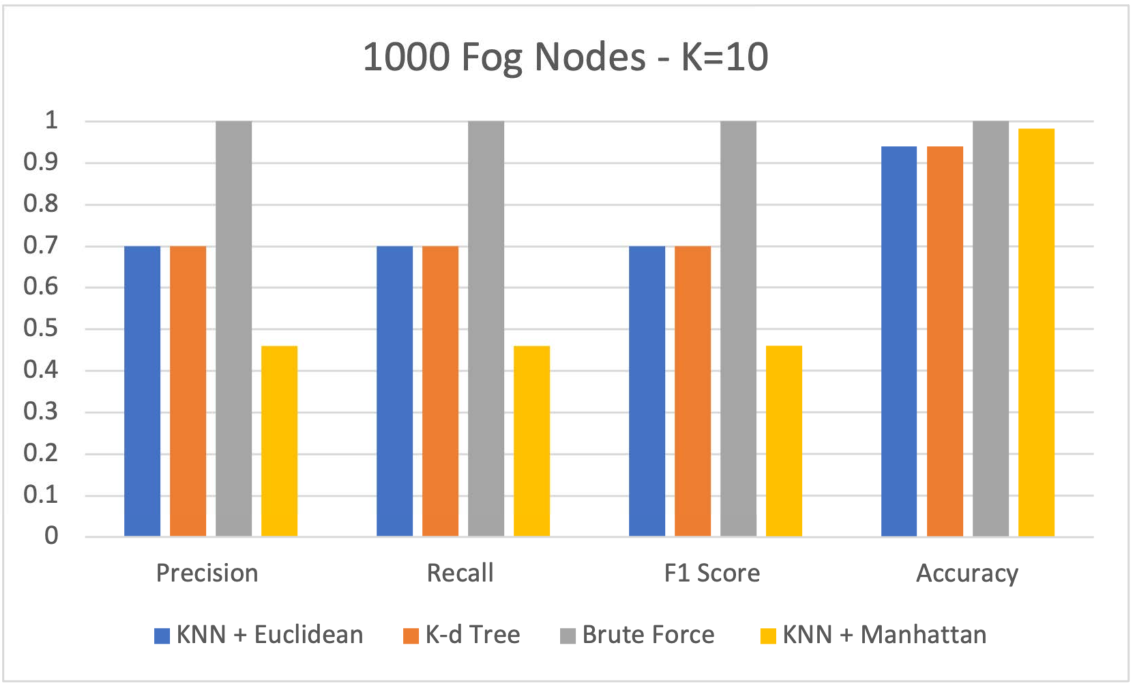


Figure 6.9 : Evaluation results of the network with 1000 fog nodes

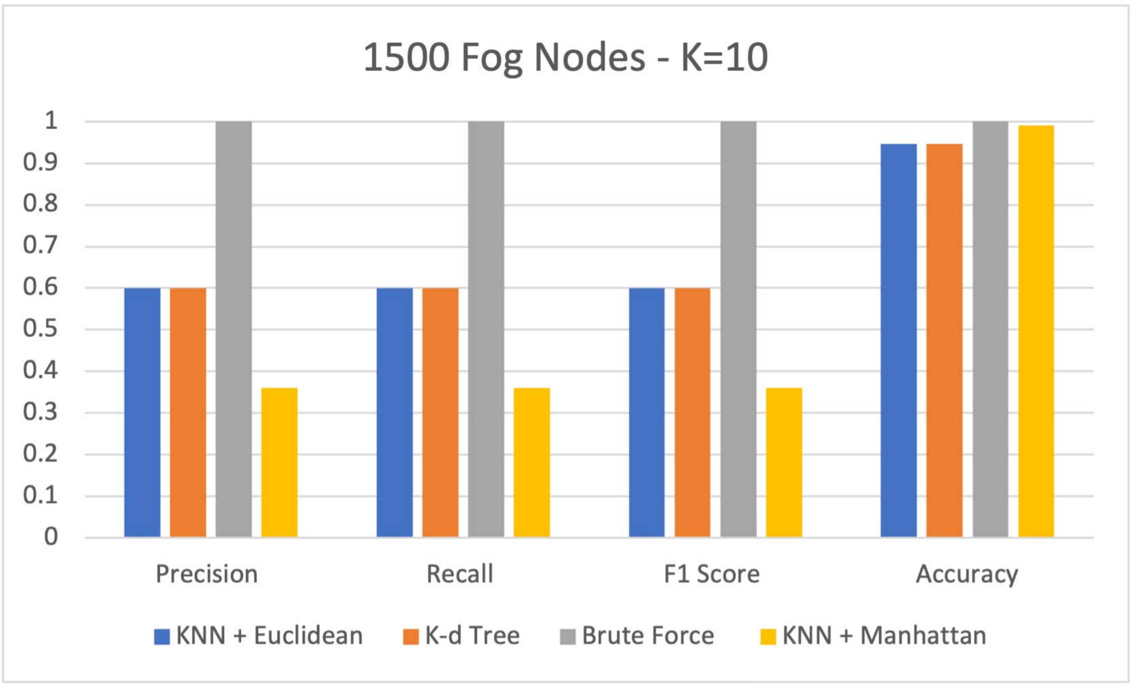


Figure 6.10 : Evaluation results of the network with 1500 fog nodes

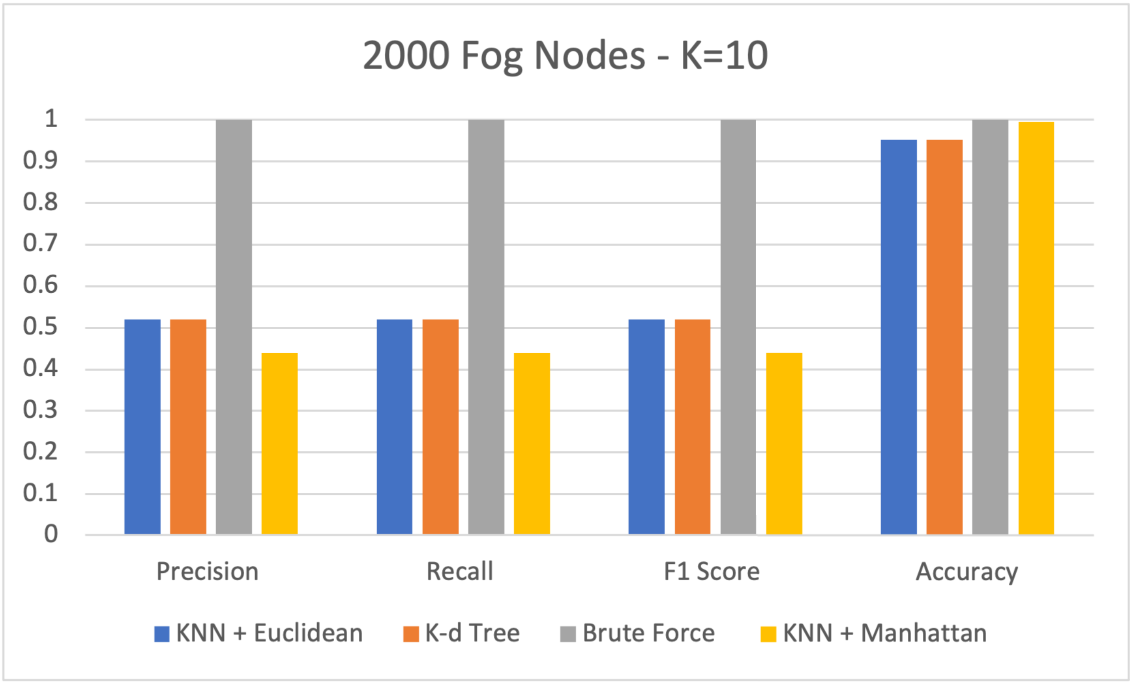


Figure 6.11 : Evaluation results of the network with 2000 fog nodes

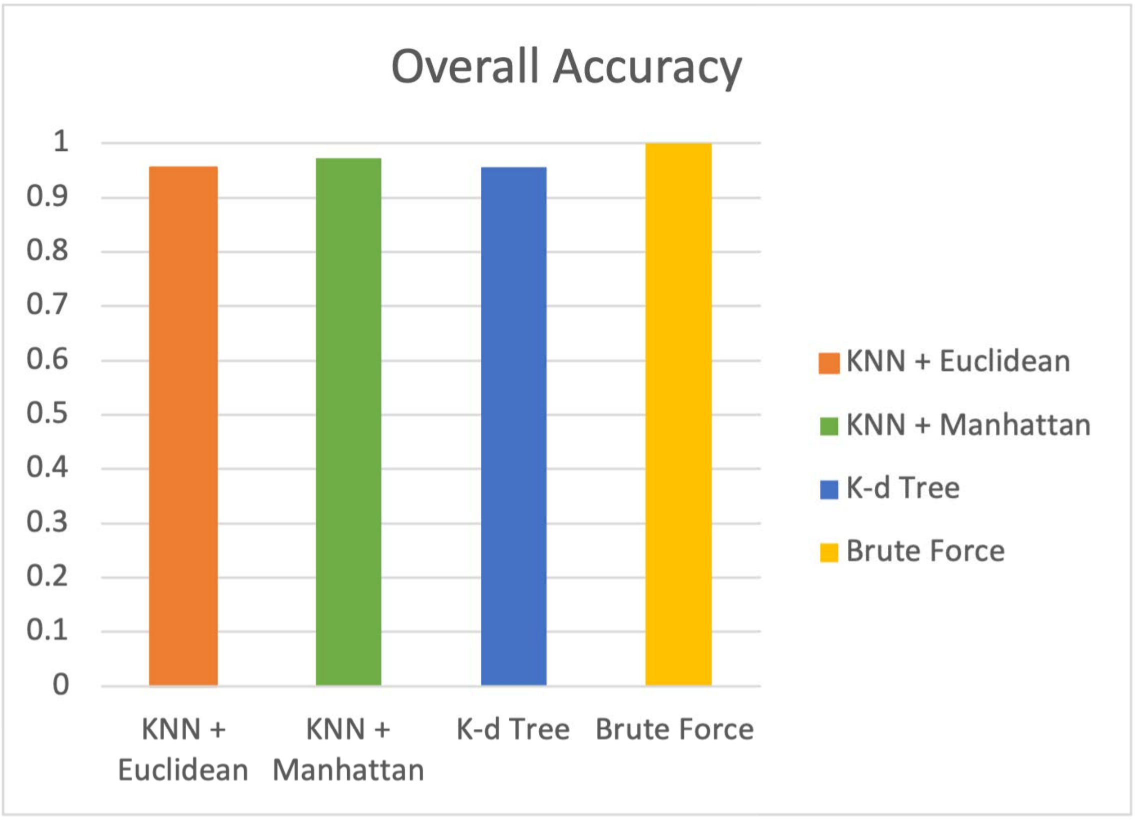


Figure 6.12 : Average evaluation results

results correctly identified from all the actual positive instances. The brute force method again achieves a recall of 1, indicating that it successfully identified all the relevant fog nodes in the network. The F1 score combines precision and recall to provide a balanced metric that considers both false positives and false negatives. The brute force method consistently attained an F1 score of 1, indicating that it balances precision and recall perfectly. The accuracy metric measures the overall correctness of the method in correctly identifying the nearest fog nodes. Once again, the haversine distance method stands out with an accuracy of 1, showcasing its effectiveness in obtaining accurate results. The evaluation clearly demonstrates the superior performance of the brute force method with haversine distance in the context of FND The perfect scores in precision, recall, F1 score, and accuracy indicates that the haversine distance, a measure of distance between two points on the Earth's surface, performs exceptionally well in FND, providing accurate nearest neighbor results. However, KNN and K-d tree have equal precision and recall values and their accuracy is also close to 1 which means these two approaches give optimal results as well because both methods use Euclidean distance to find the nearest neighbors. Both the KNN method with Euclidean distance and the K-d tree method show stable performance across all the tested fog node scenarios. The consistency in their results indicates that they are reliable methods for context-aware FND in various fog computing environments. The KNN method with Manhattan distance achieves a decent performance for smaller fog node sizes but experiences a decline as the number of fog nodes increases. This observation suggests that KNN with Manhattan distance might be more suitable for smaller-scale fog networks but might not scale well to larger and more complex environments. Overall, the results indicate that the brute force method consistently outperforms the other three methods in all evaluation metrics. This suggests that the haversine distance is highly effective in accurately discovering the nearest fog nodes in the simulated network, regardless of the number of fog nodes. Further, the K-d tree and KNN methods exhibit stable performance, but their results fall short compared to the brute force method. Additionally, the Manhattan distance approach in the KNN method performs relatively better for smaller fog node sizes but deteriorates with an increasing number of fog nodes.

However, a key point of consideration in the selection of an optimal fog search algorithm is the complexity of the algorithm itself. We use algorithm complexity as a key input to its selection. The fog node registry carries out FND on multiple occasions for fog consumers. Even a very minor incremental reduction in the complexity of the search algorithm makes a huge difference to the load on the fog discovery server or on the overhead associated with the discovery process. The complexity of brute force is `O(n^m)`. However, using this linear search method is not suitable for the approach proposed in this research because when the data increases (number of fog nodes), the complexity of finding the nearest locations also increases. The complexity of the KNN method is `O(n)`. The downsides are that KNN is very sensitive to the curse of dimensionality and expensive to compute with a `O(n)` calculation. In contrast, the complexity of the K-d tree method is `O(log(n))`. K-d tree is guaranteed log2 n depth where n is the number of points in the set.

For this reason, the execution time of the three methods is calculated to evaluate their complexity. We computed the execution time of the three methods with a different number of fog nodes (100, 500, 1000, 1500, and 2000). We ran the simulation five times then the average of the execution times is obtained. All experiments were conducted on a Mac Pro with 2.8 GHz Quad-Core Intel Core i7 processor and 16 GB 2133 MHz LPDDR3 memory. Figures 6.13 – 6.17 show the average execution times of the three proposed methods with a different number of fog nodes. The average of the overall execution times is shown in Figure 6.18.

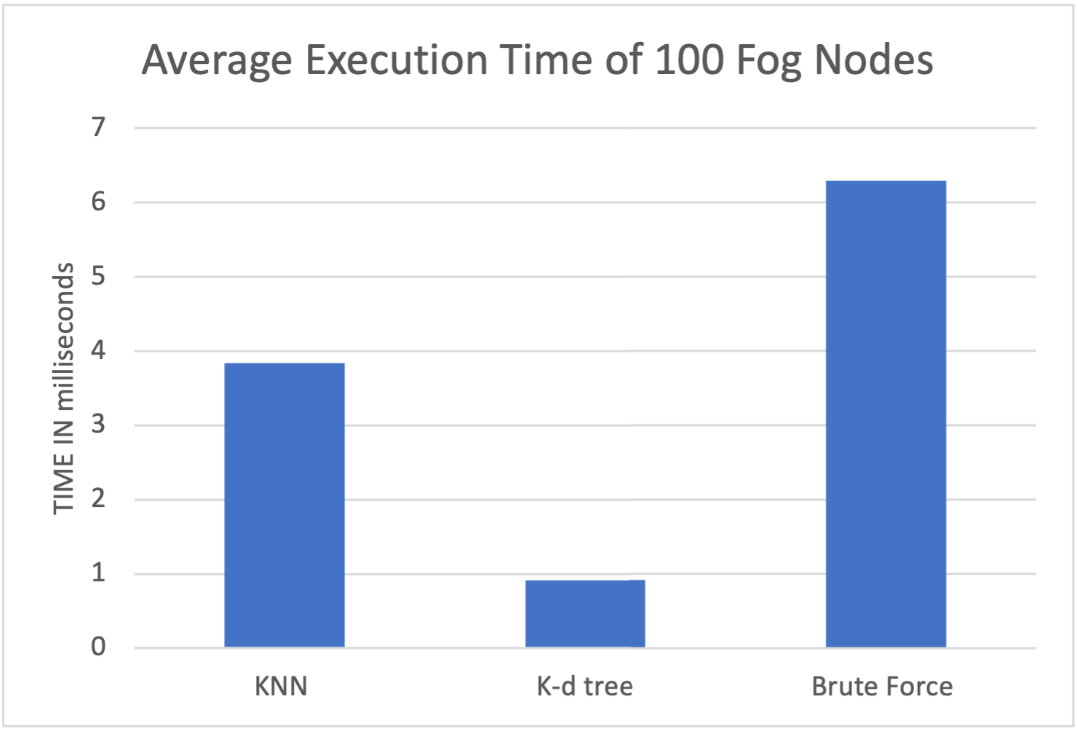


Figure 6.13 : The average execution times of the three methods when fog nodes = 100

The comparative analysis in Figures 6.13 – 6.18 reveals significant differences in the execution time between the k-d tree, KNN, and brute force methods when searching for the nearest fog nodes. The k-d tree method consistently outperforms the other two approaches in terms of execution time. Specifically, when the number of fog nodes is 100, the k-d tree method demonstrates remarkable efficiency, with an execution time of less than 1 millisecond. In contrast, the KNN method requires more than 3 milliseconds, while the brute force method exceeds 6 milliseconds. These findings highlight the advantage of utilizing the k-d tree algorithm for the efficient and speedy identification of the nearest fog nodes.

Furthermore, as the number of fog nodes increases, the execution time for all three methods also increases. For instance, when there are 100 fog nodes, the execution time for the k-d tree algorithm is approximately 0.92 milliseconds. However,

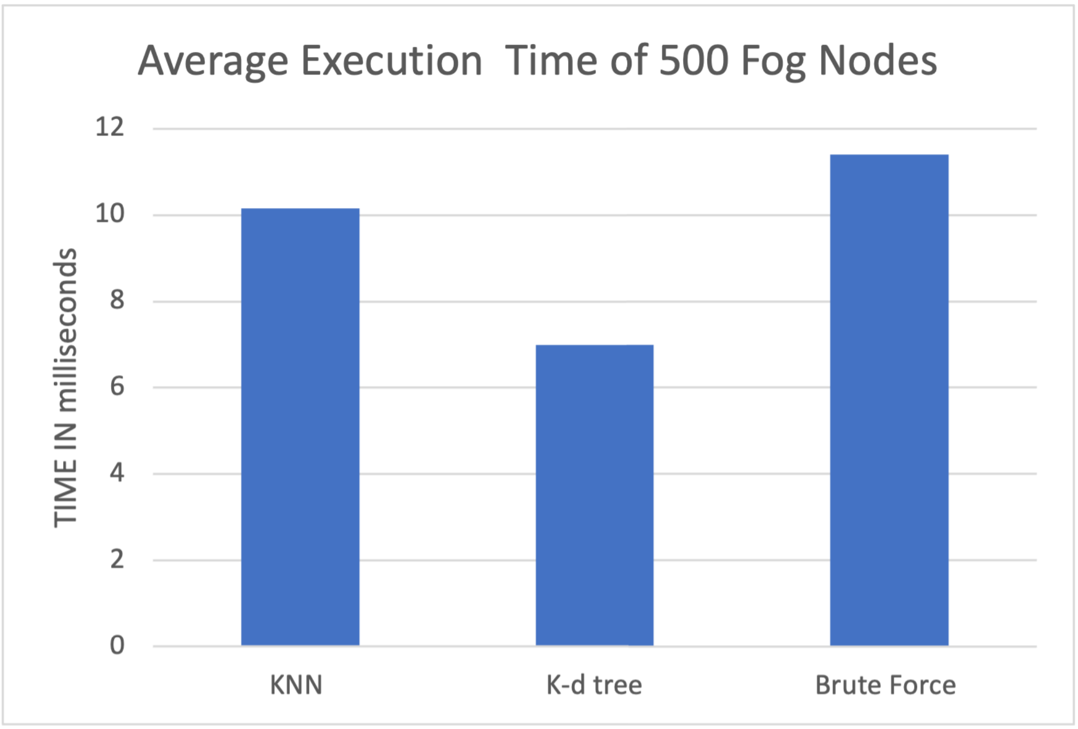


Figure 6.14 : The average execution times of the three methods when fog nodes = 500

when the number of fog nodes escalates to 1000, the execution time of the k-d tree method rises to approximately 11.8 milliseconds. This observation emphasizes the impact of scalability on the execution time of the k-d tree algorithm.

Table 6.2 : The percentage differences of execution time between k-d tree, KNN, and brute force

	K-d tree	**KNN**	**Brute force**
K-d tree	-	50% more time	83% more time
KNN	33% less time	-	22% more time
Brute force	45% less time	18% less time	-

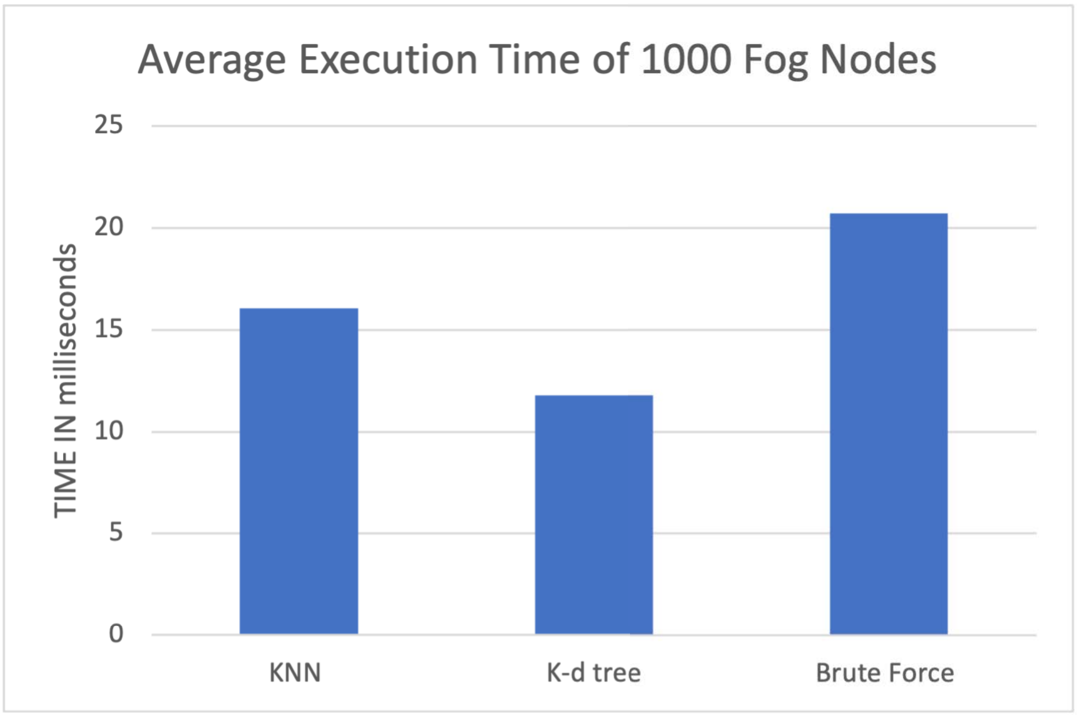


Figure 6.15 : The average execution times of the three methods when fog nodes = 1000

The results presented in Figure 6.18 and Table 6.2 offer valuable information on the performance of various methods, such as KNN, k-d tree, and brute force, particularly in terms of execution time. The observed percentage differences shed light on the comparative advantages of each approach. As Table 6.2 shows, the KNN method, although widely used and effective, exhibits a higher execution time compared to the k-d tree. It requires 50% more time, indicating that the KNN algorithm performs relatively more slowly in finding the nearest neighbors in the given context. On the other hand, the brute force method, while providing relatively accurate results, demands a significantly higher execution time. It takes 83% more time than the k-d tree approach, making it the least efficient option in terms of execution speed. The k-d tree method emerges as a promising alternative, demonstrating time savings of 33% compared to KNN and 45% compared to brute force. These findings

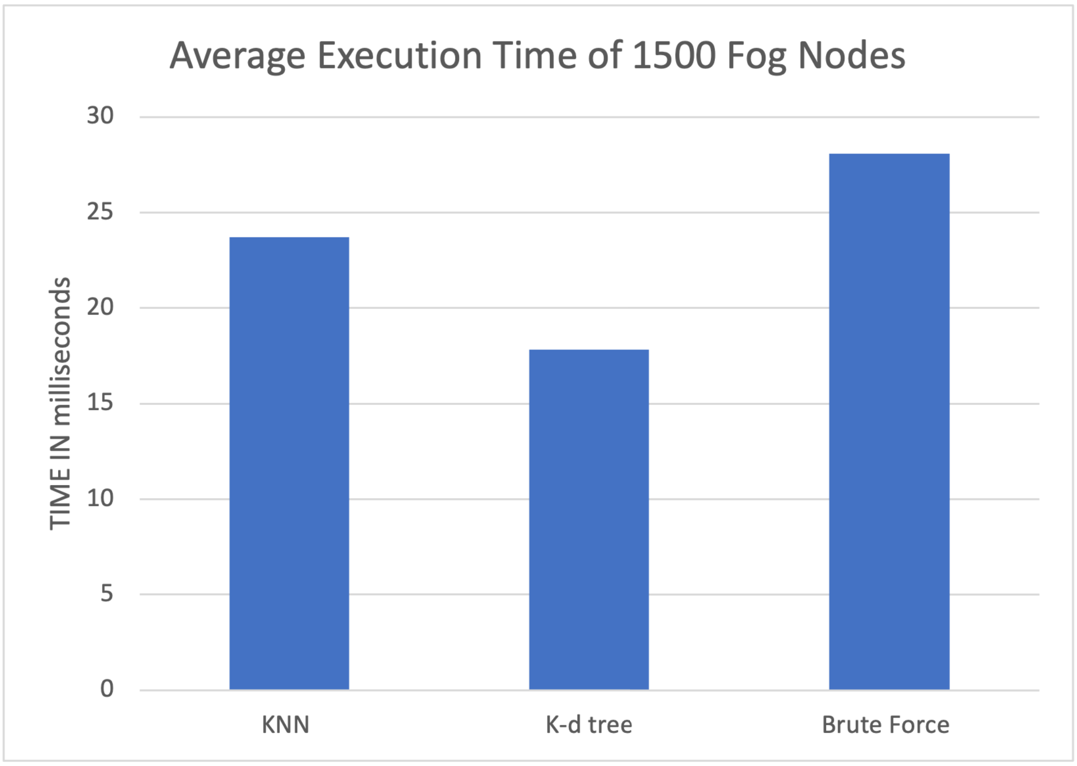


Figure 6.16 : The average execution times of the three methods when fog nodes = 1500

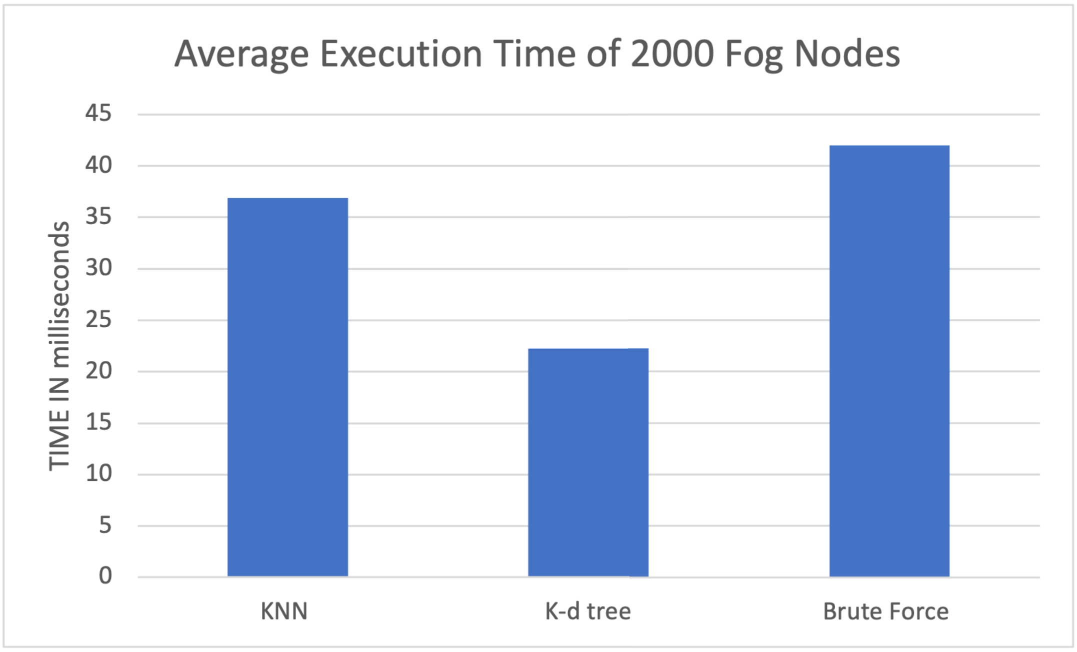


Figure 6.17 : The average execution times of the three methods when fog nodes = 2000

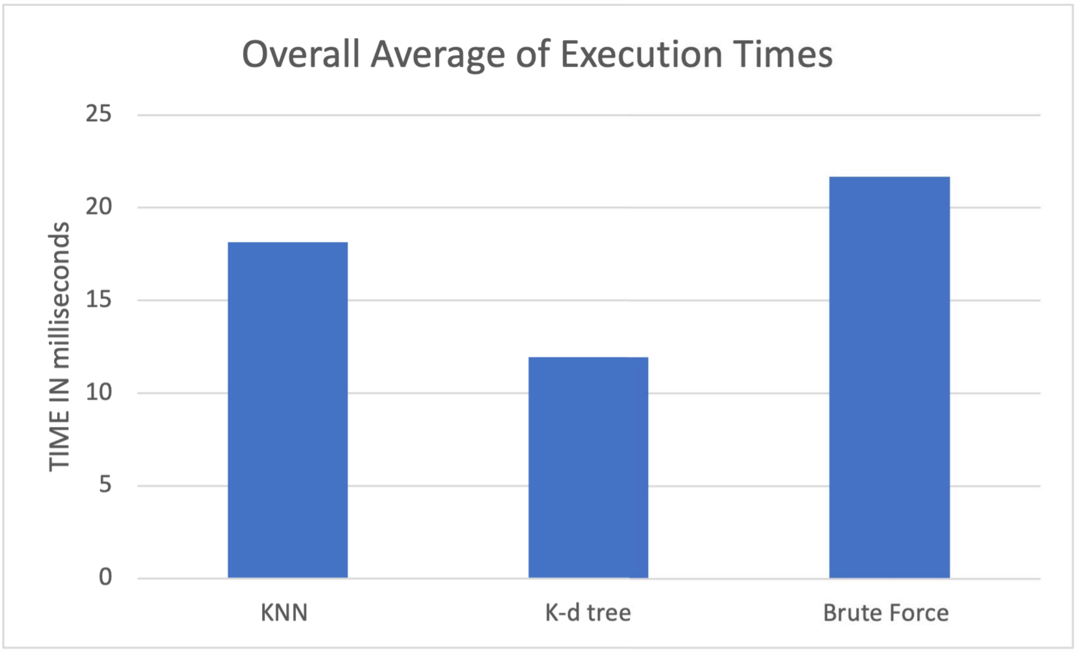


Figure 6.18 : The overall average of the execution times of the three methods

highlight the efficiency of the k-d tree in finding the nearest fog nodes, making it a favourable choice for fog consumers seeking faster results. Overall, the experiment findings concluded that the k-d tree method is highly efficient in discovering the nearest fog nodes compared to both the KNN and brute force methods, particularly in scenarios with a larger number of fog nodes. Its ability to consistently deliver faster execution times makes it an optimal choice for applications in term of time-sensitive operations and rapid decision-making.

However, choosing the appropriate method for FNDE is a critical factor in obtaining optimal and efficient outcomes, with consideration of both accuracy and execution time being of paramount importance. When deciding on the optimal method, fog consumers should carefully consider their priorities. If their primary concern is execution time, the k-d tree method proves to be the preferred choice. By leveraging its efficiency, even minor improvements in execution time can yield substantial benefits, such as reducing the load on the FNDE's fog discovery module. However, in scenarios where accuracy takes precedence over time considerations, the brute force method with haversine distance becomes the recommended approach. The brute force algorithm achieved the highest accuracy of all the tested methods. Moreover, it demonstrated remarkable precision, recall, and F1 score values, all reaching 1, as shown in Figure 6.12. This method ensures optimal results by focusing on accuracy, regardless of the increased execution time. There is no doubt that the brute force algorithm with haversine distance demonstrates excellent accuracy and precision, making it an attractive choice when accuracy is of the utmost importance. This method ensures the accurate detection of nearest neighbors by exhaustively searching all possible combinations. To handle large datasets or to respond to real-time applications, this approach can require a lot of computational power. To sum up, fog consumers need to evaluate their specific requirements, balancing the trade-off between accuracy and execution time when selecting the most suitable method for their FNDE. The findings shown in Figures 6.12, 6.18 and Table 6.2 provide valuable insights to inform their decision-making process and achieve optimal outcomes in FND.

6.6 Conclusion

In this chapter, we proposed an intelligent mechanism for FND based on the context-aware data of fog nodes and fog consumers. We proposed the FNDE framework to enable fog consumers to discover appropriate fog nodes in a context-aware manner. The FNDE must use the optimal fog nearest neighbors' search algorithm to reduce time and increase accuracy. Four fog nearest neighbors search algorithms, namely KNN with Euclidean distance, KNN with Manhattan distance, K-d tree, and brute force with haversine distance were implemented and evaluated in the FNDE using the OMNeT++ platform. Several simulation experiments were conducted, and the results show that the K-d tree search algorithm improves the overall system performance. The K-d tree algorithm achieves a high accuracy result of 95% and requires less time to find the nearest fog nodes than the KNN methods and brute force with haversine distance. The thrust of this research is to build an intelligent single criterion approach for computing the proximity between a fog consumer and fog node. Our approach is agnostic of the parameter used to quantify proximity. In our future work, we will propose intelligent multi-criteria-driven node discovery approaches based on diverse parameters that manifest proximity, such as bandwidth, physical distance, latency etc.

Chapter 7

Developing an intelligent approach to select a reliable fog node based on the trust value

7.1 Introduction

The selection process for a fog node is critical as it can greatly impact the performance and functionality of the fog network. It is essential to choose a fog node that has the necessary processing power, memory capacity, and communication capabilities to handle the tasks required by the fog consumer. It is important that when a fog consumer selects a fog node, they feel confident the right fog node has been chosen. The aim of this research is to help the fog consumer to select a trusted fog node in the network. The trusted fog node as defined in Section 3.2.16 is the fog node that has been verified to be reliable and capable of performing its intended functions by assigning a trust value as evidence. In this research, the verification process to determine whether a fog node is trusted or not is applied by the TEE. The TEE works as an agent and is responsible to predict and evaluate the trust value of a fog node to assist the fog consumer to make a reliable FNS. The important factors in evaluating the trustworthiness of a fog node includes the reputation of the fog node provider, the experience of other fog consumers with the same or similar fog nodes, and the QoS of the fog node. Thus, selecting trusted fog nodes that satisfy the fog consumers' requirements is a major challenge that requires further investigation and research attention. Furthermore, there are several constraints that need to be balanced when selecting a fog node. Trust plays an important role in helping the fog consumer to find a reliable and trusted fog node. The selection process is best achieved using AI algorithms that assist in choosing an appropriate trusted fog node that will improve network performance and reduce bandwidth, latency, and energy consumption [47]. In this chapter, we present the detailed solution to developing an intelligent approach to predict the trust value of fog nodes. This approach helps the fog consumer to select a reliable and trusted fog node based on the predicted trust value. We present the FNSE as an intelligent and reliable FNS mechanism (Chapter 4). Within the FNSE, we proposed the TEE as an intelligent and reliable trust-based assessment in a distributed fog environment. The TEE is an intelligent model to predict the trust value of fog nodes to help the fog consumer select a reliable fog node based on the QoS factors of fog nodes in the fog environment.

In this chapter, Section 7.2 presents the architecture of the TEE framework. Section 7.3 describes the implementation of TEE and the prototype setup. Section 7.4 details the experiments and results. Section 7.5 provides a discussion of the evaluation results of the proposed methods. Finally, Section 7.6 concludes the chapter.

7.2 TEE framework

In this section, we describe the development of an intelligent mechanism for predicting the trust values of fog nodes to make a reliable selection decision. This will help fog consumers choose the most reliable fog node based on the trust value. To achieve this goal, we created the TEE within the FNSE to perform a trust-driven evaluation of fog nodes and make recommendations for selection. In this study, the trustworthiness of fog nodes is determined based on the QoS of the fog nodes they have previously provided. To predict the trust value of fog nodes, we utilized three AI-based prediction models, the fuzzy logic approach, the logistic regression approach, and the DNN approach. The goal is to determine which method will yield the best results. The overall workings of the TEE framework within FNSE are depicted in Figure 7.1. As shown in Figure 7.1, first, the FNSE collects the QoS data for each fog node with which they have previously interacted. The QoS parameters are response time, availability, throughput, successability, reliability, and rank for service quality. As fog node data were deficient, we utilized the QWS dataset [131] which gauges the QoS of actual web services to represent the QoS of fog nodes. In this case, due to the lack of a fog node dataset, the QoS of web services was regarded as equivalent to that of the fog nodes. The QoS parameters for each fog node are stored in the fog repository in the DFR along with the location-based context-aware fog node data. When the QoS data is added to the repository, the FRC should be synchronized and updated, as described in Chapter 5. The DFR uses the push synchronization mechanism to send data to the CFR. Then, the CFR broadcasts this update to all DFRs. Secondly, the FNSE predicts the trust value of the fog node using TEE. The TEE is responsible for predicting the trust value of each fog node based on the value of the QoS parameters using the fuzzy logic system, logistic regression or DNN. These methods determine the trust value of the fog nodes. Then, the evaluation metrices of accuracy, recall, precision, and F1 score are computed for all methods to find the best performing methods. Thirdly, after assigning the trust value to each fog node, the FNSE ranks the fog nodes based on their trust values in ascending order. Finally, the final ranked result is displayed to the user to assist them select one of the candidate fog nodes. Algorithm 6 presents the algorithm of the TEE based on fuzzy logic. Figure 7.2 shows the working of TEE based on fuzzy logic. Algorithm 7 presents the TEE algorithm based on logistic regression. Figure 7.11 shows the working of TEE based on logistic regression. Algorithm 8 presents the TEE algorithm based on DNN. Figure 7.12 shows the working of TEE based on DNN.

.

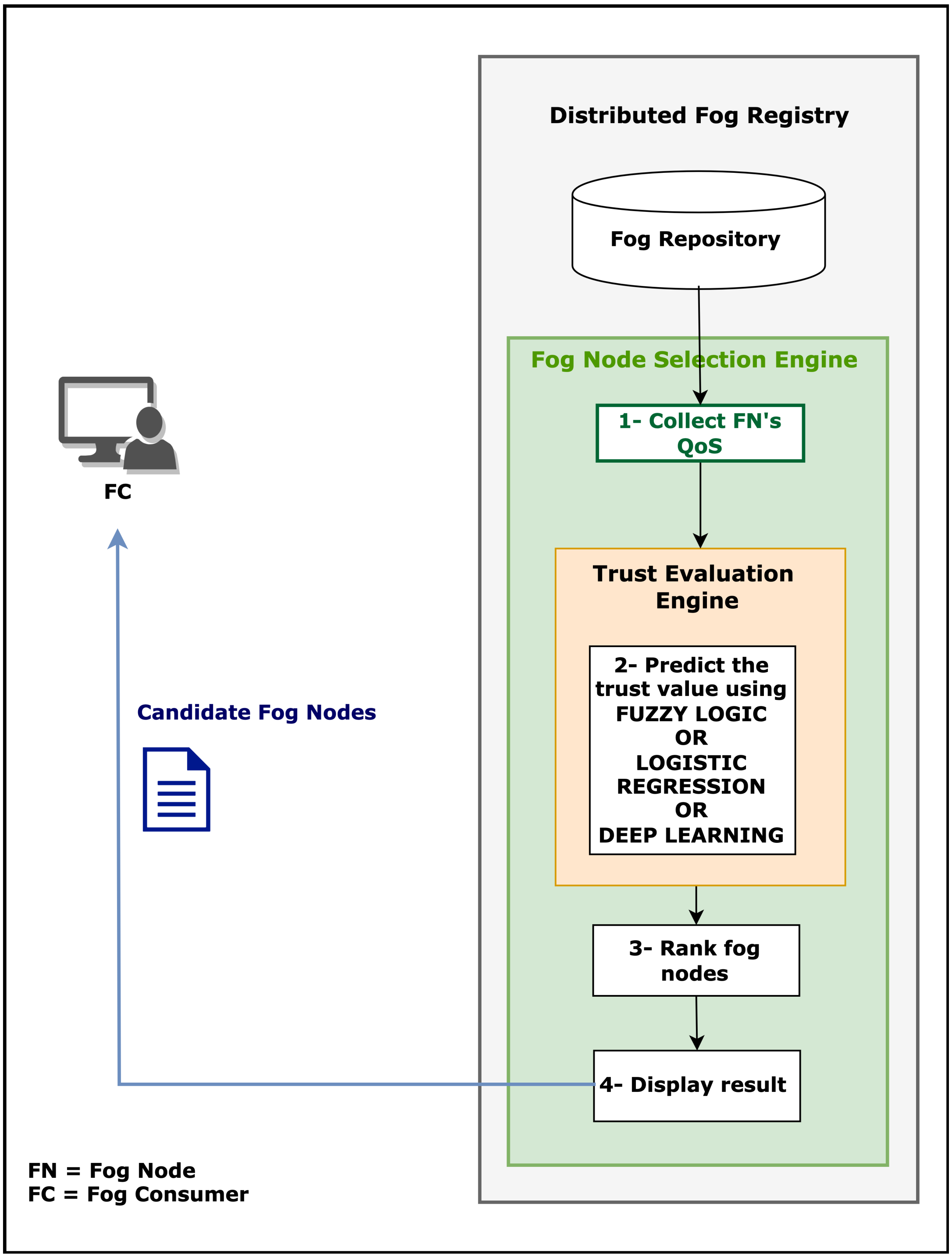


Figure 7.1 : Overview of the working of the FNSE and TEE workflow

Algorithm 6 Trust Evaluation Engine (TEE) Algorithm based on Fuzzy Logic

```
Require: QoS values for each fog node
Ensure: Ranked fog nodes based on trust value (fuzzy Logic)
 1: function PredictTrustValue(QoS values)
 2:     Fuzzify inputs: Convert crisp quantities into fuzzy quantities
 3:     Construct membership functions for inputs and outputs
 4:     Setup fuzzy rules
 5:     Compute the trust value using fuzzy logic system
 6:     De-fuzzify the output to get the trust value
 7:     return trust value
 8: end function each fog node
 9: TrustValue ← PredictTrustValue(QoS values)
10: Store TrustValue for fog node
11: Sort fog nodes based on TrustValue in ascending order
12: Display the ranked result to fog consumer
```

7.3 Implementation of TEE

In this section, we demonstrate the implementation of the three proposed TEE algorithms detailed in Algorithms 6, 7, and 8. The implementation of the TEE mechanism based on fuzzy logic is detailed in Section 7.3.1. The implementation of the TEE mechanism based on logistic regression is detailed on Section 7.3.2. The implementation of the TEE mechanism based on DNN is described in Section 7.3.3.

7.3.1 Implementation of the TEE based on fuzzy logic

Fuzzy logic provides a mathematical framework for reasoning with ambiguous and uncertain information, allowing for more flexible modeling, decision-making, and inference in a wide range of real-world applications. Fuzzy logic simulates

Algorithm 7 Trust Evaluation Engine (TEE) Algorithm based on logistic regression

Require: QoS values for each fog node

Ensure: Ranked fog nodes based on trust value (logistic regression)

```
 1: function PredictTrustValue(QoS values)
 2:     Begin
 3:         Preparing data
 4:         Building and training the logistic regression model
 5:         Testing and making predictions
 6:         return trust value predictions
 7:     End function each fog node
 8: TrustValue ← PredictTrustValue(QoS values)
 9: Store TrustValue for fog node
10: Sort fog nodes based on TrustValue in ascending order
11: Display the ranked result to fog consumer
```

human-like reasoning and decision-making in uncertain or imprecise environments. It provides a way to handle and reason with information that is vague, fuzzy, or characterized by degrees of membership or truth. In this research, we selected fuzzy logic to predict the trust value of fog nodes as a first approach because it is well-suited to handle the inherent uncertainty and imprecision associated with trust [132] [133]. Trust is a fuzzy concept that implies gradations of meaning. For a specific fog node, it is difficult to determine the exact value of the fog node trust value. Trust is not a binary concept, but rather a continuum that can vary in degrees or levels. Fog consumers may have different levels of trust in different situations or for different entities. In the existing literature, fuzzy logic has been widely employed in trust management, trust evaluation and trust prediction for web services, cloud services, etc. to help the selection, composition, and recommendation process. Sev-

Algorithm 8 Trust Evaluation Engine (TEE) Algorithm based on DNN

```
Require: QoS values for each fog node
Ensure: Ranked fog nodes based on trust value (DNN)
 1: function PredictTrustValue(QoS values)
 2:     Begin
 3:         Preparing data
 4:         Training the multilayer perceptron (MLP) neural network
 5:         Testing and making predictions
 6:         return trust value predictions
 7:     End function each fog node
 8: TrustValue ← PredictTrustValue(QoS values)
 9: Store TrustValue for fog node
10: Sort fog nodes based on TrustValue in ascending order
11: Display the ranked result to fog consumer
```

eral approaches are developed to enhance trust management and prediction based on fuzzy logic and provide sophisticated results. For example, in [132], the authors proposed the bi-directional fuzzy logic-based trust management system. This system aims to determine a node's trust level. It achieves this by synergistically considering multiple factors including quality of service, security level, social connections, historical reputation, and endorsements provided by nearby nodes. Furthermore, in [134] [133], fuzzy logic is used for cloud service trust-based selection process. According to [133], it is advantageous to use fuzzy logic in cloud service selection since it is capable of handling uncertainty and imprecise inputs as well as simulating human decision-making in a situation of uncertainty, indecision, and incompleteness. The proposed trust evaluation model incorporates fuzzy logic to derive trust values based on user feedback in terms of fuzzy linguistic terms. Also, in [135], the authors proposed a fuzzy-based trust management system to help cloud consumers identify trustworthy providers. Moreover, in [136], the authors proposed an evidence-based trust model using fuzzy logic to evaluate the trustworthiness of cloud services in real time. Furthermore, in [137], the authors designed a trust management model utilizing fuzzy logic. This model [137] helps consumers make well-informed decisions when choosing a cloud service provider (CSP) that aligns with their specific needs and preferences. Moreover, in the Web service environment, fuzzy logic is employed to carry out the trust management of web services for personalized service selection [24]. The study in [138] proposed a fuzzy trust management framework for reputation-based trust systems for Web services. The study in [139] proposed a credibility-based model for assessing Web service trust. Furthermore, the authors of the work in [140] demonstrated how to measure the trustworthiness of Web services using a fuzzy logic approach.

After analyzing the existing literature, we decided to adopt fuzzy logic to deal with the ambiguity and vagueness within the trust-based prediction process. Our fuzzy logic approach to the TEE mechanism predicts the trust value of fog nodes by considering various QoS factors such as response time, availability, throughput, successability, reliability, and rank for service quality (WsRF). The trust prediction of fog nodes is handled by the TEE in the FNSE. The working of TEE based on fuzzy logic is shown in Figure 7.2. In the TEE, we map the input space corresponding to the QoS factors to the output space which is a representation of the trust value. In this research, the input space consists of response time, availability, throughput, successability, reliability, and WsRF. The output space is the trust value. The input factors are all expressed using a fuzzy model which transforms the crisp variable of a particular input factor into a linguistic variable. We need a fuzzy inference method to perform the mapping from the input space to the output space. We use the Mamdani method [141] to perform this mapping. The fuzzy interface approach

[141] is implemented using MATLAB [142].

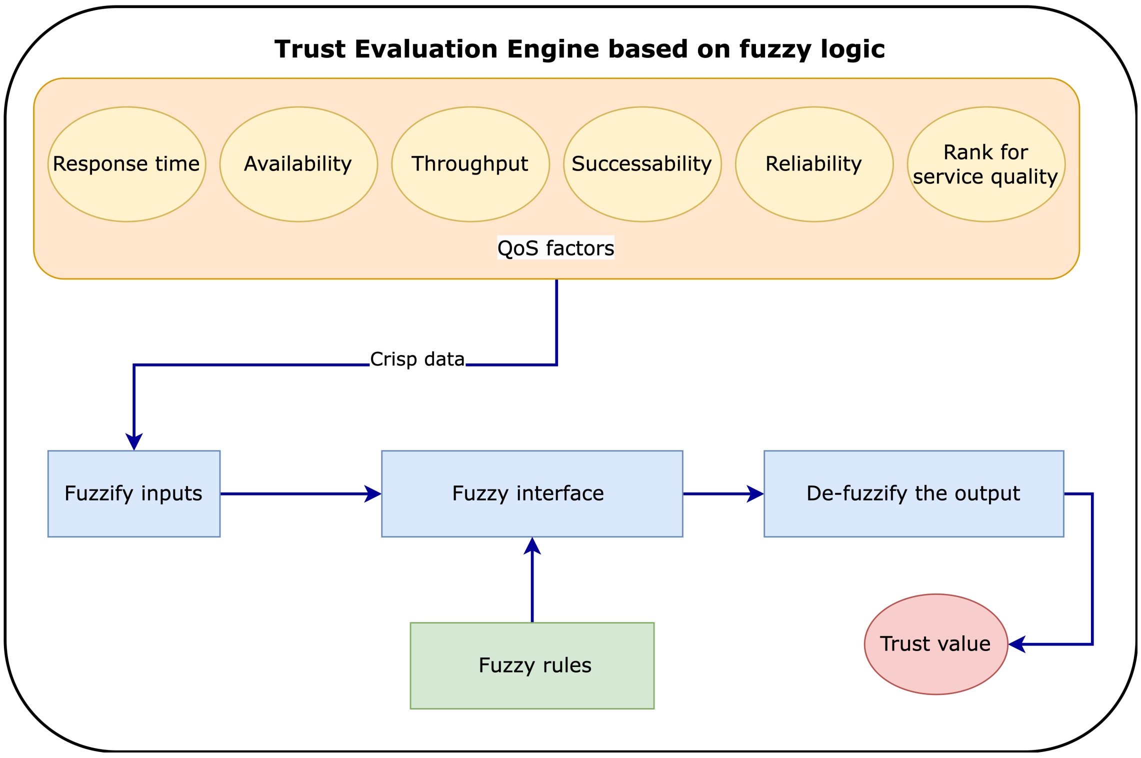


Figure 7.2 : Architecture of Trust Evaluation Engine based on Fuzzy logic

After collecting and storing the QoS value for each fog node, FNSE starts the selection process based on the QoS of fog nodes by predicting the trust value. TEE starts the prediction process based on the fuzzy interface using the following steps:

7.3.1.1 Step 1: Fuzzify inputs

The first step is the fuzzification process that converts crisp values into fuzzy values. This step outlines the definition of linguistic variables, which are inputs and outputs represented in simple terms. The trust value of a fog node is determined by six QoS factors: response time, availability, throughput, successability, reliability, and WsRF. These factors are assigned three linguistic variables, low, medium, and high. The specific range for each of these variables is listed in Table 7.1.

Additionally, we have defined three linguistic variables for the output, which is

Table 7.1 : Parametric range for the linguistic variables of input parameters

Input Parameter	**Low**	**Medium**	**High**
Response Time	0 - 299	300 - 999	1000 – 10,000
Availability	0 - 34	35 – 69	70 - 105
Throughput	0 – 9	10 – 19	20 -30
Successability	0 – 34	35 – 69	70 - 105
Reliability	5 – 14	15 – 39	40 - 100
WsRF	0 – 59	60 – 69	70 - 100

the trust value of the fog node: highly trustworthy, trustworthy, and untrustworthy. The range for each of these linguistic variables is presented in Table 7.2.

Table 7.2 : Parametric range for the linguistic variables of output parameters

Output Parameter	**Highly Trustworthy values range**	**Trustworthy values range**	**Untrustworthy values range**
Trust value	0 - 1	1 - 2	2 – 3

7.3.1.2 Step 2: Construct membership functions for inputs and outputs

In this step, we use the Gaussian membership function for all the input parameters, as shown in Figures 7.3, 7.4, 7.5, 7.6, 7.7 and 7.8. Furthermore, we use the triangular membership function for the output parameter as shown in Figure 7.10

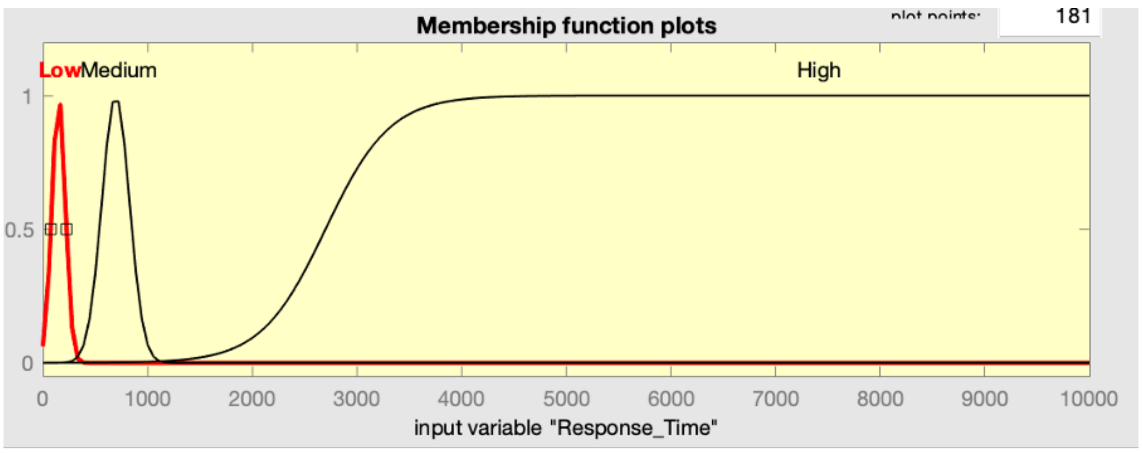


Figure 7.3 : Membership function of response time input

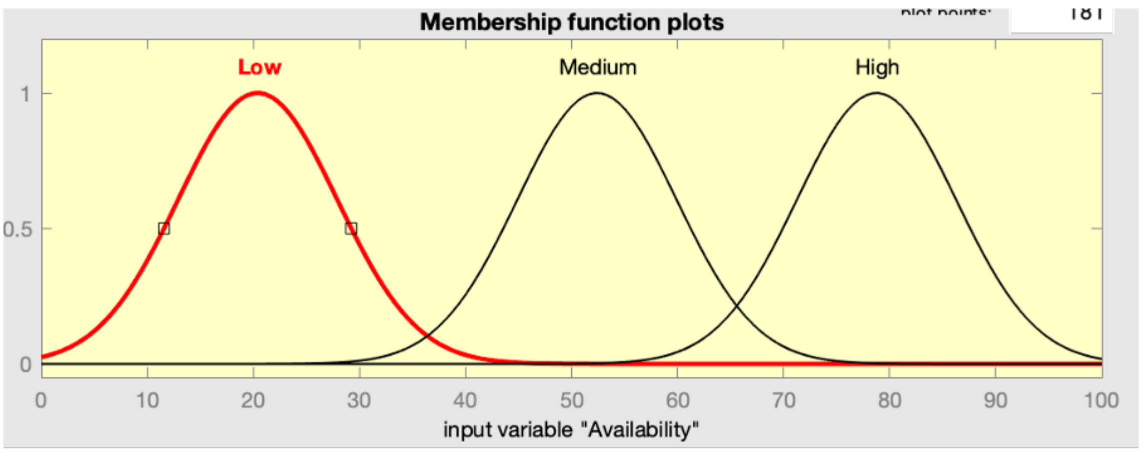


Figure 7.4 : Membership function of availability input

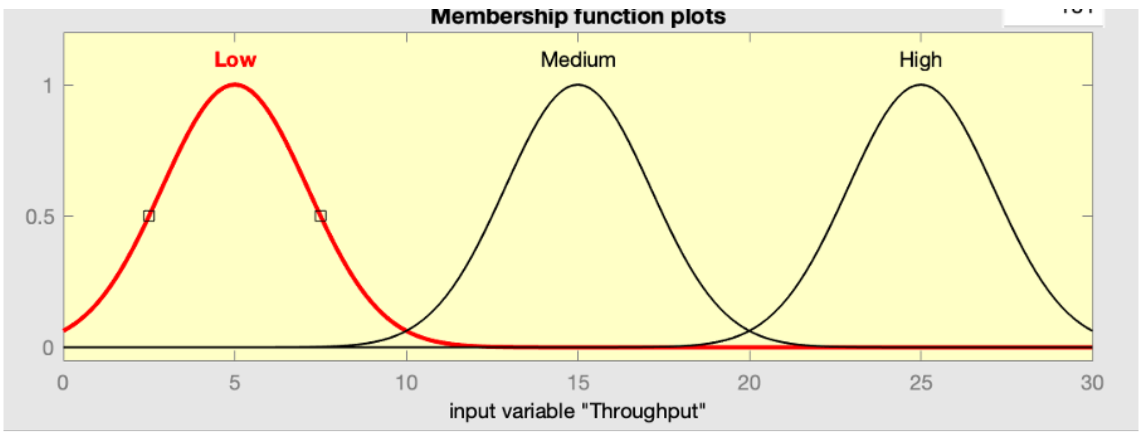


Figure 7.5 : Membership function of throughput input

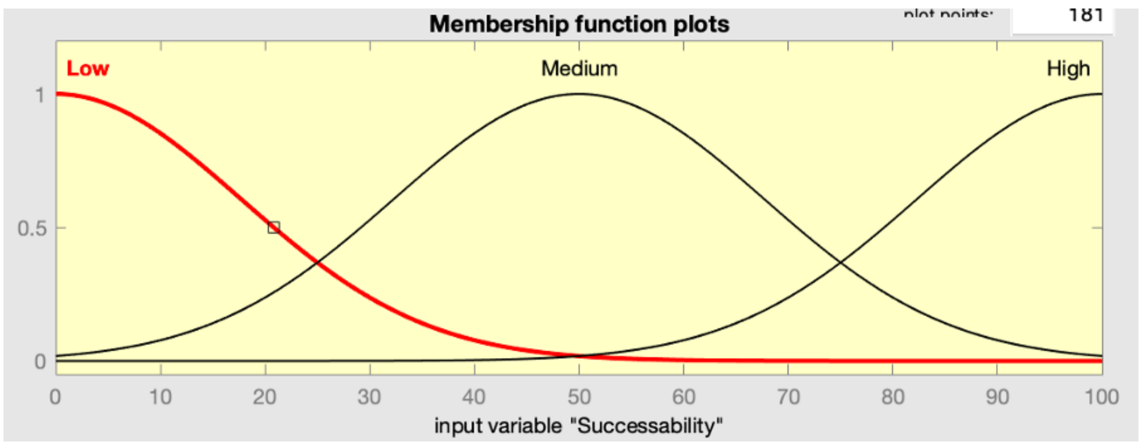


Figure 7.6 : Membership function of successability input

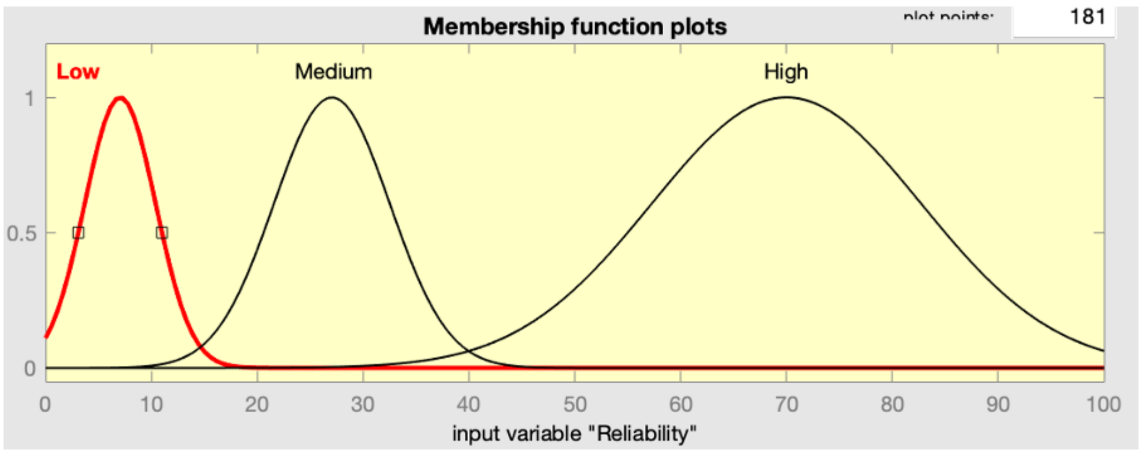


Figure 7.7 : Membership function of reliability input

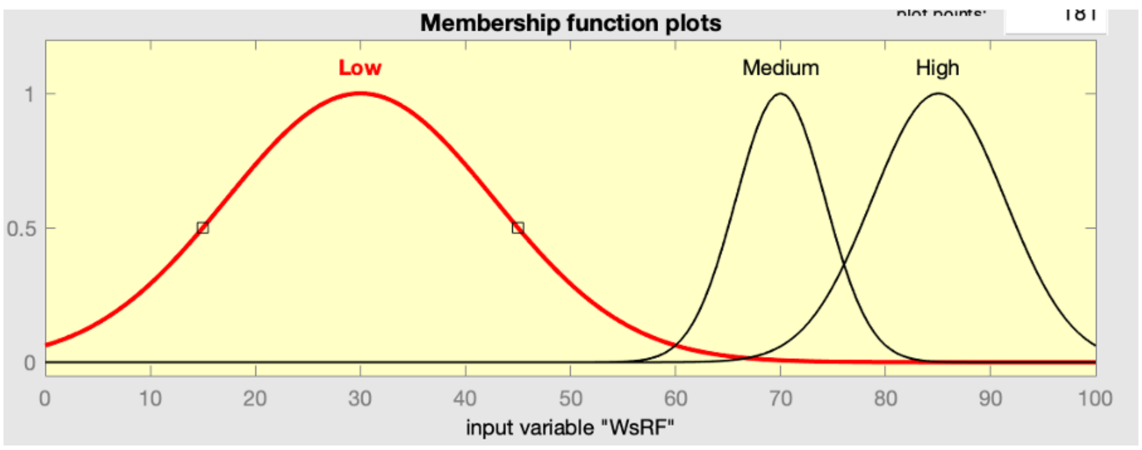


Figure 7.8 : Membership function of WsRF input

7.3.1.3 Step 3: Setup fuzzy rules

We used the Mamdani interface rules on the input parameters based on our experiments. The input parameters are response time, availability, throughput, successability, reliability, and rank for Web Service Quality (WsRF). The rules consist of antecedents which are the input parameters denoted by Y1,Y2,...,Yn and consequences which are the output parameter denoted by Z1, Z2 ,... , Zn. The crisp inputs denoted by Xi will be fuzzified into Yi. The if-then-else Mamdani rule is defined as follows:

$$R(i) : \text{If } X_1 \in Y_1 \text{ and } X_2 \in Y_2 \dots \text{ and } X_n \in Y_n \text{ Then } Z_1 \in Z \tag{7.1}$$

where i is 1,2,3 ...n. n is the total number of rules. Section 7.4.3 explains the fuzzy rules we have defined. (All fuzzy rules are presented in Appendix A)

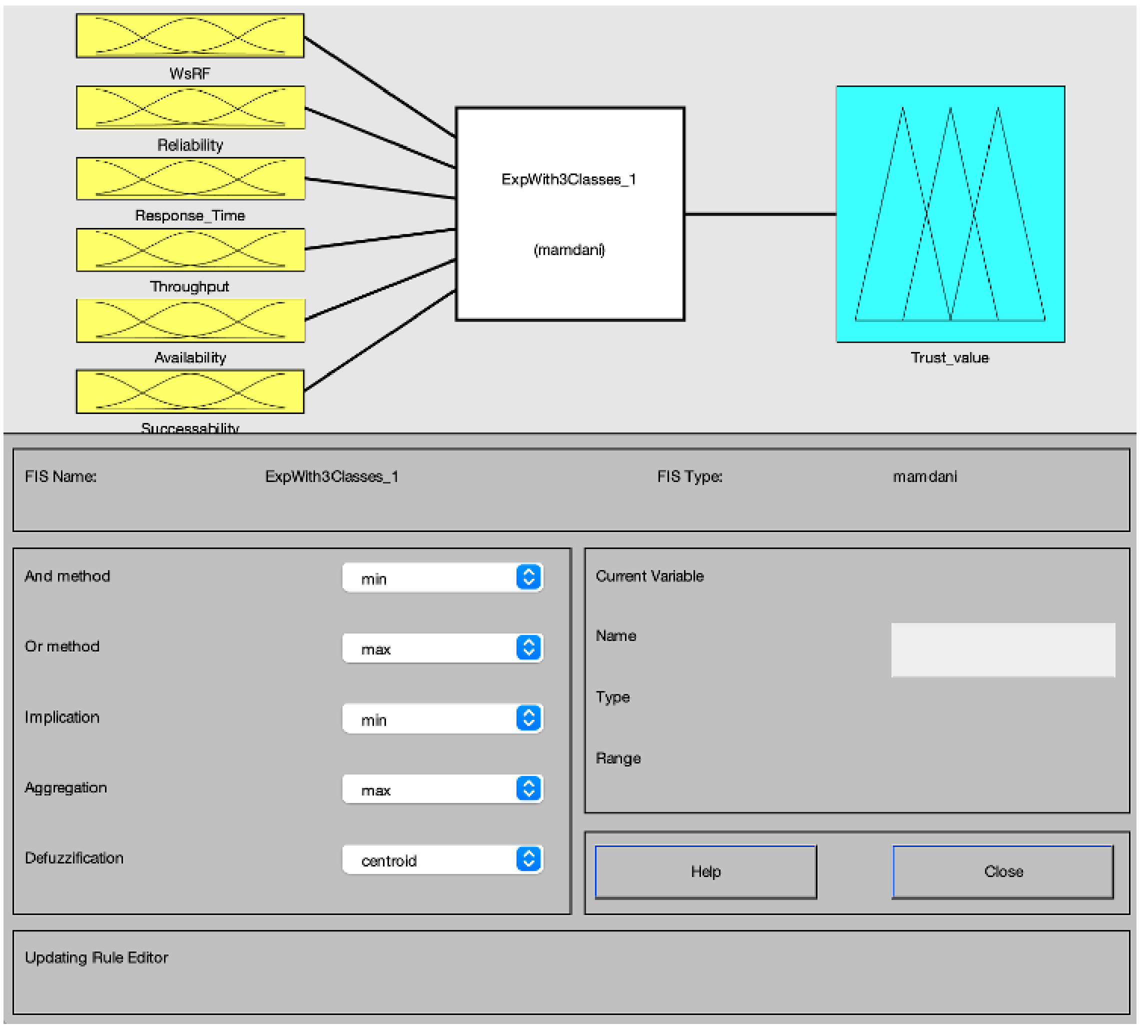
7.3.1.4 Step 4: Compute the trust value

By setting up the fuzzy interface system as shown in Table 7.3 and Figure 7.9, the trust value will be obtained as the fuzzy value.

7.3.1.5 Step 5: De-fuzzify the output

The final step is the defuzzification process where the fuzzy result is mapped to crisp value which is the trust value in the proposed approach. The output of each rule in the fuzzy logic system is represented as a fuzzy set that is calculated based on the membership function and implementation method of the FIS. These multiple fuzzy sets are combined into a single set through the aggregation process in the FIS. Then, the combined fuzzy set is transformed into a final crisp output value using the centroid method of defuzzification [142].

- If the predicted trust value is between (0-1) then the trust value is 1 (highly



FIS Name: ExpWith3Classes_1

FIS Type: mamdani

And method: min

Or method: max

Implication: min

Aggregation: max

Defuzzification: centroid

Updating Rule Editor

Figure 7.9 : Fuzzy interface system with membership functions of inputs and output

Table 7.3 : Fuzzy interface system setup values

Fuzzy Variable Name	**Value**
FIS type	Mamdani
Operator	AND
Membership function	Gaussian
Implication of each fuzzy set	Min
Aggregation operator	Max
Defuzzification method	Centroid

trustworthy).

- If the predicted trust value is between (1-2) then the trust value is 2 (trustworthy).
- If the predicted trust value is between (2-3) then the trust value is 3 (untrustworthy).

The output of the de-fuzzifying process is presented in Figure 7.10.

7.3.2 Implementation of the TEE based on logistic regression

In this section, we illustrate the working of TEE using logistic regression. In this research, logistic regression has been proposed as a prediction model for the trust value of fog nodes. Logistic regression is utilized in the TEE because it is a well-established statistical method for classification tasks, making it suitable for trust prediction in scenarios where the trust value falls into discrete categories (e.g., trustworthy, untrustworthy). It is interpretable, allowing stakeholders to understand the contribution of each QoS factor to trust prediction. Logistic regression also

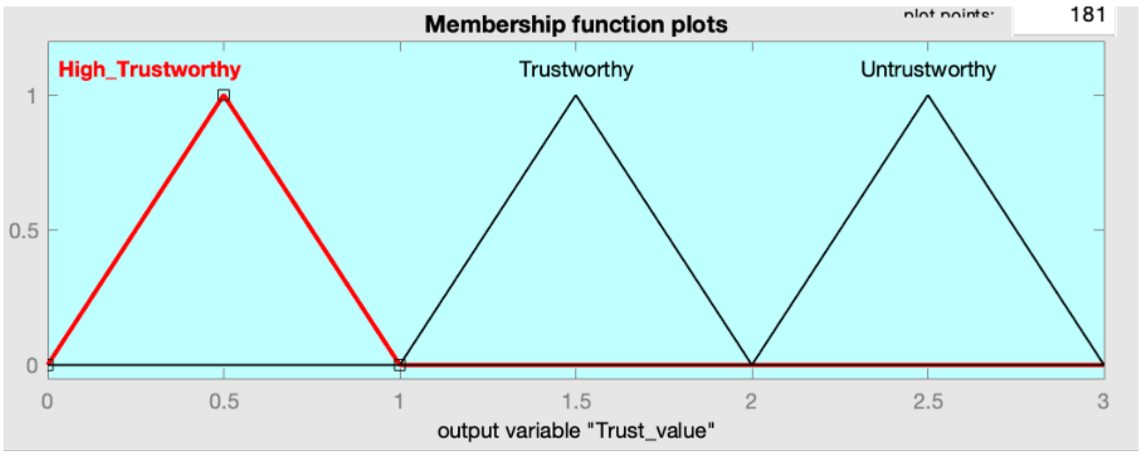


Figure 7.10 : Defuzzification process and membership function of trust value

handles numerical and categorical inputs effectively, accommodating various types of QoS data. The logistic regression can serve as a baseline model for comparison with other AI-based methods and provides insights into the importance of individual QoS factors. In [143], logistic regression has been successfully utilized in IoT for trust prediction and management. This approach provides a promising result compared with another machine learning algorithms. Furthermore, the work in [144] proposed a logistic regression-based trust model that uses logistic regression to predict and manage trust in IoT environments. The model computes the integrated trust value based on direct trust, reputation score, and experience trust, and then uses logistic regression to predict the node's behavior (trusted or malicious). The architecture and working of TEE based on logistic regression is shown in Figure 7.11.

We implemented the logistic regression approach using the scikit-learn library in Python [145]. Scikit-learn is a powerful machine learning library that provides a range of algorithms and tools for data analysis and modeling. The TEE is a multiclass prediction module. We used the one-vs-one technique of logistic regression to enable the effective utilization of logistic regression for multi-class scenarios by transforming them into a set of binary classification tasks. The TEE algorithm

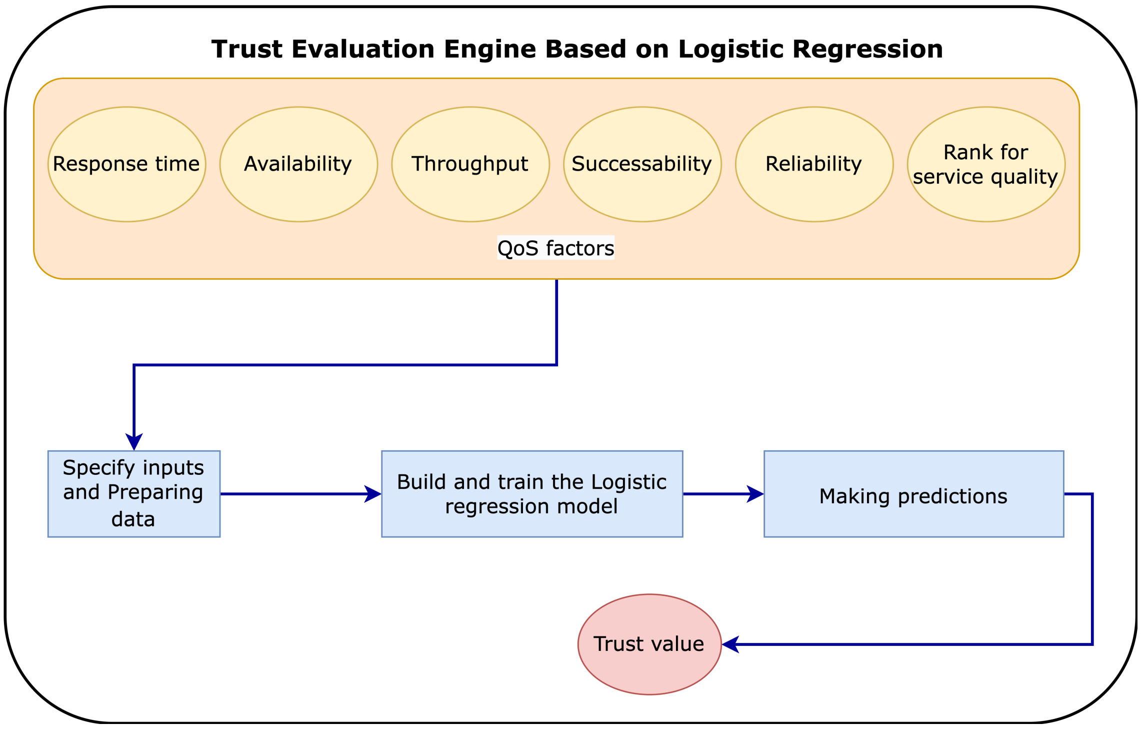

Trust Evaluation Engine Based on Logistic Regression
Response time
Availability
Throughput
Successability
Reliability
Rank for service quality
QoS factors
Specify inputs and Preparing data
Build and train the Logistic regression model
Making predictions
Trust value

Figure 7.11 : Architecture of Trust Evaluation Engine based on logistic regression

includes the following steps to predict the trust value of fog nodes using logistic regression.

7.3.2.1 Step 1: Preparing data

In this step, data preparation was conducted using the QoS web service dataset, known as the QWS dataset [131]. This process involved cleaning the data to remove any inconsistencies and applying feature selection techniques to identify relevant parameters. Furthermore, the dataset was partitioned into training and testing sets to facilitate the evaluation of this prediction model. To enhance the model's generalization capabilities and mitigate overfitting issues, the k-fold cross-validation technique was employed during the training process, ensuring robustness in the model's performance across various data subsets.

7.3.2.2 Step 2: Building and training the logistic regression model

This step involved training the logistic regression model using the provided training data. During training, the model learned to establish a mapping between the input QoS features and the corresponding multi-class trust values (e.g., highly trustworthy, trustworthy, and untrustworthy).

7.3.2.3 Step 3: Testing and make predictions

Once the logistic regression model was trained, it outputed the predicted trust value based on the input QoS parameters.

7.3.3 Implementation of the TEE based on the DNN

In this section, we illustrate the working of the TEE framework based on DNN. In this research, a DNN approach is also proposed for predicting the trust value of fog nodes. The TEE uses this approach, which is based on QoS factors. To achieve this, the multilayer perceptron (MLP) neural network algorithm [146] is used. The MLP NN is a feedforward network which has demonstrated promising results in prediction scenarios in many studies [147] [148]. In [147], the MLP NN resulted in 83% accuracy which is higher than KNN and SVM. The researchers in [148] applied the MLP method to predict the temperature at the end point of an electric arc furnace. The MLP architecture consists of an input layer, one or more layers in the middle, called hidden layers, and an output layer. The working framework based on MLP NN is employed in the TEE in FNSE and is depicted in Figure 7.12.

We implemented different MLP NNs including a different number of layers and neurons to capture the best performance and results. The standard MLP is a cascade of single-layer perceptrons. There is a layer of input nodes, a layer of output nodes, and one or more intermediate layers. The interior layers are sometimes called hidden layers because they are not directly observable from the system's inputs and outputs

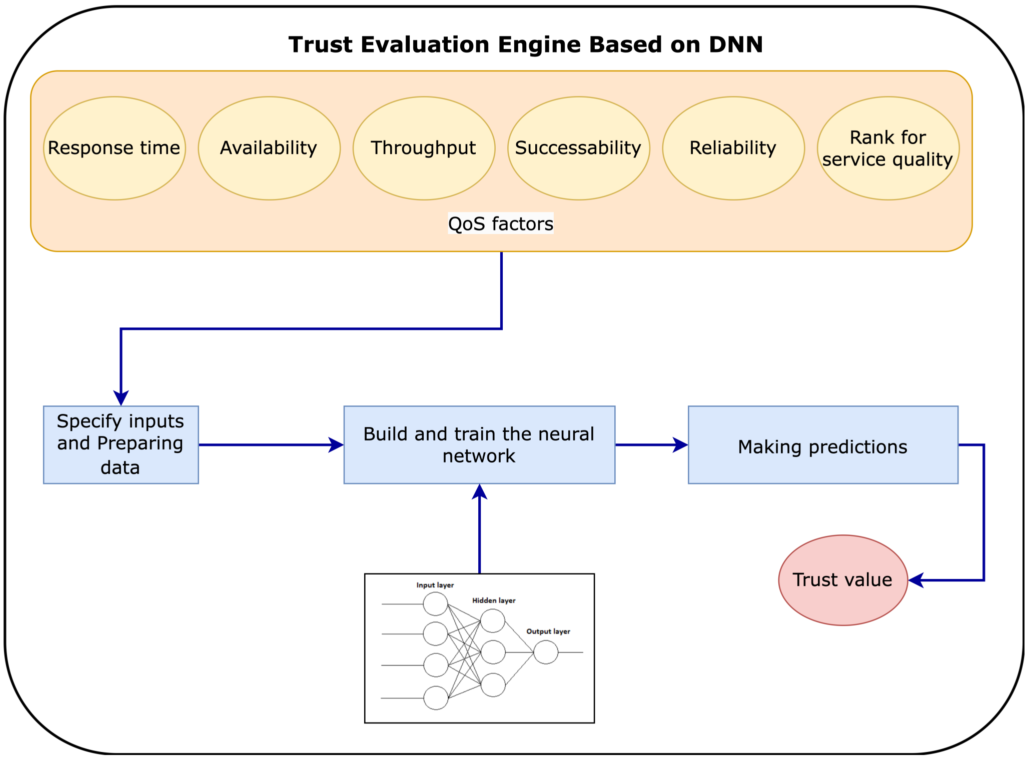


Figure 7.12 : Architecture of Trust Evaluation Engine based on DNN

[149]. Section 7.4.5 discusses in detail the experiments and results of different neural networks. The MLP NN is implemented using Python [150]. The construction of the MLP NN was facilitated by the Keras library [151]. After collecting and storing the QoS for each fog node, FNSE starts the selection process based on the QoS of the fog node by predicting the trust value. TEE starts the prediction process based on MLP NN using the following steps [151]:

7.3.3.1 Step 1: Preparing data

The dataset preparation for trust value prediction consisted of cleaning, normalization, and scaling the QoS factors, along with the appropriate partitioning for training and testing. Data must be numerical and categorical data should be converted to real-value representations. The values should also be scaled in a consistent way, such as from 0 to 1. In Section 7.4, we specified the inputs and used data on fog nodes to predict the trust value. Then, we split the dataset into training and testing sets to evaluate the model's performance accurately. To enhance the model's generalization capabilities and mitigate overfitting issues, the k-fold cross-validation technique was employed during the training process, ensuring robustness in the model's performance across various data subsets.

7.3.3.2 Step 2: Training the network

The data is split into a training set and a testing set and the gradient descent algorithm is used to train the MLP NN. The forward pass processes the input and produces an output, which is compared to the expected output and an error is calculated by the backpropagation algorithm. This process is repeated for all the examples in the training data and one round network update is referred to as an epoch. The network can be trained for a varying number of epochs.

7.3.3.3 Step 3: Testing and making predictions

When the data and the MLP NN model are trained, the MLP NN model can make predictions on trust values. Then, the prediction results are validated against the testing data to evaluate the model. The experiments and results of the TEE mechanism based on MLP NN are discussed in detail in Section 7.4.

7.4 Experiments and results

7.4.1 Dataset

Due to the limited availability of fog node data, we opted to utilize the QWS dataset [131] as a substitute. The QWS dataset is a well-established resource that provides measurements of QoS for real web services. It comprises a comprehensive set of 365 web services, with each service evaluated for various QoS parameters. These parameters include response time, availability, throughput, successability, reliability, and WsRF. Detailed descriptions of these parameters can be found in Table 7.4. In the context of our research, we treated the QoS measurements from the web services in the QWS dataset as representative of the QoS exhibited by fog nodes. While this is an approximation due to the unavailability of a dedicated fog node dataset, it allows us to leverage the wealth of information captured in the QWS dataset to analyze and evaluate the performance of fog computing systems. The trust assessment of fog nodes relies on QoS parameters. The TEE plays a crucial role in predicting the trust value of each fog node, employing either the fuzzy logic system, logistic regression, or DNN. These methods utilize QoS parameters to determine the trust value assigned to fog nodes. Subsequently, the evaluation metrics, including accuracy, recall, precision, and F1 score, are computed for all methods to identify the most effective approach in assessing trustworthiness.

7.4.2 Feature selection

The initial phase of all the experiments is the feature selection process to determine which features are important to ensure the best performance of the selection methods. Furthermore, irrelevant or redundant features may result in overfitting, increase the time and cost of training and decrease accuracy. Hence, to avoid overfitting and computational complexity, we need to determine an optimal number of features and focus on the most informative features during this stage to improve the performance of our predictive models. We applied the feature selection approach using a similarity measure and fuzzy entropy called Luukka [152] which is a statistical method that measures the discriminatory power of each feature in a dataset. It assesses the ability of each feature to distinguish between different classes or categories within the data. During the feature selection process, the Luukka function is applied to the dataset, and the features are ranked based on their entropy values. The higher the Luukka function value for a feature, the more significant it is in dif-

Table 7.4 : QoS parameter description

Parameter name	**Description**
Response Time	Time taken to send a request and receive a response
Availability	Number of successful invocations/total invocations
Throughput	Total number of invocations for a given period of time
Successability	Number of responses / number of request messages
Reliability	Ratio of the number of error messages to total messages
WsRF	Web Service Relevancy Function: a rank for Web Service Quality

ferentiating between the classes. By selecting the features with the highest Luukka function values, we can identify the most informative and discriminative features for our analysis. We used the Luukka function [152] in MATLAB [142] and the features with a low entropy value are removed because the contribution of this feature to the difference between classes is not significant and the most crucial features have the smallest entropy values. We began with six crucial features or parameters, then reduced these to five and ultimately ended up with four parameters. For the first stage of the experiments, we started with six inputs namely response time, availability, throughput, successability, reliability, and WsRF. Then, in the second stage, by applying the Luukka function, the reliability feature is removed, and the five inputs are response time, availability, throughput, successability, and WsRF. The last stage involves four parameters which are response time, availability, throughput, and WsRF as the successability feature is removed.

7.4.3 TEE mechanism based on the fuzzy logic experiments

For the fuzzy logic-based approach, we implemented several experiment scenarios based on the number of input parameters. We started with six important parameters, namely response time, availability, throughput, successability, reliability, and WsRF (1). Then, we ran the fuzzy logic system (2), calculated the performance metrics and execution time (3), and applied the feature selection method (4) which removes one feature. The process was repeated from step 2 until reaching the optimal results. The setup of the experiments with a different number of parameters is shown in Table 7.5. In the first experiment, there are six inputs, namely response time, availability, throughput, successability, reliability, and WsRF. In the second experiment, there are five inputs namely response time, availability, throughput, successability, and WsRF as the reliability feature is removed. In the third experiment, there are four inputs as successability is removed. In the last experiment, the throughput feature is removed. The output is always one of three values, namely highly trustworthy, trustworthy, untrustworthy. The experiment setup was designed and tested using MATLAB software [142]. All experiments were conducted on Mac Pro with 2.8 GHz Quad-Core Intel Core i7 processor and 16 GB 2133 MHz LPDDR3 memory. Refer to Table 7.5 for details of all the experiments. For all experiments, we employed the Mamdani fuzzy interface system and the AND operator. The Gaussian membership function is used for all inputs and the triangular membership function is used for the output. The implication operator of each fuzzy set is Min, and the aggregation operator is MAX. Moreover, the centroid method is used for the defuzzification process. Each experiment has different inputs and rules. In the first experiment scenario, there are 6 input parameters namely response time, availability, throughput, successability, reliability, and WsRF, and there are 231 fuzzy rules, as shown in appendix Table A.1. In the second experiment scenario, there are 5 input parameters namely response time, availability, throughput, successability, and WsRF and there are 47 fuzzy rules, as shown in appendix Table A.2. In the third experiment scenario, there are 4 input parameters namely response time, availability, throughput and WsRF and there are 46 fuzzy rules, as shown in appendix Table A.3. Finally, in the fourth experiment scenario, there are 3 input parameters namely response time, availability and WsRF and there are 18 fuzzy rules, as shown in appendix Table A.4. Then, we compared the results of each experiment and evaluated these results based on four well-known evaluation metrics, namely accuracy, precision, recall and F1 score and compared the performance of our model for each experiment where the execution time of the method was also calculated (further details are provided in Section 7.5).

Table 7.5 : Overview of fuzzy logic experiments

Experiment number	**Number of inputs**	**Number of outputs**	**Number of rules**
Experiment 1	6	3	231 Table A.1
Experiment 2	5	3	47 Table A.2
Experiment 3	4	3	46 Table A.3
Experiment 4	3	3	18 Table A.4

7.4.4 TEE mechanism based on the logistic regression experiments

For the logistic regression-based approach, we implemented several experiment scenarios based on the number of input parameters. We started with six important parameters, namely response time, availability, throughput, successability, reliability, and WsRF (1). Then, we ran the logistic regression model (2), calculated the performance metrics and execution time (3), and applied the feature selection method (4) which removes one feature. The process was repeated from step 2 until reaching the optimal results. In the first experiment, there are six inputs, namely response time, availability, throughput, successability, reliability, and WsRF. In the second experiment, there are five inputs namely response time, availability, throughput, successability, and WsRF as the reliability feature is removed. In the third experiment, there are four inputs as successability is removed. The output is always one of three values, namely highly trustworthy, trustworthy, untrustworthy. The experiment setup was designed and tested using the scikit-learn library in Python [145]. All experiments were conducted on Mac Pro with 2.8 GHz Quad-Core Intel Core i7 processor and 16 GB 2133 MHz LPDDR3 memory.

For the three experiments, the dataset is partitioned into a training set and a testing set, allocating 70% of the data for training and 30% for testing. This division facilitates the model in learning from a substantial portion of the dataset, enabling the evaluation of its performance on unseen instances. The sigmoid activation function is applied to the logistic regression model, ensuring the transformation of calculated weighted sums into probabilities. To handle the multi-class classification in our research, we adopt the ”one-vs-one” technique, which converts the multi-class problem into multiple binary classification tasks. Moreover, the k-fold cross-validation technique is employed to yield a more robust estimate of the logistic regression model's performance. By training and testing on different subsets of the data, this methodology becomes particularly valuable when dataset size is constrained, aiding in the prevention of overfitting.

7.4.5 TEE mechanism based on the DNN experiments

For the DNN-based approach, different neural networks are implemented and tested with a different number of parameters to capture the best performance and the result of the optimal neural network. We conducted extensive experiments on 30 neural networks with a different number of layers and neurons, as detailed in Table 7.6. We conducted these extensive experiments because there is no rule of thumb to find out how many hidden layers are needed. In many cases, one hidden layer works well, however, we conducted several experiments with different set ups of neural networks. We started with a 2-layer MLP which includes the input layer, one hidden layer, and output layer. For example, in the first experiment, the input layer has six nodes, the hidden layer has 30 nodes and the output layer has three nodes. We expressed the nodes of each layer as integer number which separates the layers using a forward-slash character (/). For the first network, the expression should be n/30/3 where n in the input layer is changed in the iteration as the number of parameters.

Table 7.6 : Details of DNN experiments

Experiment number	**Number of layers**	**Number of neurons**
Experiment 1	2	n/30/3
Experiment 2	2	n/100/3
Experiment 3	2	n/500/3
Experiment 4	3	n/30/20/3
Experiment 5	3	n/100/50/3
Experiment 6	3	n/500/250/3
Experiment 7	4	n/30/20/10/3
Experiment 8	4	n/100/50/50/3
Experiment 9	4	n/500/200/200/3
Experiment 10	5	n/30/30/20/10/3
Experiment 11	5	n/100/100/50/50/3
Experiment 12	5	n/500/500/200/200/3
Experiment 13	5	n/500/300/200/100/3
Experiment 14	5	n/1000/500/500/200/3
Experiment 15	5	n/1000/700/500/200/3
Experiment 16	5	n/500/500/500/500/3
Experiment 17	6	n/50/50/30/20/10/3
Experiment 18	6	n/100/100/100/50/50/3
Experiment 19	6	n/500/500/500/200/200/3
Experiment 20	6	n/500/400/300/200/100/3
Experiment 21	6	n/500/500/500/500/500/3
Experiment 22	6	n/1000/800/500/200/100/3
Experiment 23	6	n/1000/1000/500/500/100/3
Experiment 24	7	n/500/500/500/200/200/200/3

Experiment number	Number of layers	Number of neurons
Experiment 25	7	n/500/400/300/300/200/100/3
Experiment 26	7	n/500/500/500/500/500/500/3
Experiment 27	7	n/50/40/30/20/20/10/3
Experiment 28	10	n/50/50/50/30/30/30/20/20/10/3
Experiment 29	10	n/100/100/100/100/100/100/100/100/100/3
Experiment 30	10	n/500/500/500/500/500/200/200/200/200/3

For each experiment, we employed the six, five and four parameters which we represented as three iterations. So, in the first iteration, we conducted the 30 neural networks with six inputs namely response time, availability, throughput, successability, reliability, and WsRF as the starting point. Then, we applied the feature selection method as detailed in Section 7.4.2 and the reliability parameter is excluded. In the second iteration, we applied the 30 neural networks with five inputs namely response time, availability, throughput, successability, and WsRF. After this, again the feature selection method is employed and the successability parameter is removed. So, for the third iteration, the 30 neural networks are applied with four inputs, namely response time, availability, throughput, and WsRF. Table 7.6 provides details of the number of layers and neurons in each network for the 30 experiments. All experiments were conducted on a Mac Pro with 2.8 GHz Quad-Core Intel Core i7 processor and 16 GB 2133 MHz LPDDR3 memory.

The setup of MLP NN is as follows: 70% of the data is used as the training set and 30% as the testing set, the ReLU activation function is used in the hidden layers, the softmax activation function is used in the output layer, and the number of epochs is 150, where neural networks calculate the weighted sums. The calculated sum of weights is passed as input to the activation function in the hidden layers.

The activation function is a function to map the inputs to the desired output. In this research, there are three possible trust values, highly trustworthy, trustworthy and untrustworthy. So, the softmax activation function is suitable for multi-class classification problems.

7.5 Evaluation and discussion

This section presents the evaluation of the fuzzy logic-based approach in Section 7.5.1. The evaluation of the logistic regression approach is presented in Section 7.5.2. The evaluation of the DNN-based approach is presented in Section 7.5.3. Based on the evaluation process explained in Section 4.7.2, we compared the results of the three proposed approaches and the evaluation metrics used for all approaches are accuracy, precision, recall, F1 score and execution time. We compared the best performance results for all experiments for all approaches with six, five and four parameters.

7.5.1 Evaluation of TEE using fuzzy logic

For the fuzzy logic approach, we compared the results of the four experiments. We evaluated these results based on the proposed evaluation process (Section 4.7.2) which includes four evaluation metrics, namely accuracy, precision, recall and F1 score and compared the performance of our model for each experiment and the execution time of the method was also calculated. Figures 7.13 and 7.14 show the results of the evaluation metrics and the execution time of the four experiments. The best evaluation results were obtained in experiment 3 when the inputs are four parameters. The selection of four parameters achieved the highest accuracy (90%) and the lowest execution time among other experiments. Decreasing the number of parameters decreases the training cost and time and improves accuracy significantly (see the results of experiment 3 in Table 7.5).

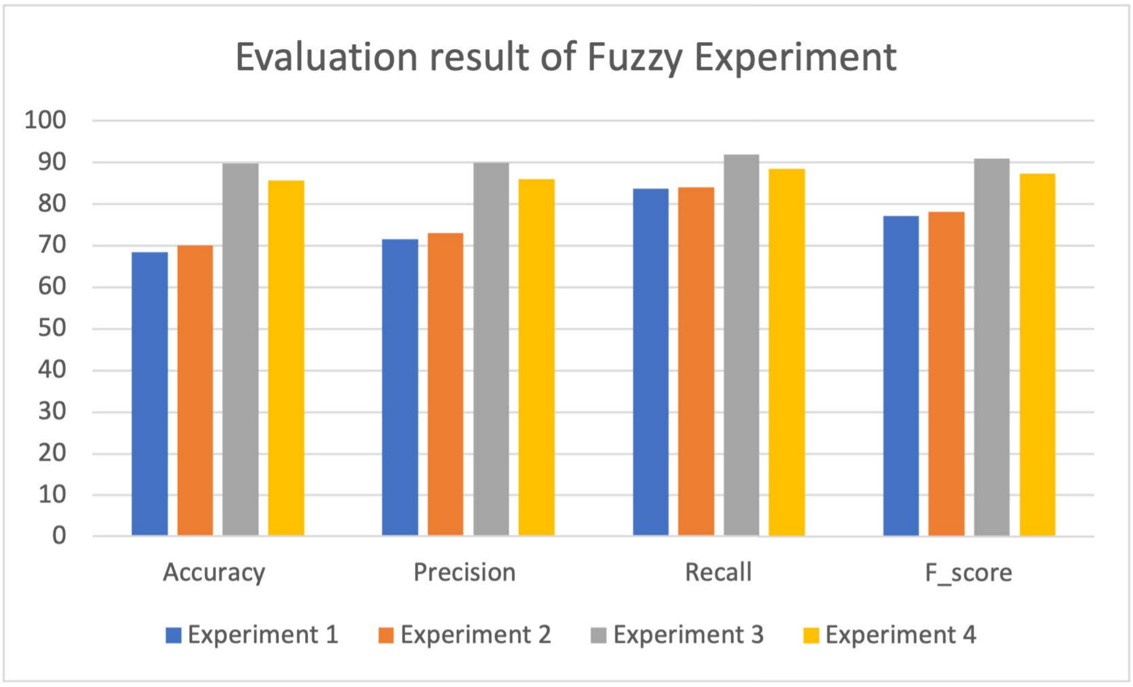


Figure 7.13 : Evaluation results of different fuzzy logic experimental scenarios

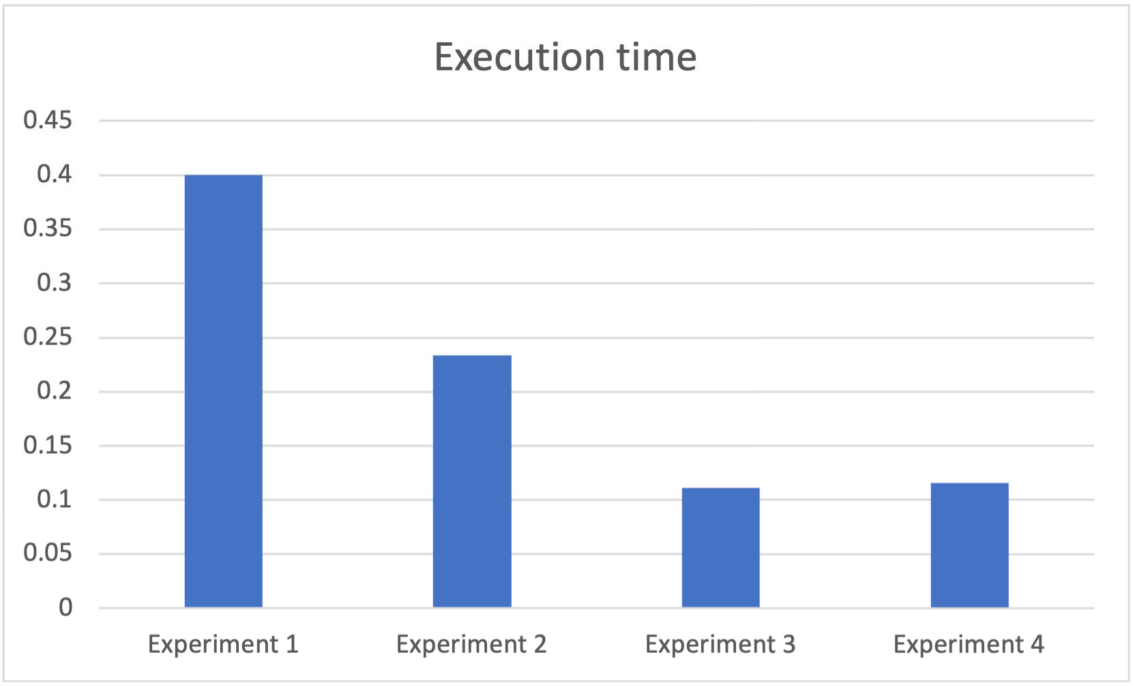


Figure 7.14 : Execution time of different fuzzy logic experimental scenarios

The evaluation results indicate that the fuzzy logic model performs best when using a reduced number of parameters, such as in experiment 3 with 4 parameters. This experiment with 4 parameters achieves higher accuracy, precision, recall, and F1 scores compared to the models with 6 and 5 parameters. Moreover, it is observed that as the number of parameters decreases, the execution time also reduces. Experiment 3 with 4 parameters has the lowest execution time of 0.1116 seconds, closely followed by Experiment 4 with 3 parameters at 0.1155 seconds. In contrast, Experiment 1 with 6 parameters has the longest execution time of 0.4002 seconds.

The fuzzy logic model demonstrates its effectiveness in predicting the trust value of fog nodes based on QoS parameters. Specifically, employing 4 or 3 parameters in the model leads to the highest accuracy, precision, recall, and F1 scores, indicating superior performance. Additionally, the reduced number of parameters results in faster execution times, making the fuzzy logic model both accurate and computationally efficient. Overall, the results highlight the capability of the fuzzy logic approach in handling the complexity of the trust prediction problem and offer valuable insights into fog node reliability and trustworthiness, aiding decision-making processes in fog computing environments.

7.5.2 Evaluation of TEE using logistic regression

For the logistic regression approach, we compared the results of the three experiments. We evaluated these results based on the proposed evaluation process (Section 4.7.2) which includes four evaluation metrics, namely accuracy, precision, recall and F1 score and compared the performance of our model for each experiment and the execution time of the method was also calculated. Figures 7.15 and 7.16 show the results of the evaluation metrics and the execution time of the three experiments.

The best evaluation results were obtained in experiment 3 with the model with four parameters. The selection of four parameters achieved the highest accuracy of

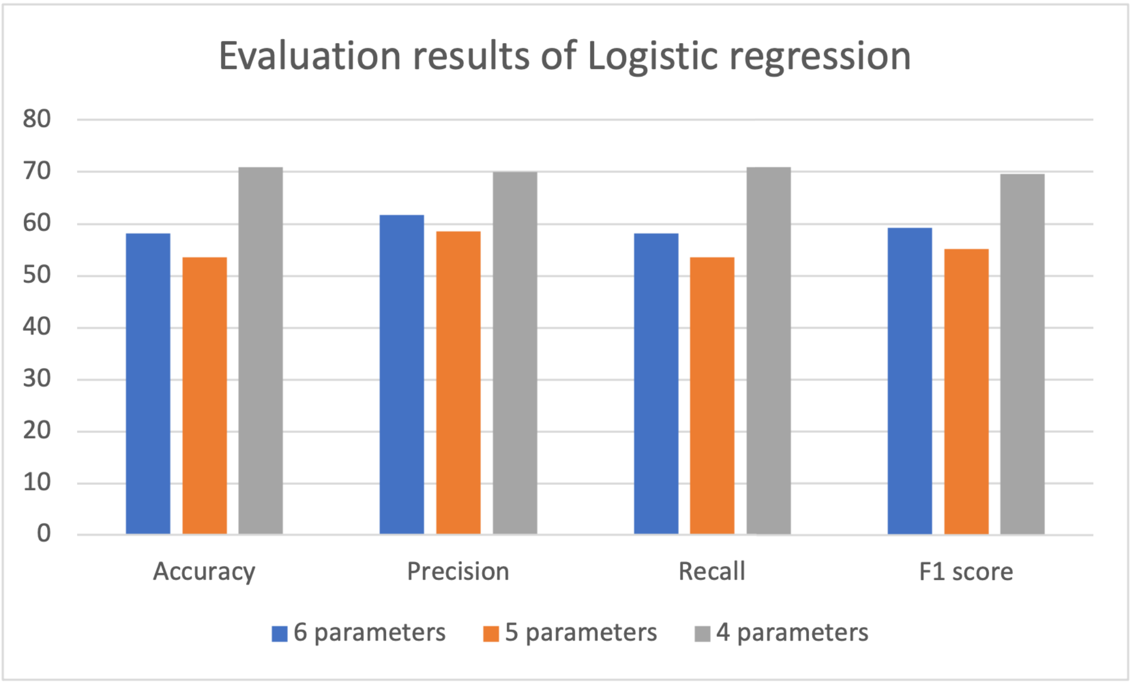


Figure 7.15 : Evaluation result of different logistic regression experimental scenarios

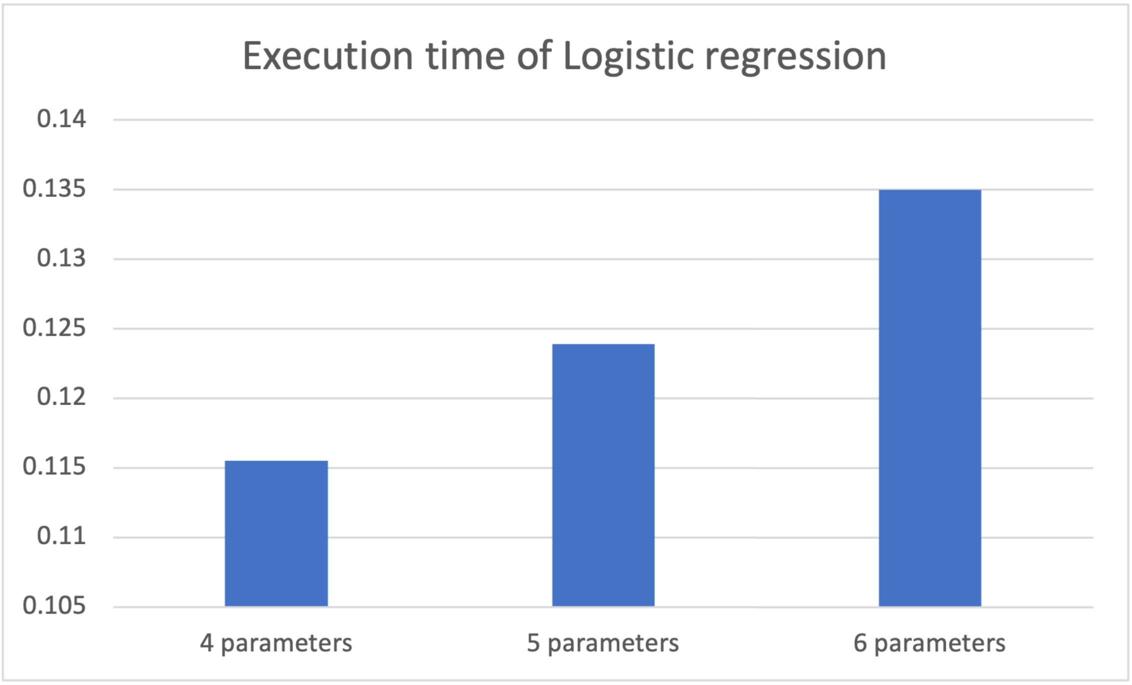


Figure 7.16 : Execution time of different logistic regression experimental scenarios

70%, precision of 69.95%, recall of 70%, and F1 score of 69.63%. Additionality, the execution time of the logistic regression model is provided for each configuration. It shows that the model with 4 parameters has the shortest execution time of 0.11551 seconds, followed by the model with 5 parameters taking 0.1239 seconds, and the model with 6 parameters requiring the longest time of 0.135 seconds.

The results suggest that using a smaller subset of parameters (4 parameters) results in better overall model performance in terms of accuracy, precision, recall, and F1 score compared to using 5 or 6 parameters. However, the difference in performance is relatively small, and the model with 5 parameters could be a good trade-off between performance and simplicity. Moreover, it is observed that reducing the number of parameters also leads to a decrease in execution time, making the model with 4 parameters not only more accurate but also computationally more efficient. In summary, the logistic regression model shows varying performance based on the number of parameters used, and the model with 4 parameters seems to be the most favorable choice in terms of accuracy and computational efficiency.

7.5.3 Evaluation of TEE using DNN

We performed three iterations for the 30 experiments, as shown in Table 7.6. In the first iteration, we evaluated the 30 experiments using six parameters (response time, availability, throughput, successability, reliability, and WsRF) as inputs and three outputs. For the second iteration, the 30 experiments also are evaluated by using five parameters (response time, availability, throughput, successability, and WsRF) as inputs and three outputs. The third iteration is evaluated using four parameters (response time, availability, throughput, and WsRF) as inputs and three outputs. We used the evaluation process which includes four well-known evaluation metrics, namely accuracy, precision, recall and F1 score and compared the performance of our model for each experiment. We ran the model five times (since the results converged after five times) and calculated the average of the evaluation metric results of the 30 experiments with 6, 5, and 4 parameters, as shown in Figures 7.17, 7.18, 7.19, and 7.20.

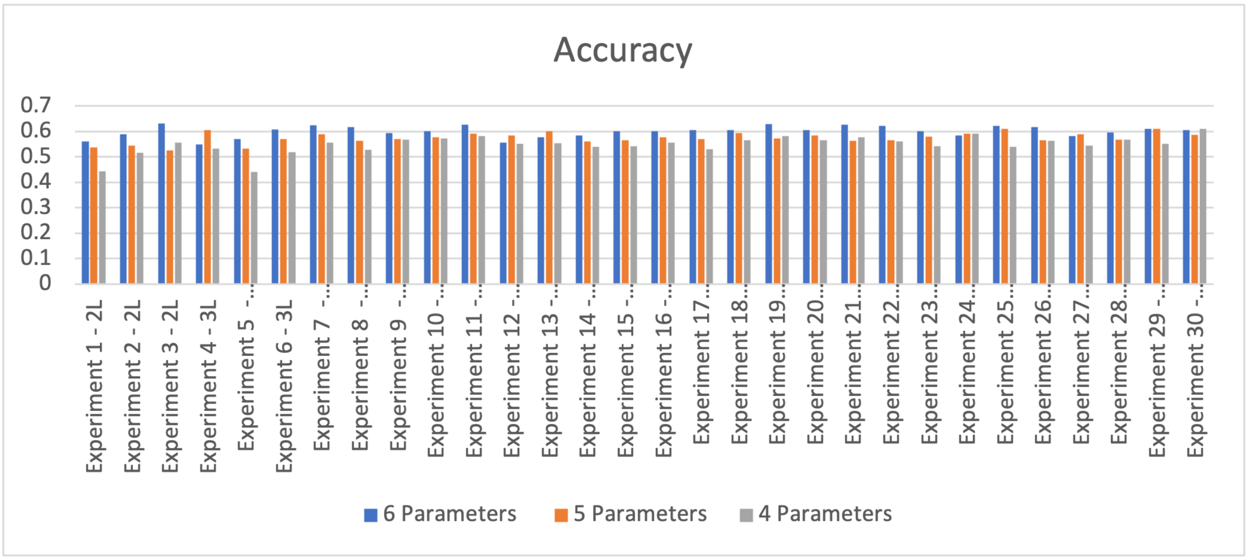


Figure 7.17 : Accuracy of DNN experiments

The results show that when the number of parameters increases, the accuracy of the DNN approach increases. The highest accuracy is obtained when there are six parameters, two layers and the number of neurons of the first and second layer is 6 and 500 respectively. We note that when the number of layers and neurons increases, the accuracy value is not affected, hence the best performance is not achieved. Furthermore, the execution time of each experiment is calculated. The execution time increases when the number of layers and neurons increases. Figure 7.21 shows the execution time of the 7 best performance experiments of DNN with a different number of layers and neurons.

7.5.4 Discussion

The best fuzzy logic performance is achieved when there are four parameters, namely response time, availability, throughput, and WsRF. The optimal setup of

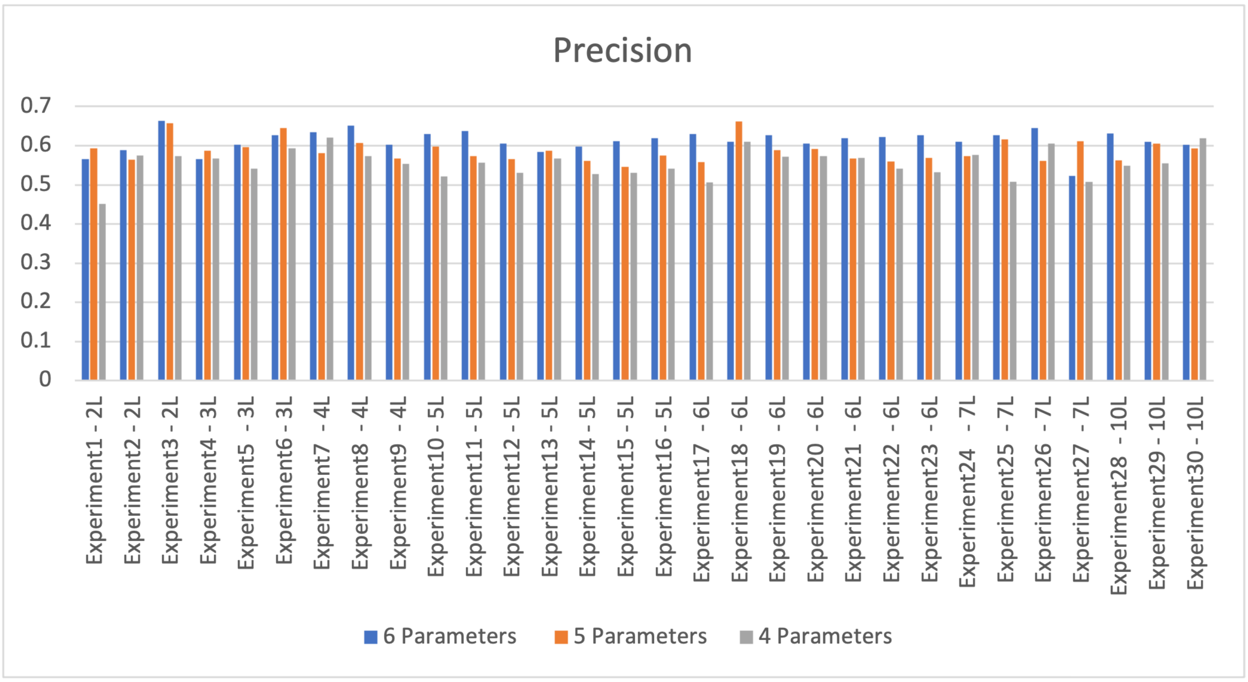


Figure 7.18 : Precision of DNN experiments

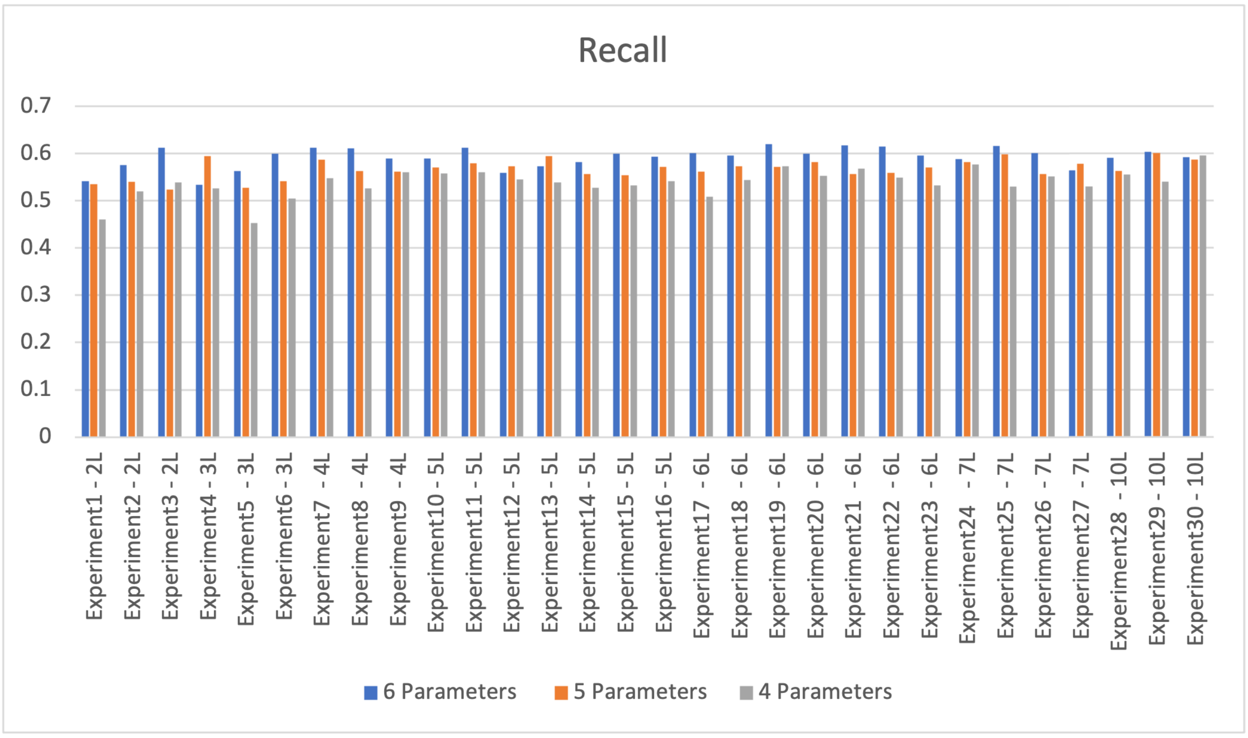


Figure 7.19 : Recall of DNN experiments

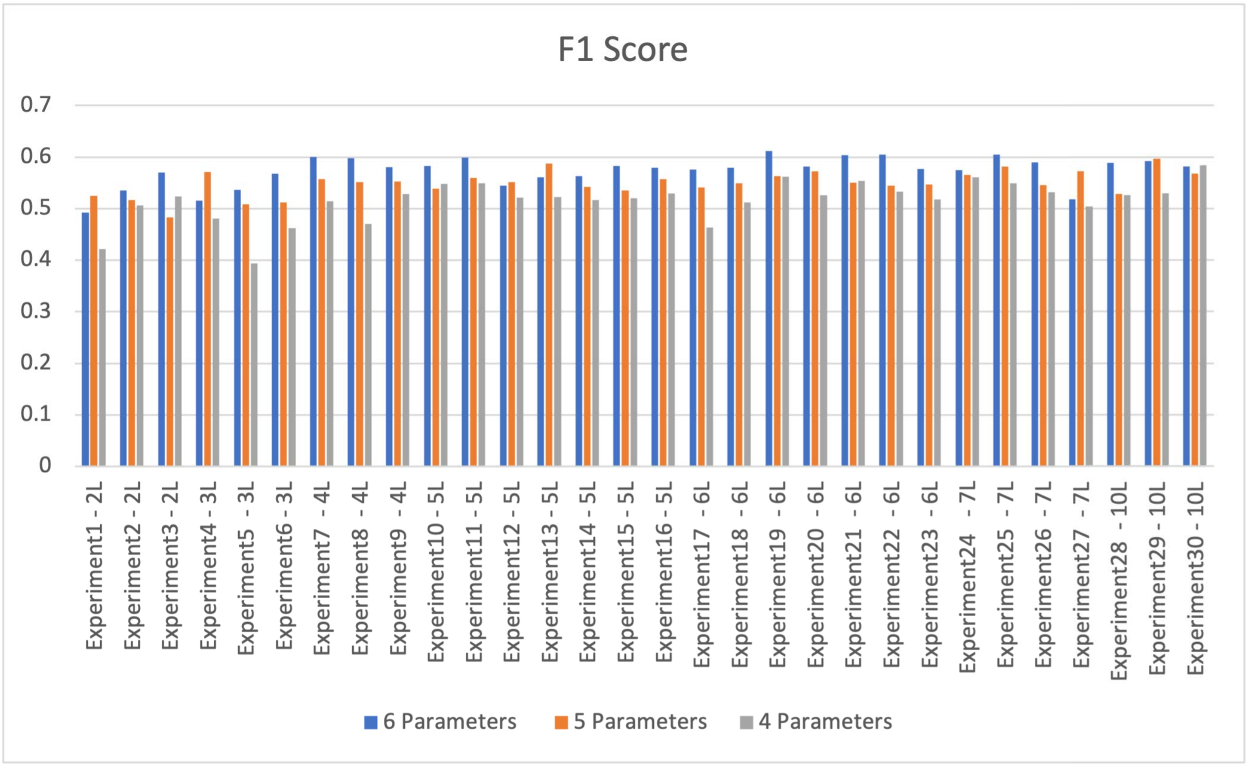


Figure 7.20 : F1 score of DNN experiments

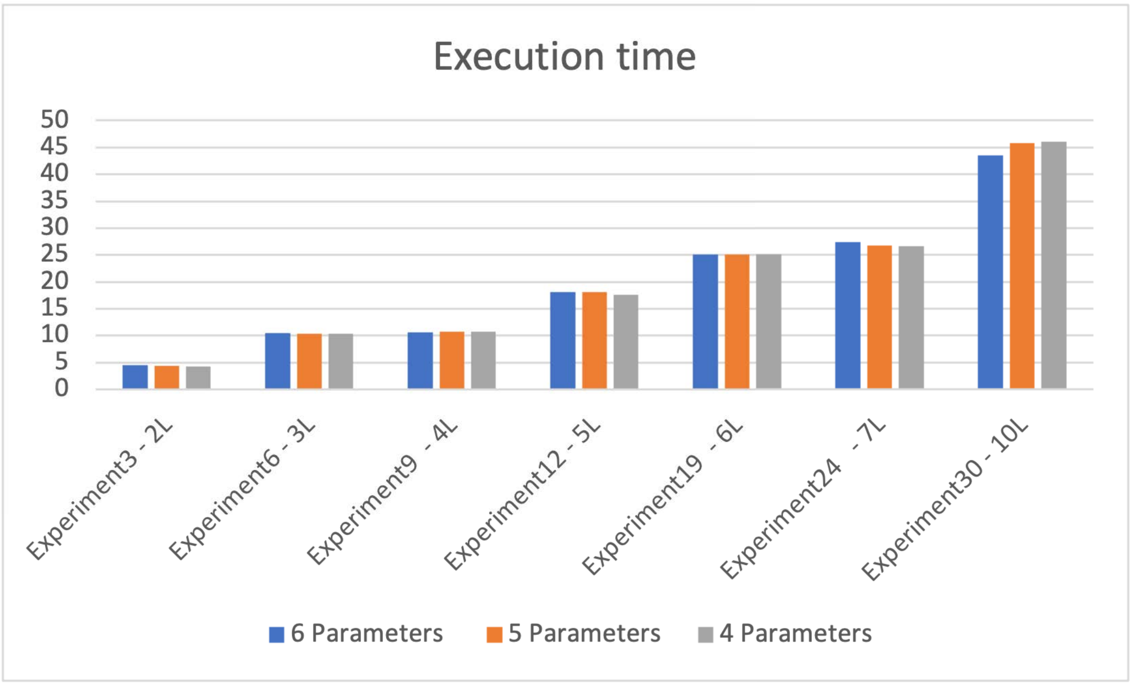


Figure 7.21 : The average execution time of best 7 DNN experiments with a different number of layers and neurons

the fuzzy logic interface when the FIS type is Mamdani, the operator is AND, the membership function is Gaussian, the implication of each fuzzy set is Min, the aggregation operator is Max, the defuzzification method is centroid, and the number of rules is 46 rules. This setup obtains the highest accuracy of 90% and the lowest execution time as shown in Figure 7.22. For the logistic regression model, the optimal performance is achieved when the number of parameters is 4. It obtains an accuracy of 70% as shown in Figure 7.15. However, MLP NN achieves the best performance of 63% accuracy when there are six parameters, namely response time, availability, throughput, successability, reliability, and WsRF. The MLP NN consists of two layers and the number of neurons of the first input layer is 6 and the number of neurons of the second layer is 500 neurons. However, this setup has a high execution time, as shown in Figure 7.23.

By comparing the three methods of TEE, the results show that fuzzy logic achieved the highest performance in terms of accuracy, precision, recall, and F1 score compared to DNN and logistic regression. Figure 7.24 shows the evaluation metrics results of the three approaches with a different number of parameters and Figure 7.25 compares the execution time of the three approaches.

Figure 7.24 shows that the fuzzy logic approach achieved its best performance when four parameters were used, achieving an impressive accuracy of 90% while maintaining the lowest execution time. This indicates that the fuzzy logic approach effectively captures the underlying patterns and relationships in the data, resulting in accurate predictions with efficient computation. The fuzzy logic approach shows its ability to handle uncertain or imprecise data. The high accuracy, precision, recall, and F1 score suggest that the fuzzy logic model is able to handle non-linear and uncertain relationships effectively, resulting in better predictions.

However, the logistic regression approach is a simpler linear model compared

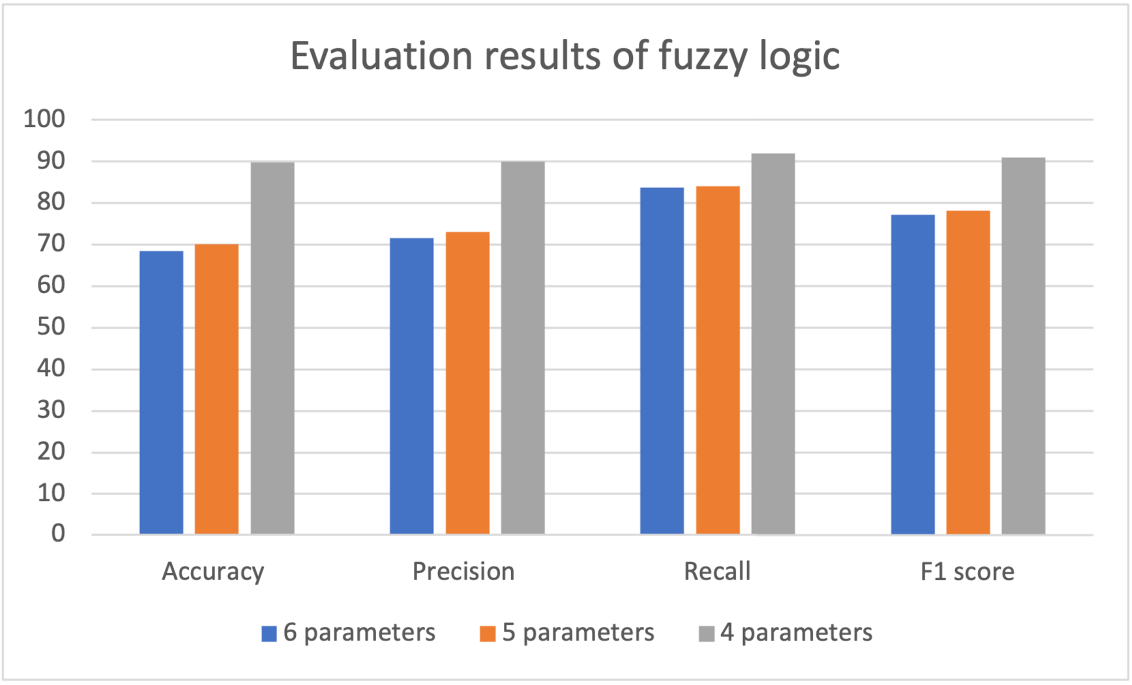


Figure 7.22 : Evaluation results of fuzzy logic with different parameters

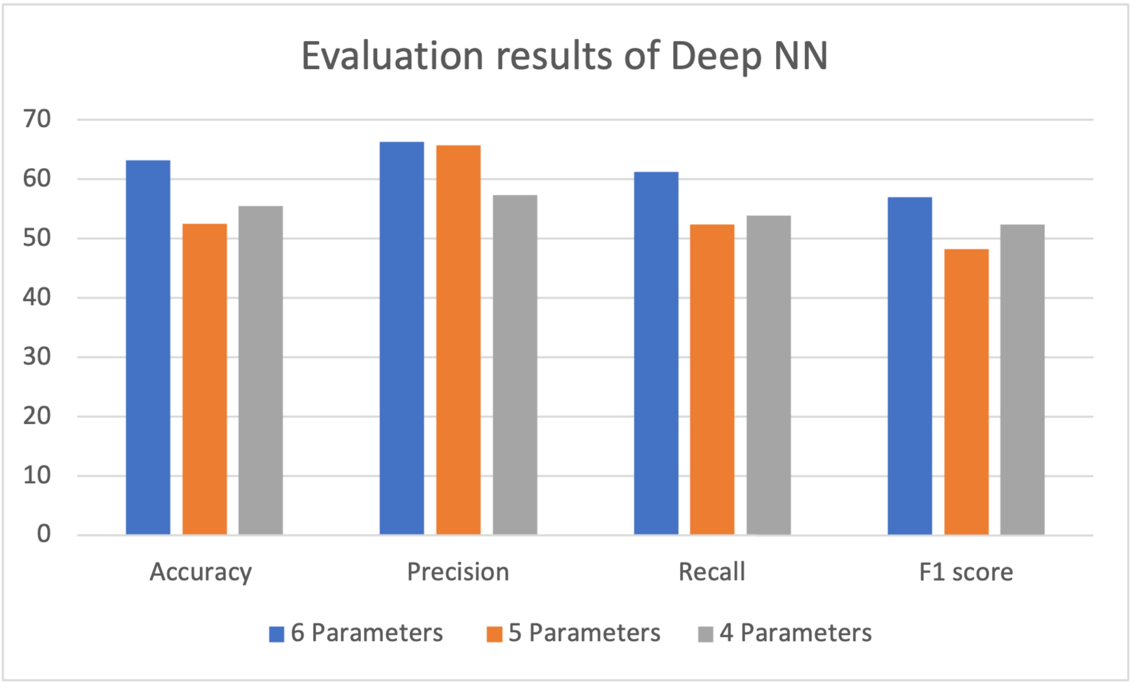


Figure 7.23 : Evaluation results of DNN with different parameters

to fuzzy logic and DNN. Figure 7.15 shows that it performs well in cases where the relationships between features and target variables are linear. The moderate performance of logistic regression might indicate that the relationships between QoS metrics and trust values are not purely linear but still have some linearity, which the model can capture.

On the other hand, the DNN approach achieved its highest performance with six parameters, reaching an accuracy of 63%. However, it should be noted that this performance improvement came at the cost of a higher execution time. This suggests that the DNN approach, with its ability to model complex nonlinear relationships, may require more computational resources and time for training and prediction. The lower performance of the DNN model in this case could be due to insufficient data. DNNs often require a large amount of data for training.

By examining these results, we can gain valuable insights into the strengths and trade-offs of each approach. The fuzzy logic-based approach excels in accuracy and efficiency with a reduced feature set, making it suitable for scenarios that prioritize high accuracy and fast decision-making. On the other hand, the DNN approach demonstrates its capacity to handle more complex data patterns but requires additional computational resources and time. These findings provide valuable guidance for selecting the most suitable approach based on specific requirements and constraints.

7.6 Conclusion

In conclusion, the selection of fog nodes poses a critical challenge in the realm of fog computing. To address this issue, we introduced the TEE framework in this chapter. The TEE framework presents an intelligent and dependable mechanism for predicting the trust value of fog nodes. Within the TEE framework, we proposed three distinct prediction mechanisms: TEE based on fuzzy logic, TEE based on

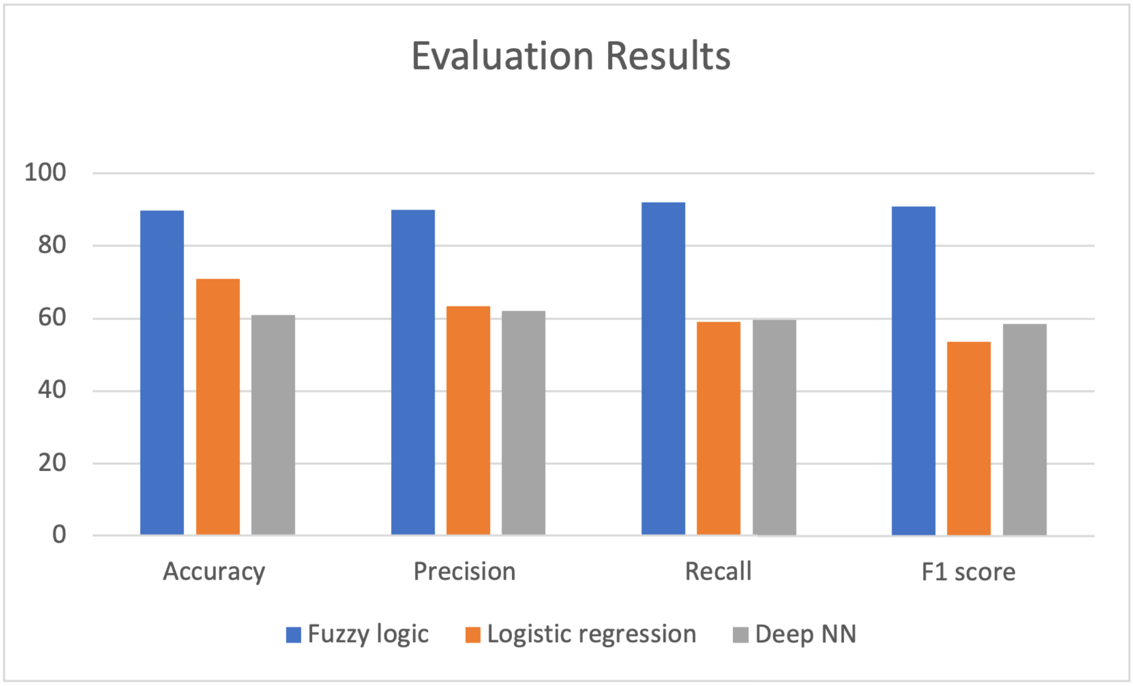


Figure 7.24 : Evaluation results of fuzzy logic, logistic regression and DNN with different parameters

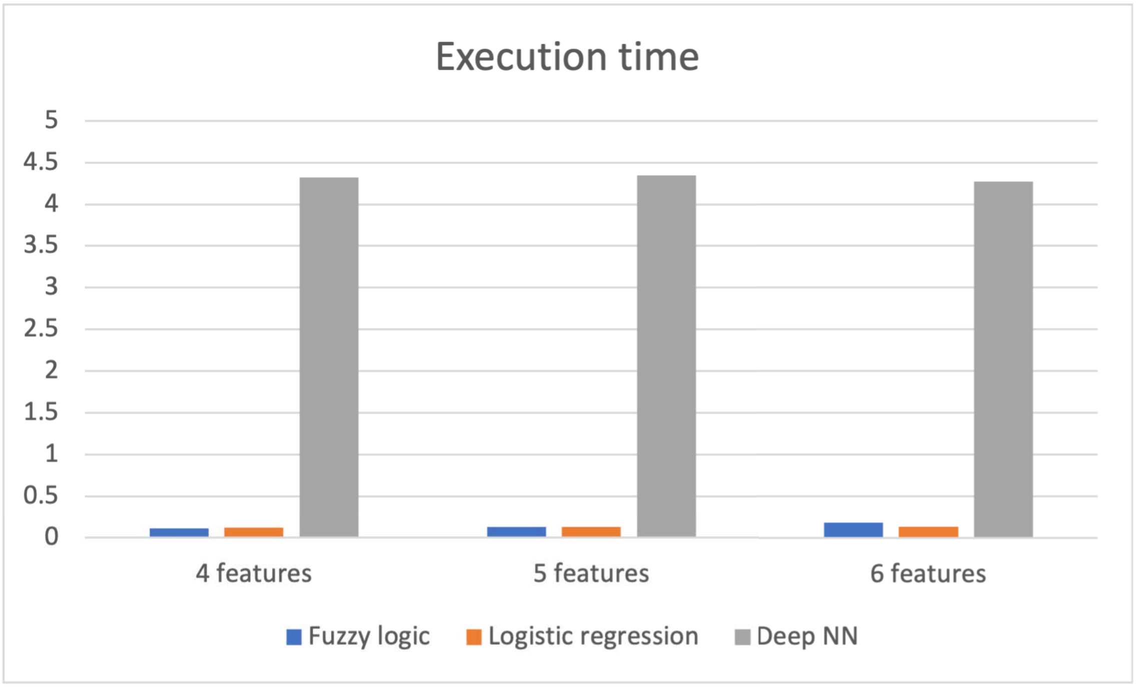


Figure 7.25 : Execution time of fuzzy logic, logistic regression and DNN with different parameters

logistic regression, and TEE based on DNN. These mechanisms aim to predict the trust value of fog nodes to aid in the selection process. Through a series of comprehensive experiments, we evaluated the performance of each approach. The results unequivocally revealed that the TEE based on the fuzzy logic approach exhibits the most promising and the best performance. Notably, it achieved high accuracy of 90%, precision of 90%, recall of 92%, and F1 score of 91%, showcasing its effectiveness in trust prediction. Additionally, the fuzzy logic approach demonstrated efficiency by consuming less time and facilitating rapid predictions. By embracing the TEE mechanism based on fuzzy logic, we can significantly enhance the overall performance of the FNSE. This improvement leads to a more reliable and effective selection process, where fog consumers can confidently choose trustworthy nodes with the shortest latency for optimal service provisioning. The TEE framework marks a crucial advancement in fog computing, paving the way for improved decision-making and enhanced system performance.

Chapter 8

Developing an intelligent approach to bootstrap new fog nodes into a fog ecosystem

8.1 Introduction

A key issue with a number of reputation systems is that they are unable to rank new fog node providers objectively during the process of node selection. This disadvantages new fog node providers as the lack of previous QoS data renders them ineligible for trust-based ranking. This issue is called the cold-start problem as defined in Section 3.2.17. It is an important and challenging issue in the reputation and trust system. The cold-start problem is characterized by the lack of any prior historical information about a new fog node such as QoS data or reputation rates. A new fog node which has recently joined the network has unknown QoS data. New fog nodes need to establish network connectivity and configure their communication protocols during the cold start phase. This configuration process can take time, leading to delays in joining the network and participating in collaborative tasks. Predicting the trust value of a fog node which recently joined the fog ecosystem is fundamental to the development of a trust-based FNS. We propose an intelligent and reliable FNS mechanism which is an intelligent approach to enable fog consumers to select appropriate and reliable fog nodes in a trustworthy manner (as discussed in Chapter 7). Our selection approach is based on the trust value of the fog node based on the values of the QoS data. If the fog node has historical information of QoS data provided to this fog node, the TEE in the FNSE is responsible to carry out the prediction of the trust value. With the trust value of fog nodes, the FNSE will be able to rank the fog node to select the most reliable fog node in the network (Chapter 7).

However, if the QoS values of the fog nodes are unknown, this means the FNSE is unable to make a meaningful selection of fog nodes. To solve the problem of cold-start fog nodes, we proposed BE which is an intelligent trust-based fog node bootstrapping framework. This framework is designed to address the cold-start problem in fog computing environments. It enables fog consumers to make informed and trustworthy decisions when selecting fog nodes for their applications. The cold-start problem refers to the challenge faced when new fog nodes join the network without any prior historical information, such as QoS values or reputation rates. This lack of information makes it difficult to objectively assess the trustworthiness of these nodes during the FNS process. To overcome this challenge, the proposed framework incorporates key components, including the BE which is responsible for the trust evaluation process of new fog nodes to help the FNSE in the selection process. BE works as an intelligent-based trust evaluation agent responsible for predicting and evaluating the trust value of new fog nodes to assist the FNSE to make a reliable fog node selection.

In this chapter, we present the detailed solution for developing an intelligent approach to predict the trust value of new fog nodes with unknown QoS data. This approach helps to bootstrap new fog node providers by predicting the initial QoS values to this fog node recently joining the fog network, then predicting the trust value of the new fog node to make a reliable FNS. We propose BE which is an intelligent framework to predict the trust value of new cold-start fog nodes to help the fog consumer select a reliable fog node based on the QoS of fog nodes in the fog environment.

This chapter is organised as follows: Section 8.2 presents the architecture of the

BE framework. Section 8.3 describes the implementation of BE and the prototype setup. Section 8.4 provides a discussion of the evaluation results of the proposed methods. Finally, Section 8.5 concludes the chapter.

8.2 BE Framework

This section presents the development of an intelligent BE mechanism that predicts the trust value of new fog nodes to bootstrap them within the fog ecosystem. The objective of this research is to enable reliable FNS for fog consumers based on trust values. This selection process is integrated into the FNSE, as described in Chapter 4. When a fog node lacks previous QoS data, the BE takes charge of performing a trust-driven assessment of the new fog nodes and recommending a suitable and reliable fog node for selection. The BE specifically addresses the cold-start phase that new fog nodes experience when joining the network. During this phase, fog nodes have limited or no information about the network and lack the necessary historical QoS attributes. The significance of the cold-start problem increases in dynamic fog environments where nodes frequently join or leave the network. To address this challenge, the BE employs two key modules namely the *QoS prediction module* and the *Reputation prediction module*. Figure 8.1 illustrates the architecture of BE.

The *QoS prediction module* is responsible for bootstrapping the trust value of new fog nodes. It employs a predictive algorithm to estimate the initial QoS values of the newly joined fog nodes. The QoS prediction module begins by clustering all fog nodes in the network based on their contextual attributes, such as geographical location. This clustering helps in grouping fog nodes that are in close proximity to each other. When a new fog node joins the network, it is assigned to the closest cluster based on its location. The QoS prediction module then utilizes the KNN algorithm [113] to identify the nearest neighbors within the closest cluster. Research

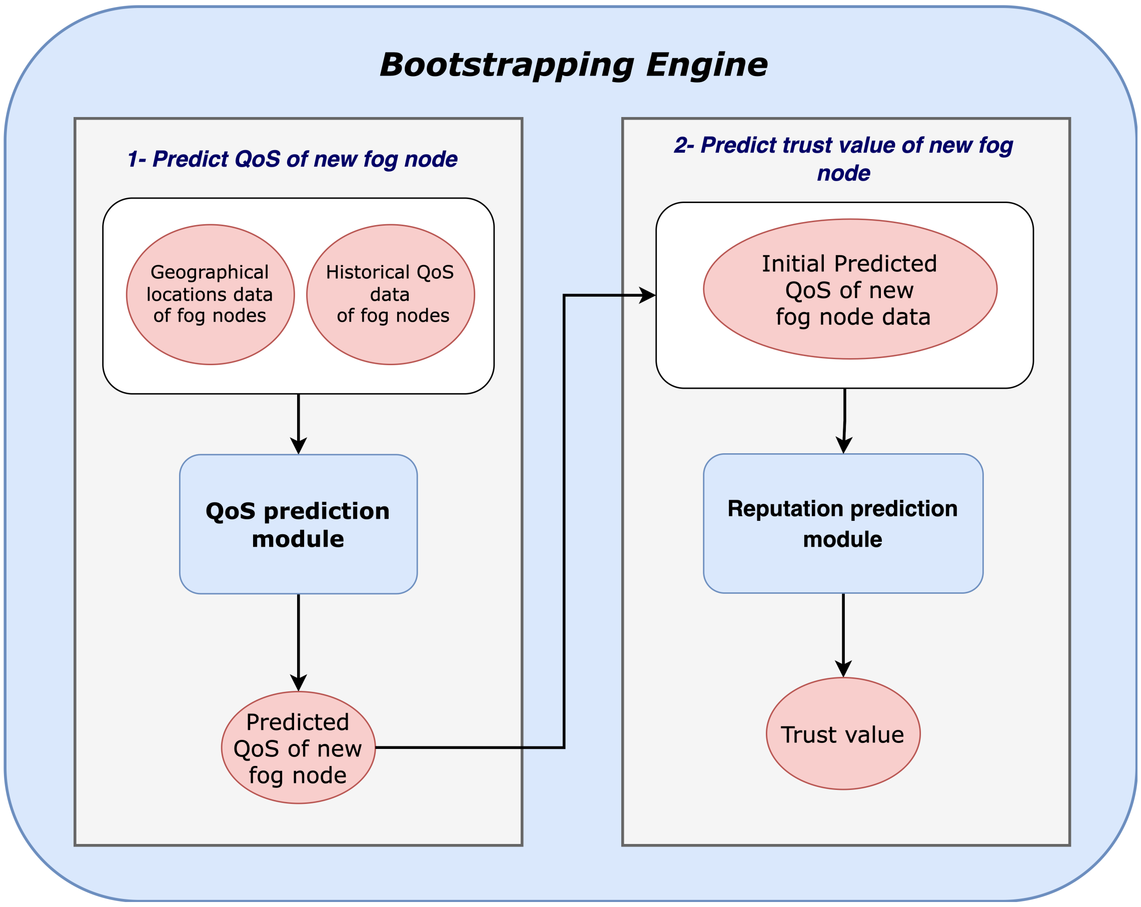

Bootstrapping Engine
1- Predict QoS of new fog node
Geographical locations data of fog nodes
Historical QoS data of fog nodes
QoS prediction module
Predicted QoS of new fog node
2- Predict trust value of new fog node
Initial Predicted QoS of new fog node data
Reputation prediction module
Trust value

Figure 8.1 : Bootstrapping engine model

[153] suggests that web services in the same geographical region are more likely to provide similar QoS performances to the same users. Leveraging this insight, the QoS prediction module predicts the initial QoS value for the new fog node by calculating the average QoS data from its nearest fog nodes, typically considering 10 nearest neighbors (further details are given in Section 8.3.2).

Subsequently, the *reputation prediction module* includes a trust evaluation process that predicts and evaluates the trust value of the new fog nodes. This assessment is crucial for assisting the FNS process and enabling fog consumers to make reliable choices. The reputation prediction module predicts the trust value of the new fog node based on the predicted initial QoS value, employing one of the proposed reputation prediction techniques (further details are given in Section 8.3.3). Once the trust value is predicted, it is assigned to the new fog node along with all other fog nodes in the network. This allows the FNSE to rank all fog nodes based on their trust values, enabling fog consumers to select a fog node that aligns with their trust requirements. This stepwise working of the BE is as follows (refer to Figure 4.6):

Step 1: **Data Collection:** The FNSE collects QoS information for each fog node based on their previous interactions. This step serves as the starting point (Step 1 in Figure 4.6).

Step 2: **New Fog Node:** If a fog node is new and does not have any previous interaction history, the BE proceeds to predict its initial QoS values (Step 2 in Figure 4.6).

Step 3: **QoS Prediction for New Fog Nodes:** The QoS prediction module in BE is responsible for predicting the QoS values of the new cold-start fog node (Step 2.1 in Figure 4.6) which includes:

Step 3.1: **Clustering** - The QoS prediction module applies the K-means algorithm to cluster fog nodes based on their contextual attributes, such as geographical location.

Step 3.2: **Closest Cluster** - When a new fog node joins the network, the QoS prediction module determines the closest cluster based on its geographical location. The KNN algorithm is used to measure similarity and identify similar fog nodes (nearest neighbors).

Step 3.3: **QoS Prediction** - The QoS prediction module predicts the initial QoS value for the new fog node by calculating the average QoS values of its nearest neighbors (neighbors obtained in Step 3.2).

Step 4: **Trust Value Prediction:** The reputation prediction module is responsible for predicting the trust value of the new fog node based on its initial QoS prediction. Various reputation prediction techniques can be employed for this purpose (detailed in Step 2.2 in Figure 4.6).

Step 5: **Fog Node Ranking:** The FNSE ranks the provided fog nodes based on their trust values in ascending order. This ranking ensures that fog nodes with higher trust values are prioritized in the selection process (detailed in Step 3 in Figure 4.6).

Step 6: **Display of Ranked Results:** The final ranked results, including trust values and other relevant information, are presented to the fog consumer. This allows the fog consumer to make an informed decision in selecting the most reliable fog node for their requirements (detailed in Step 4 in Figure 4.6).

All the steps involved in Step 3 will be carried out offline when the new fog node comes to the area. So, when the fog consumer wants to know the trust value of the new fog node, the BE will measure the similarity between the new fog node and all the nearest fog nodes in its cluster. Similar nearest fog nodes should provide similar

QoS features.

8.3 Implementation of BE

In this section, we present the implementation of the BE within the FNSE. The BE module is responsible for addressing the situation when the FNSE encounters a new fog node that lacks QoS data. To tackle this challenge, our proposed approach comprises two steps, as shown in Figure 8.1. First, in Section 8.3.2, the QoS prediction module in the BE focuses on predicting the QoS values for the new fog node. This prediction process involves leveraging intelligent techniques to predict the expected QoS performance of the new cold-start fog node. By analyzing relevant factors such as geographical location data and the QoS data of existing similar fog nodes, the QoS prediction module provides an estimation of the QoS attributes that the new cold-start fog node is likely to exhibit. Once the QoS prediction for the new fog node is determined, the reputation prediction module in BE proceeds to the next step, as outlined in Section 8.3.3. Here, the focus shifts towards predicting the trust value of the new fog node. Trust estimation is crucial in fog computing environments to ensure reliable interactions among fog nodes and the overall network. By considering factors such as the QoS of the fog node, the reputation prediction module assesses the trustworthiness of the new fog node. By systematically following these two steps, the BE module plays a vital role in the FNSE. It enables the system to make informed decisions regarding the inclusion of new fog nodes in the network. By predicting the QoS values and trustworthiness of the new fog node, the system can assess its potential contributions and evaluate whether it aligns with the desired goals and requirements of the fog computing environment.

8.3.1 Dataset

In the fog computing era, one of the major challenges is the lack of available datasets specifically tailored to fog nodes and fog services. Due to the limited availability of such datasets, we had to explore alternative sources of data to tackle the problem at hand. In our research, we addressed the cold start problem of fog nodes by utilizing QoS data obtained from web services. To ensure the reliability and relevance of the data used, we leveraged the QWS dataset [131] and WS-dream dataset [154]. To achieve this objective, we adopted a methodology involving the integration of the QWS and WS-dream datasets, chosen for their respective merits. The QWS dataset provides essential QoS data, which was utilized to achieve objective 3. Conversely, the WS-dream dataset contains valuable geographical information, thereby supplementing our analysis. Specifically, our approach involved combining 2507 web services from the QWS dataset with an additional 5825 web services sourced from the WS-dream dataset. Through this integration process, we identified a subset of 338 web services within the WS-dream dataset that possessed explicit geographical location information. The QWS dataset [131] comprises QoS data which includes response time and throughput but the WS-dream dataset [154] provides essential location information such as IP addresses, country, continent, region, city, latitude, and longitude details associated with each web service. While the QWS and WS-dream datasets primarily focus on web services, we repurposed it to fit our research requirements in the context of fog computing. Before integrating the datasets into our research, we performed a meticulous data cleaning process to enhance the dataset's integrity and consistency. This involved removing any null or missing values to ensure the dataset's reliability and eliminate potential biases. Subsequently, we combined the location information, including latitude and longitude, with the corresponding QoS data for each web service. This fusion of location information and QoS metrics allowed us to construct a comprehensive dataset that we considered representative of fog node data. For our research on addressing the cold start problem of fog nodes, we specifically focused on two key parameters: geographical location information and QoS metrics. The geographical location information, consisting of latitude and longitude, enabled us to examine the spatial aspects and proximity of fog nodes. Additionally, the QoS parameters, including response time, availability and throughput of services in fog nodes, provided crucial insights into the performance and capabilities of fog nodes as described in Table 8.1. By leveraging the WS-dream dataset, incorporating location information, and considering essential QoS parameters, our research aims to develop an effective intelligent approach to mitigate the challenge posed by the cold start problem in fog computing. While the lack of fog node specific datasets posed a significant hurdle, our approach utilizing web service data allowed us to explore the intricacies of fog nodes and propose innovative solutions in real-world scenarios.

Table 8.1 : A snapshot of the dataset

Fog node ID	**Latitude**	**Longitude**	**Response Time**	**Throughput**
1	44.98	-93.2638	302.75	7.1
97	44.98	-93.2638	621.2	8.3
87	41.8392	-88.3612	82.25	22.7
133	41.8392	-88.3612	87.9	14.7
34	39.0061	-94.6337	786.5	4.8

8.3.2 QoS prediction module

Our objective in this phase is to make predictions regarding the unknown QoS of a recently joined fog node in the network. This process entails two main steps:

clustering the fog nodes based on the similarity of their geographical locations and subsequently predicting the QoS by leveraging the nearest neighbors of the new fog node. To cluster the fog nodes according to their geographical location similarity, we utilize the K-means clustering algorithm [155]. This algorithm facilitates the grouping of fog nodes that possess similar geographical coordinates into different clusters. By considering the latitude and longitude information associated with each fog node, we can effectively cluster them based on their spatial proximity. Through this clustering process, we establish groups that exhibit geographical similarities among the fog nodes. Following the clustering step, we employ the KNN algorithm [113] to identify the fog nodes that are closest in geographical proximity to the new fog node. The KNN algorithm functions by computing the proximity between fog nodes based on their geographical information. By determining the K-nearest neighbors of the new fog node, we can leverage the known QoS values associated with these fog nodes to predict the QoS value for the new fog node. By utilizing both the K-means clustering algorithm and the KNN algorithm, we establish a robust framework for predicting the QoS value of the new fog node based on its geographical proximity to the existing fog nodes. This prediction process effectively addresses the cold start problem by providing an estimate of the QoS even before the new fog node has accumulated sufficient data or established connections with other nodes in the network. Figure 8.2 illustrates the working of the QoS prediction module for a new fog node within BE.

8.3.2.1 Clustering fog nodes

Upon the new fog node joining the network, it becomes crucial to identify the nearest fog nodes and determine the appropriate cluster within the new fog node's geographical region. The study in [153] demonstrates that web services, denoted s and t, are more likely to exhibit similar QoS performances for the same users if

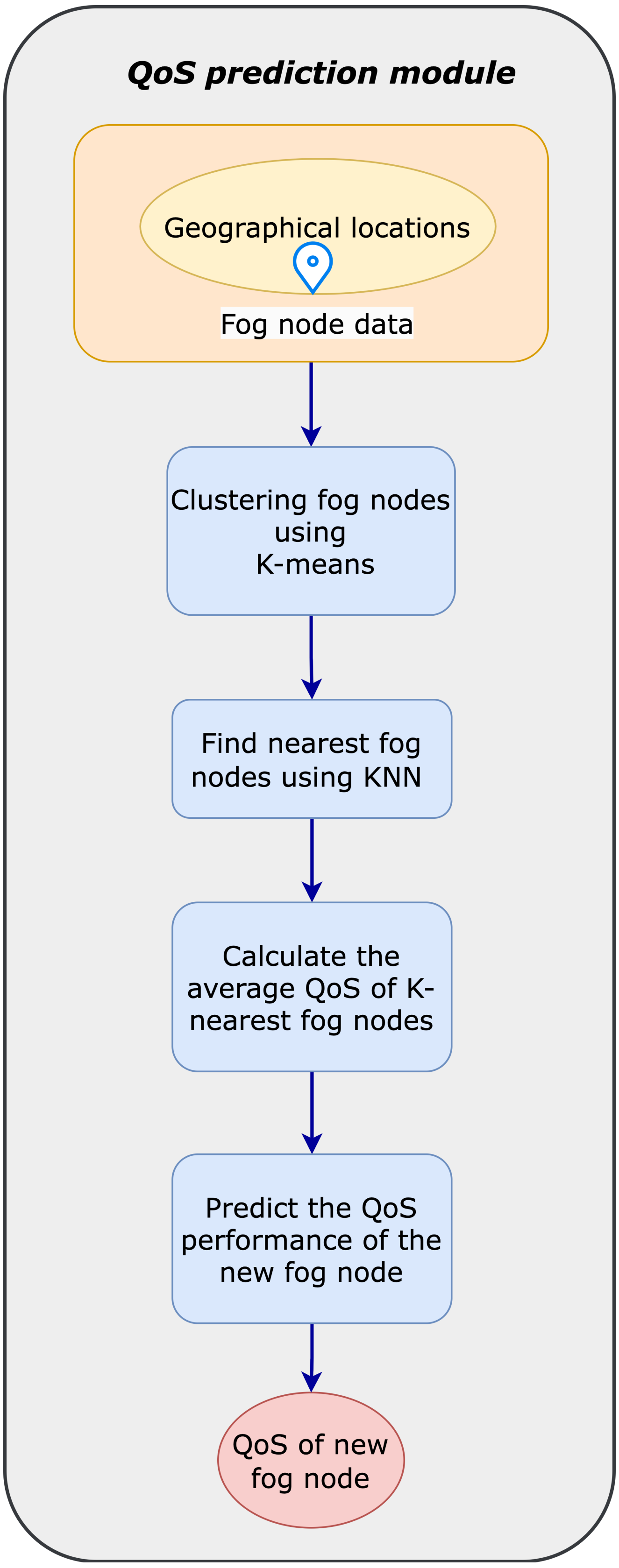

QoS prediction module
Geographical locations
Fog node data
Clustering fog nodes using K-means
Find nearest fog nodes using KNN
Calculate the average QoS of K-nearest fog nodes
Predict the QoS performance of the new fog node
QoS of new fog node

Figure 8.2 : Working of the QoS prediction module

they are situated within the same geographical region. Therefore, clustering fog nodes based on their geographical locations and identifying the closest fog nodes to the new node can aid in achieving similar QoS performance for the cold-start fog node. In our research, we utilized the K-means clustering algorithm implemented in MATLAB [142]. We implemented different clustering scenarios with a different number of K from 2 to 6. The next step involves determining the optimal value of K for clustering. Two clustering evaluation techniques are employed to evaluate the optimal number of clusters such as Davies-Bouldin criterion clustering evaluation [156] and the silhouette evaluation [157].

The Davies-Bouldin criterion is based on a ratio of within-cluster and between-cluster distances. The optimal clustering solution has the smallest Davies-Bouldin index value. The silhouette value measures the similarity of each fog node within a cluster and compares it with other clusters. Silhouette values range from -1 to 1, where a higher value indicates greater consistency among fog nodes within a cluster, while a lower value suggests poor alignment.

Table 8.2 : Clustering evaluation comparison with a different value of K

Number of clusters	**K=2**	**K=3**	**K=4**	**K=5**	**K=6**
Davies-Bouldin evaluation	0.4708	0.3080	0.4586	0.4267	0.4419
Silhouette evaluation	0.8372	0.9183	0.8087	0.8567	0.8571

Based on the evaluation presented in Table 8.2 and the analysis shown in Figures 8.3 and 8.4, we identify that the most suitable number of clusters, denoted as K, for our specific scenario is 3. Figure 8.3 clearly illustrates that the lowest Davies-Bouldin value is achieved when utilizing three clusters, indicating that this number of clusters is optimal. Furthermore, Figure 8.4 provides additional evidence as it demonstrates that the highest silhouette value is attained with three clusters, reinforcing the conclusion that the optimal number of clusters is 3.

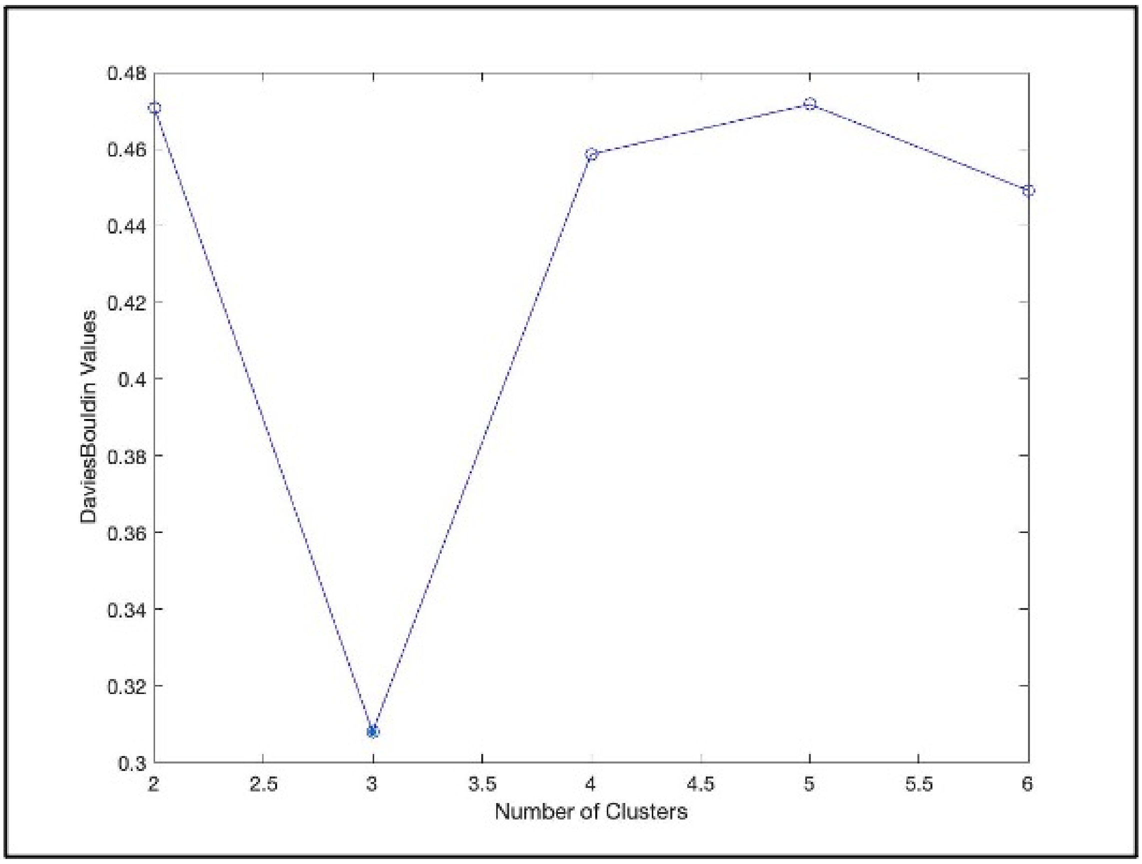


Figure 8.3 : Davies-Bouldin clustering evaluation

Subsequently, we implemented the k-means clustering algorithm in MATLAB, utilizing the squared Euclidean distance metric [131] and the fog nodes were grouped into 3 clusters based on their geographical locations. The geographical distribution of the fog nodes, segmented into 3 clusters, is depicted in Figure 8.5. This clustering approach facilitates the organization and grouping of fog nodes according to their proximity, thereby aiding the cold-start fog node in achieving similar QoS

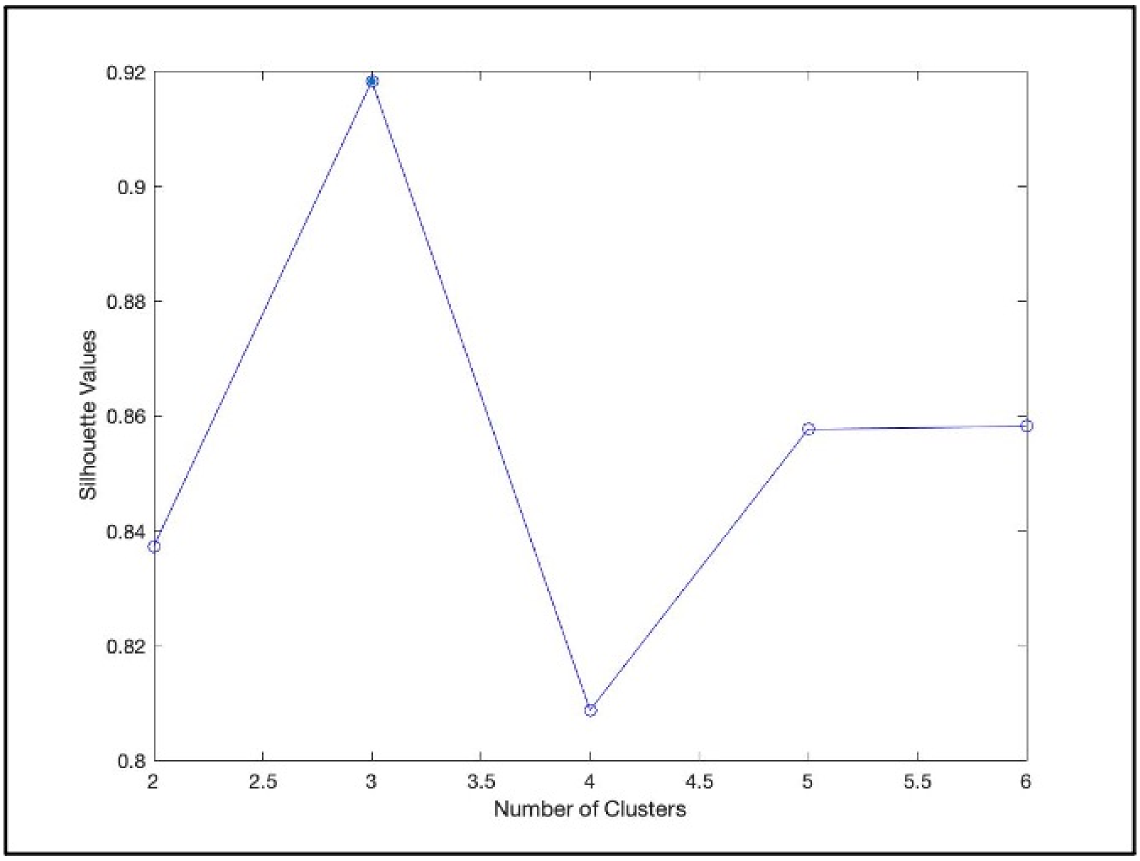


Figure 8.4 : Silhouette clustering evaluation

performance to the fog nodes in its cluster. By leveraging geographical clustering and identifying the nearest fog nodes, we enable the cold-start fog node to align its performance with other fog nodes in its geographical region. This enhances the QoS provision and ensures a more seamless integration of the new fog node into the network.

8.3.2.2 Nearest Neighbor Identification and QoS Prediction

In this research, predicting the QoS value of the new fog node is based on the nearest neighbors to the new fog node. To address the issue of a new fog node experiencing a cold start issue, we devised a prediction-based method to estimate the QoS values which includes response time and throughput for the recently initialized fog node. This enables us to estimate the node's performance even before it has

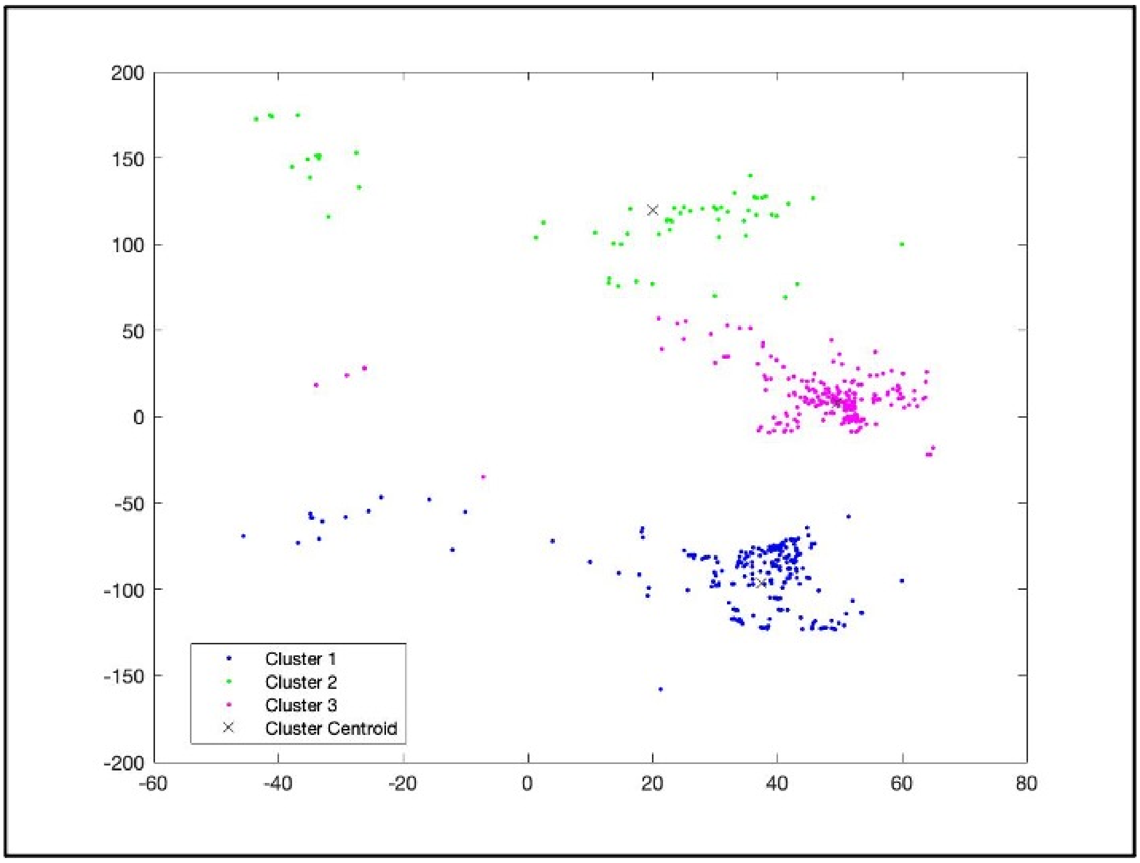


Figure 8.5 : Result of clustering the fog nodes

accumulated enough data or established connections with other fog nodes and fog consumers within the network. Our approach revolves around utilizing the KNN algorithm [113] to analyze the closest neighbors. When a new fog node joins the network, we take into consideration the geographical locations of the nearest fog nodes in its region or cluster. These neighboring fog nodes act as reference points for predicting the QoS of the new node. By leveraging the KNN algorithm, we identify the nearest fog nodes based on factors such as geographical locations. By incorporating the average QoS values of these nearest fog nodes, we can predict the QoS performance of the new fog node that is experiencing a cold start. The KNN algorithm is implemented by the scikit-learn library APIs in Python [128]. Scikit-learn is a powerful machine learning library that provides a range of algorithms and tools for data analysis and modeling. In our implementation, we specifically employ the Euclidean distance metric as the distance measure for the KNN algorithm. The Euclidean distance is a commonly used distance metric in machine learning, which calculates the straight-line distance between two data points in a multi-dimensional feature space. By using the Euclidean distance metric, we can effectively determine the similarity between the new fog node and its nearest neighbors based on their geographical locations. During the prediction phase, the KNN algorithm identifies the k-nearest neighbors of the new fog node based on their Euclidean distances. We specified the value of k with 10. The KNN model then considers the average QoS values of these k-nearest neighbors and predicts the QoS performance of the new fog node based on this information.

8.3.3 Reputation prediction module

Once the fog nodes have been clustered and the initial QoS values have been determined for the new cold-start fog node, the BE model takes over the task of intelligently predicting the trust value of the new cold-start fog node. This prediction is crucial in assessing the reliability and credibility of the fog node before integrating it into the network. To achieve this, the BE model utilizes the reputation prediction module, which incorporates various techniques and algorithms based on the QoS data of all fog nodes. The reputation prediction module serves as a means to estimate the trust value for the new cold-start fog node, which lacks any previous interactions or known providers. Within the reputation prediction module, we propose three intelligent methods aimed at achieving the best performance and prediction results include fuzzy logic-based reputation prediction, regression-based reputation prediction, and deep learning-based reputation prediction, as shown in Figure 8.6.

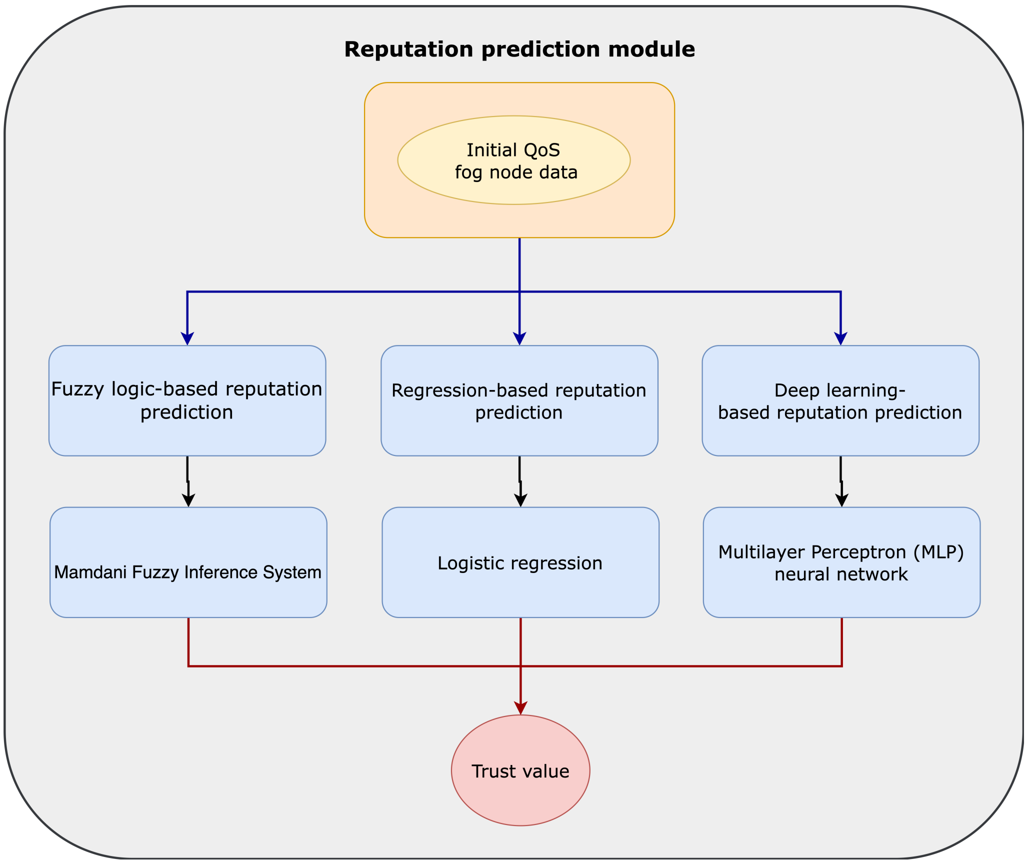

Reputation prediction module
Initial QoS fog node data
Fuzzy logic-based reputation prediction
Regression-based reputation prediction
Deep learning-based reputation prediction
Mamdani Fuzzy Inference System
Logistic regression
Multilayer Perceptron (MLP) neural network
Trust value

Figure 8.6 : Working of the Reputation Prediction module

8.3.3.1 *Fuzzy Logic-Based Reputation Prediction*

Using fuzzy logic to predict the trust value of fog nodes based on QoS factors such as response time and throughput involves developing a fuzzy inference system that captures the uncertainty and imprecision in the data to make informed trust predictions. Fuzzy logic provides a flexible and interpretable framework to model complex relationships between input variables and output trust values, which are linguistic terms in this case (e.g., highly trustworthy, trustworthy, untrustworthy). We implemented fuzzy logic in BE with Python using the scikit-fuzzy library [158]. Scikit-fuzzy provides functions to define membership functions, create fuzzy rules, perform fuzzy inference, and defuzzify the output.

The following steps are followed to apply the fuzzy logic prediction in the BE:

Step 1: Fuzzification: The first step is to convert the crisp value of inputs to fuzzy set by identifying the linguistic terms of each input. For each QoS factor (response time and throughput), we identify three linguistic terms, namely 'low', 'medium', or 'high'. For each QoS factor, we need to define fuzzy membership functions that map the numerical input values to these linguistic terms (high, medium, low). These membership functions describe how each input value belongs to each linguistic term, as shown in Figures 8.7 and 8.8. The triangular membership function is used for this task. The specific range for each of these variables is listed in Table 8.3.

Table 8.3 : Parametric range for the linguistic variables of the input parameters

Input Parameter	**Low**	**Medium**	**High**
Response Time	0 - 299	300 - 999	1000 – 10,000
Throughput	0 – 9	10 – 19	20 -30

Additionally, we defined three linguistic variables for the output, which is the trust value of the fog node: highly trustworthy, trustworthy, and untrustworthy. The range for each of these linguistic variables is presented in Table 8.4. The membership function of the trust value describes how the output value belongs to each linguistic term, as shown in Figures 8.9.

Table 8.4 : Parametric range for the linguistic variables of the output parameters

Output Parameter	**Highly Trustworthy values range**	**Trustworthy values range**	**Untrustworthy values range**
Trust value	0 - 1	1 - 2	2 – 3

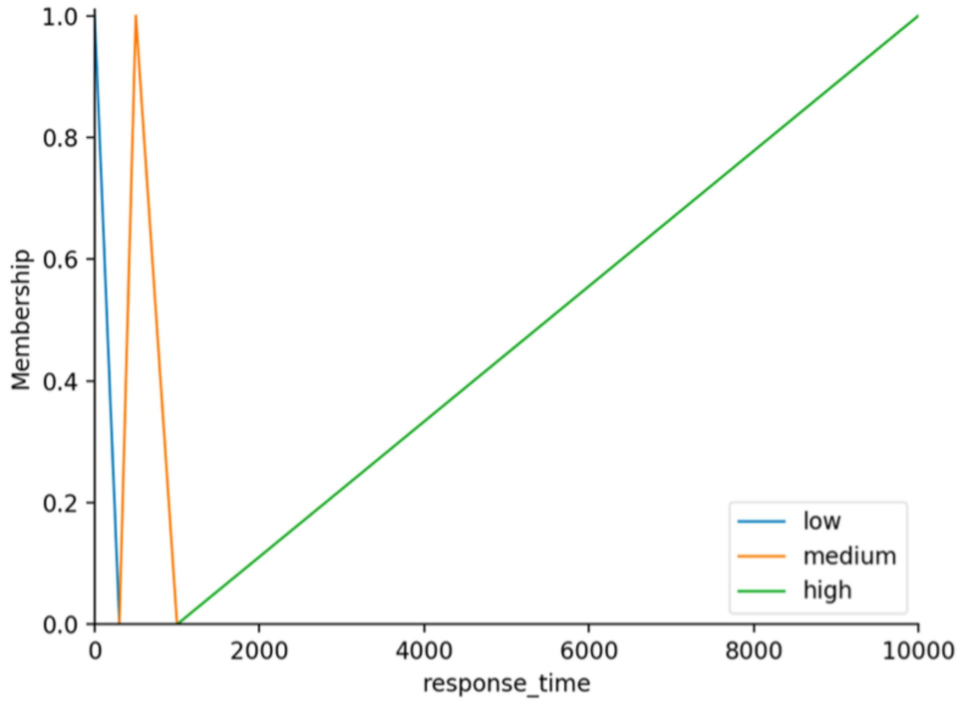


Figure 8.7 : Membership function of response time input

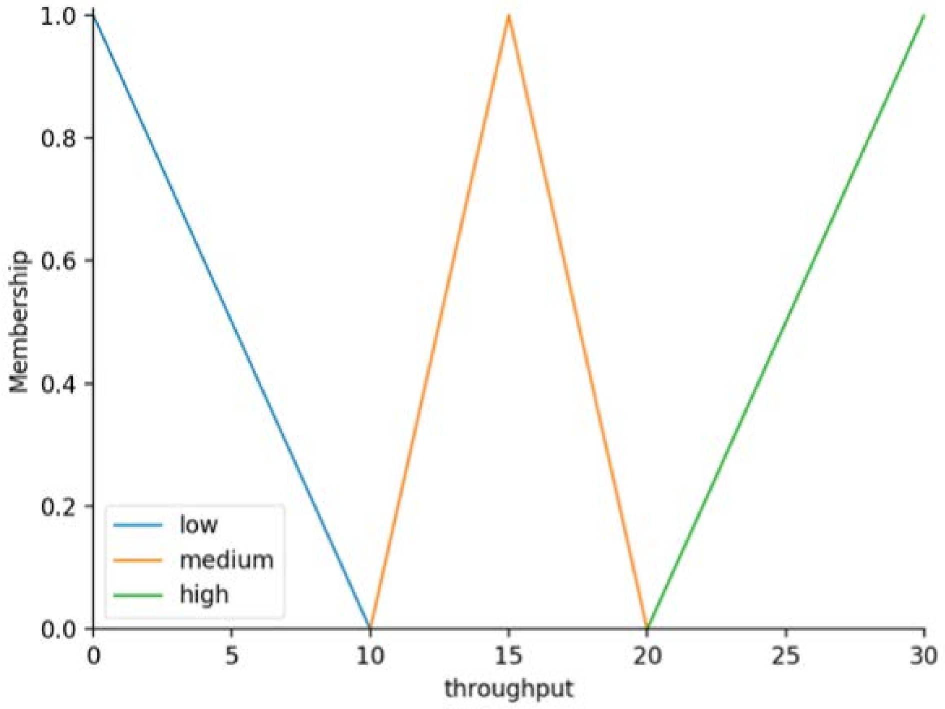


Figure 8.8 : Membership function of throughput input

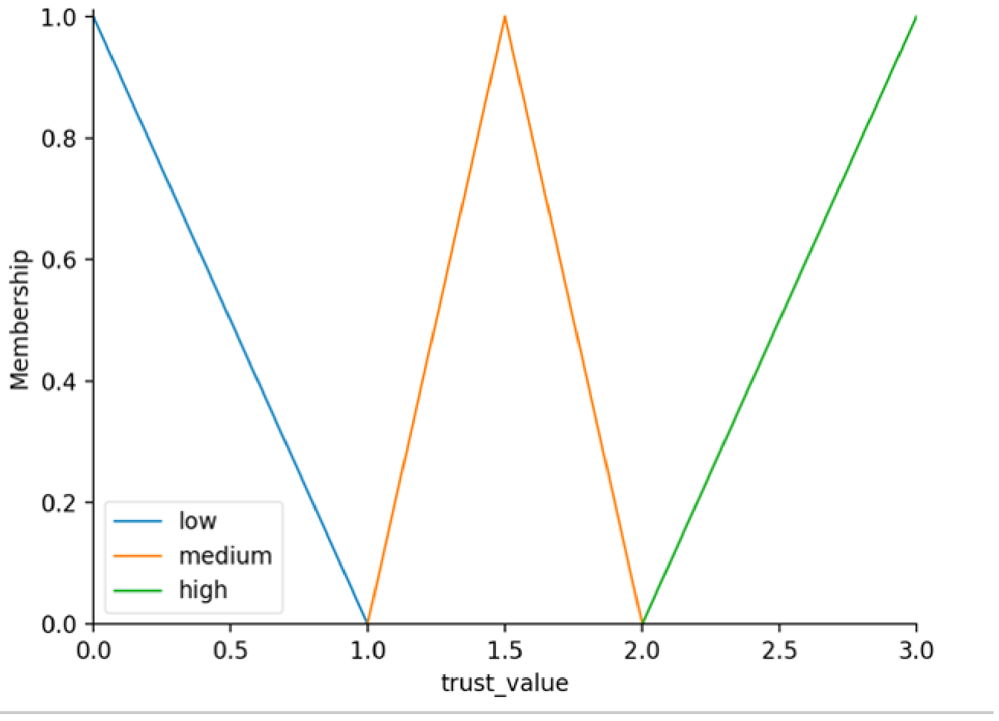


Figure 8.9 : Membership function of trust value output

Step 2: Set up fuzzy rules: Fuzzy rules capture the relationships between input QoS factors and output trust values. These rules are expressed in the form of IF-THEN statements, where each antecedent (IF part) represents a combination of linguistic terms from the input factors, and the consequent (THEN part) represents the trust value linguistic term. Table 8.5 details the fuzzy rules of the prediction model.

Step 3: Fuzzy Inference: Fuzzy inference applies the fuzzy rules to the input QoS values to determine the trust value. The fuzzification process converts crisp input values into fuzzy sets using the defined membership functions. The rules' antecedents are evaluated using fuzzy logic operators (AND) to determine their degree of truth. The fuzzy outputs are then aggregated to obtain a combined trust value using operators like the maximum average.

Table 8.5 : Fuzzy rules for the Bootstrapping Engine

Response time	**Throughput**	**Trust value**
Low	High	Highly trustworthy
Low	Medium	Trustworthy
Low	Low	Untrustworthy
Medium	High	Highly trustworthy
Medium	Medium	Trustworthy
Medium	Low	Untrustworthy
High	High	Untrustworthy
High	Medium	Trustworthy
High	Low	Highly trustworthy

Step 4: Defuzzification: Defuzzification converts the aggregated fuzzy trust value back into a crisp value for interpretation. Various defuzzification methods can be used, such as centroid, the mean of maximum (MOM), or weighted average. In this research, we used the centroid defuzzification method. The defuzzified trust value represents the predicted reputation of the fog node. It falls into one of the three distinct linguistic terms: highly trustworthy, trustworthy, or untrustworthy. The trust value provides insights into the level of confidence associated with the fog node's reliability and trustworthiness.

- If the predicted trust value is between (0 - 1), then the trust value is 1 (highly trustworthy).
- If the predicted trust value is between (1 - 2), then the trust value is 2 (trustworthy).
- If the predicted trust value is between (2 - 3), then the trust value is 3 (untrustworthy).

By employing fuzzy logic in this manner, we can effectively predict the trust value of new fog nodes based on their QoS factors. The fuzzy logic approach is well-suited for handling uncertainty and vagueness in QoS data, making it a valuable tool for reputation prediction in fog computing environments. The interpretability of the fuzzy inference system enables users to understand and trust the predictions, supporting decision-making processes related to resource allocation and service selection.

8.3.3.2 Regression-Based Reputation Prediction

The regression-based reputation prediction method leverages regression algorithms to estimate the reputation of a new fog node. These algorithms establish a mathematical relationship between the fog node's QoS attributes, such as response time and throughput, and its corresponding trust value. By analyzing the QoS data and applying regression techniques, this approach provides a quantitative estimation of the fog node's trust value based on its observed QoS performance. For this method, we specifically utilize the logistic regression algorithm [159], which is a supervised machine learning algorithm commonly used for classification and regression tasks. Logistic regression aims to predict the probability of an instance belonging to a specific class. In our research, logistic regression is employed to predict the reputation of the new fog node. The reputation is represented by a trust value categorized into three distinct levels: 1 (highly trustworthy), 2 (trustworthy), and 3 (untrustworthy). We implement logistic regression using the scikit-learn library APIs in Python [145]. To train and evaluate the logistic regression model, we divide the WS-dream dataset into training and test sets. The logistic regression method is then applied to classify the fog nodes based on their QoS attributes. To assess the performance of the method, we calculate various evaluation metrics such as accuracy, precision, recall, and F1 score. Additionally, we provide a classification report and a confusion matrix to gain further insights into the classification results. By employing the logistic regression algorithm within the reputation prediction module, we can quantitatively estimate the trust value of the new fog node. This approach enhances decision-making processes and resource allocation strategies, as it provides a numerical indication of the fog node's expected trustworthiness based on its observed QoS attributes.

8.3.3.3 Deep Learning-Based Reputation Prediction

The third method employed for reputation prediction involves the use of deep learning techniques, specifically deep neural networks, to predict the trust value of the new cold-start fog node. By training deep learning models with the QoS data from existing fog nodes, these models are capable of capturing intricate patterns and relationships present in the data. Subsequently, the trained models can be utilized to predict the trust value of the new cold-start fog node based on its QoS attributes, resulting in a highly accurate estimation of its trust value. In our rsearch, we specifically utilized the multilayer perceptron (MLP) neural network as a deep learning model for reputation prediction. The construction of the MLP neural network was facilitated by the Keras library [151][160], a powerful Python library that integrates popular deep learning backends such as TensorFlow and Theano, ensuring the efficient implementation and utilization of the MLP neural network. Our implementation of the MLP neural network model consists of three hidden layers. The first hidden layer is composed of 24 nodes, the second hidden layer contains 12 nodes, and the third hidden layer comprises 8 nodes. These hidden layers are equipped with the rectified linear unit (ReLU) activation function, which enables the network to effectively capture non-linear relationships within the data. Furthermore, the weights of the network are initialized using a commonly used method, a weight initialization technique that facilitates proper network convergence. By leveraging the deep learning technique and employing the MLP neural network model, we are able to effectively exploit the complex patterns and dependencies within the QoS data to predict the reputation of the new fog node with a high degree of accuracy. This approach significantly enhances the precision and reliability of reputation prediction by considering intricate non-linear relationships. The adoption of Keras simplifies the model construction process and ensures the efficient training and evaluation of the MLP neural network.

We compared the three reputation prediction methods, namely the fuzzy logic-based, regression-based, and deep learning-based approaches and several factors need to be considered to determine the best approach (further information is provided in Section 8.4.2). Accuracy plays a crucial role in evaluating the performance of each method. The accuracy metric measures how closely the predicted reputation aligns with the actual reputation. The method that achieves the highest accuracy in predicting the trust value of the new fog node can be considered the most effective. Secondly, the complexity of implementing and deploying each method is an important consideration. This includes factors such as ease of training the models, computational resource requirements, and overall algorithmic complexity. If a less complex method can achieve comparable accuracy to more complex approaches, it may be preferred. The interpretability of the reputation prediction results is another significant factor. It refers to how easily the predicted reputation can be understood and justified based on the input QoS attributes. In scenarios where explainability is crucial, a method that provides transparent and interpretable results may be preferred.

8.4 Evaluation and discussion

This section focuses on evaluating the modules proposed in the BE framework based on the evaluation process explained in Section 4.7.2. Firstly, in Section 8.4.1, we evaluate the QoS prediction module, which utilizes clustering and the KNN algorithm. The aim of this module is to predict the QoS values of new cold-start fog nodes by considering the geographical nearest fog nodes within their clusters. Through extensive evaluation, we assess the effectiveness of this module in accurately predicting QoS values for cold-start fog nodes. In Section 8.4.2, we evaluate the reputation prediction module. This module incorporates three AI-based methods for reputation prediction. The primary objective of the reputation prediction module is to predict the trust value of new fog nodes based on the predicted QoS values obtained from the QoS prediction module. We thoroughly assess the performance and effectiveness of these AI-based methods in accurately predicting the trustworthiness of new fog nodes. By conducting comprehensive evaluations of both the QoS prediction module and the reputation prediction module, we gain valuable insights into the overall performance and reliability of the BE framework. The evaluation process allows us to gauge the effectiveness of these modules in addressing the challenges associated with cold-start fog nodes and ensuring the trustworthiness of the fog computing environment.

8.4.1 Evaluation of QoS prediction module

The evaluation of our QoS prediction approach based on geographical nearest fog nodes obtained from the KNN algorithm has yielded highly accurate results. To assess the accuracy of our predictions, we compared the predicted QoS values with the actual QoS values observed during the operation of the new fog node. Through extensive testing and analysis, we determined that our predicted QoS values for the new cold-start fog node achieved an impressive accuracy rate of 85%. By accurately estimating the QoS values for the cold-start fog node, our prediction-based approach enables us to gain valuable insights into its anticipated performance. This information is crucial in facilitating an efficient selection process for integrating the fog node into the network. By leveraging the KNN algorithm and considering the QoS values of the nearest fog nodes in the region or cluster, we can make informed predictions about the QoS performance of the new fog node, even in the absence of sufficient historical data. The high accuracy achieved by our QoS prediction module demonstrates its effectiveness in addressing the uncertainties associated with the cold start problem. By accurately predicting the QoS values based on the characteristics of the nearest fog nodes, we can predict the trust value of the new fog node into the network. This enables smoother operations and minimizes the disruptions typically associated with the cold start phase. The obtained accuracy rate of 85% indicates that our prediction-based approach provides reliable and trustworthy QoS estimations for cold-start fog nodes. This accuracy enables more efficient decision-making in terms of reputation prediction than FNS.

8.4.2 Evaluation of reputation prediction module

We evaluated the three methods of the reputation prediction module based on the commonly known evaluation metrices, accuracy, precision, recall, and F1 score. By carefully evaluating and comparing these methods, it is possible to identify the best approach of the three methods. Comprehensive experiments and performance evaluations are conducted to determine the most suitable approach for a particular fog computing environment. Through the implementation of these methods individually within the reputation prediction module, we compare and evaluate the results of each method. The BE module should provide a comprehensive and robust approach to predict the trust value of the new fog node, facilitating the decision-making process for integrating the fog node into the network. This ensures that only trusted and reliable fog nodes are selected, contributing to the overall integrity and reliability of the fog computing environment. The comprehensive evaluation of all the proposed methods offers valuable insights into the best reputation prediction process for new cold-start fog nodes. By employing the most effective intelligent method determined through this evaluation, the BE module enhances the reputation prediction process for new cold-start fog nodes. Each method brings unique advantages, allowing the system to select the most suitable technique based on the specific requirements and characteristics of the fog computing environment. The evaluation process enables the selection of a method that ensures the reliability and high accuracy of the reputation prediction process, contributing to the overall effectiveness and reliability of the fog computing environment. Leveraging one of these fuzzy logic, regression, or deep learning techniques, the BE module enhances the capability of the selection framework to handle the cold start problem and assess the trustworthiness of new fog nodes. This intelligent prediction mechanism enables efficient FNS and improved overall performance within the fog computing network. Several experiments have been conducted for the three methods and the evaluation results of the four metrices are depicted in the Table 8.6.

Table 8.6 : Evaluation metric results of BE with the three approaches

	Fuzzy Logic	**Logistic Regression**	**DNN**
Accuracy	90.74	78.42	57.14
Precision	91.09	79.67	55.46
Recall	93.51	78.42	57.14
F1 score	92.28	77.56	56.0

Using the information provided in Table 8.6 and Figure 8.10 Table 8.6 and Figure 8.10, we can comprehensively compare the performance of the three methods: fuzzy logic, logistic regression, and DNN for predicting the trust value of new cold-start fog node. Fuzzy logic achieved the highest accuracy of 90.74%. This indicates that the model accurately predicts the trust value of fog nodes based on QoS factors,

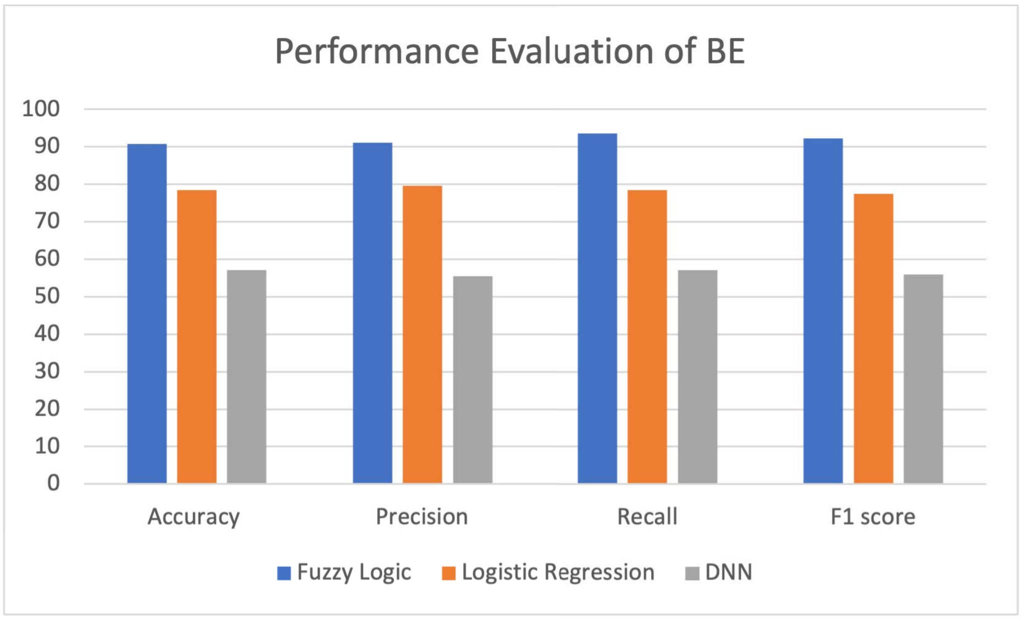


Figure 8.10 : Overall evaluation of BE

leading to better decision making in fog computing environments. The precision of 91.09% suggests that fuzzy logic effectively minimizes false positives, reducing the likelihood of incorrectly classifying trustworthy fog nodes as untrustworthy or vice versa. With a recall of 93.51%, fuzzy logic excels in identifying highly trustworthy, trustworthy, or untrustworthy fog nodes, minimizing false negatives and improving the overall trust prediction process. Fuzzy logic achieved an impressive F1 score of 92.28%, indicating a good balance between precision and recall, ensuring accurate positive predictions and minimizing errors in trust evaluation.

However, logistic regression achieved an accuracy of 78.42%, which is lower than fuzzy logic. It suggests that the model may not fully capture the complex relationships between QoS factors and the trust value, leading to reduced predictive accuracy. The precision of 79.67% indicates that logistic regression effectively minimizes false positives, similar to fuzzy logic, but slightly less accurate in positive predictions. With a recall of 78.42%, logistic regression may miss identifying some truly trustworthy or untrustworthy fog nodes, leading to a higher number of false negatives compared to fuzzy logic. The F1 score of 77.56% further confirms the trade-off between precision and recall, where logistic regression slightly lags behind fuzzy logic in achieving a balanced performance.

On the other hand, DNN obtained an accuracy of 57.14%, significantly lower than both fuzzy logic and logistic regression. This suggests that the DNN model may not be capturing the underlying patterns in the data effectively. The precision of 55.46% indicates a higher number of false positives, making DNN less precise in predicting trustworthy or untrustworthy fog nodes compared to the other two methods. With a recall of 57.14%, DNN's ability to identify truly trustworthy or untrustworthy fog nodes is comparable to its accuracy, indicating a less reliable trust prediction performance. The F1 score of 56.0% further confirms the challenges in achieving a balanced performance between precision and recall for DNN. Based on the evaluation results, fuzzy logic outperforms both logistic regression and DNN in all evaluation metrics, including accuracy, precision, recall, and F1 score. It demonstrates its superiority in trust value prediction due to its ability to handle uncertainty and imprecision inherent in fog computing data. Logistic regression shows relatively good precision and recall, but it lags behind fuzzy logic in overall accuracy and F1 score. It may be limited in capturing complex relationships in the trust evaluation task. DNN performs the least effectively of the three methods, with significantly lower accuracy, precision, recall, and F1 score. This could be attributed to challenges in optimizing the complex neural network architecture or inadequate training data.

Fuzzy logic emerges as the most effective method for predicting the trust value of fog nodes based on QoS factors. Its robust performance, balanced precision and recall, and high accuracy make it a preferable choice for trust evaluation in fog computing environments. Logistic regression shows promising results but falls short of fuzzy logic's performance, while DNN requires further optimization to achieve better accuracy and reliability in trust prediction.

8.5 Conclusion

In conclusion, the cold-start problem poses a significant challenge in reputation and trust systems, particularly in the context of FNS. The lack of historical information and QoS data for new fog nodes renders them ineligible for objective ranking and trust-based evaluations. This limitation creates a disadvantage for new fog node providers as they struggle to establish credibility and participate effectively in collaborative tasks. To address this issue, we proposed the BE as an intelligent-based trust evaluation agent within the FNSE framework. The BE plays a crucial role in predicting and evaluating the trust value of new fog nodes that join the fog ecosystem during the cold start phase. By leveraging intelligent techniques, the BE enables the FNSE to make reliable selections based on trust values. The BE's functionality revolves around predicting the initial QoS values for new fog nodes by QoS prediction module and subsequently estimating their trust values by the reputation prediction module. Several experiments are conducted to measure the accuracy of the trust value prediction and their results show that the fuzzy logic-based reputation prediction method outperforms the logistic regression and DNN approaches in terms of accuracy, precision, recall, and F1 score. The fuzzy logic approach achieved the highest accuracy compared with other proposed approaches. The proposed BE mechanism not only addresses the cold-start problem but also empowers the FNSE to make informed and trustworthy FNSs. By providing a reliable assessment of the trustworthiness of new fog nodes, the BE contributes to the overall effectiveness and efficiency of the fog ecosystem. It ensures that fog consumers can make informed decisions based on trust values derived from historical QoS attributes.

Chapter 9

Conclusion and Future Work

9.1 Introduction

This chapter marks the end of the thesis and offers insights into potential directions for future research. The primary objective of the thesis has been to identify and select the most suitable fog nodes that meet the requirements of fog consumers. This aim has been accomplished through the development of the distributed consortium which we called FRC. For this FRC, we proposed intelligent frameworks for context-aware FND, trust-based FNS and to address the cold-start problem in fog computing. The proposed approach incorporates state-of-the-art techniques from artificial intelligence, including machine learning, fuzzy logic, and deep learning, to enhance the efficiency and effectiveness of the fog ecosystem. By leveraging these AI-driven methodologies, the FND and FNS domains have been significantly improved, leading to more optimal fog node selection and fog resource allocation.

This chapter is organised as follows: In Section 9.2, the research issues of this thesis are addressed. The contributions of this thesis to the existing literature are discussed in Section 9.3. Future research directions are presented in Section 9.4.

9.2 Problems Addressed in This Thesis

This thesis examines the issues of FND and FNS using AI frameworks. In Chapter 2, we provided an overview of the existing research and highlighted the problems to be addressed in this thesis. Our research objectives are as follow:

1. Propose a distributed framework to host fog node information in the network

and keep the concurrency, consistency, and integrity of the fog network.

2. Propose and develop an intelligent and distributed framework for FND that is capable of allocating the optimal fog node based on the context-aware data of fog nodes and fog consumers using three nearest neighbor algorithms.

3. Propose and develop an intelligent and reliable framework, FNS, that helps the fog consumer select the most suitable fog node based on the QoS of fog nodes using machine learning, fuzzy logic, and DNN methods.

4. Propose and develop an intelligent trust evaluation system by predicting the trustworthiness of fog nodes and bootstrapping the reputation of the new cold-start fog node.

9.3 Contributions of This Thesis to the Existing Literature

To achieve the research objectives outlined in Section 9.2, this thesis develops an intelligent, reliable, and distributed framework for discovering and selecting the best fog node options. The contributions of this thesis are summarized as follows:

9.3.1 Contribution 1: Comprehensive State-of-the-art Survey of the Existing Literature

In the first part of this thesis, we conducted an extensive research study focusing on FND and FNS. We provided the first SLR in the area of FND and FNS and presented a critical analysis of the existing FND approaches, highlighting their shortcomings in meeting the defined requirements. We also critically evaluated the existing FNS approaches and identified the gaps in meeting the defined requirements. Additionally, the limitations of the existing FND and FNS approaches that need to be addressed are discussed. The findings from this study highlighted a significant need to explore the development of context-aware FND in an intelligent and scalable manner. Remarkably, none of the studies in the existing literature presented a comprehensive framework for an intelligent and reliable trust-based assessment in a distributed fog environment. Additionally, the research revealed a dearth of studies exploring trust-based FNS relying on distributed registries. Moreover, no prior works investigated the intelligent prediction of the trust value for new fog nodes, which is essential for seamlessly integrating them into the fog ecosystem. The SLR is presented in Chapter 2 of this thesis and has been published in the Q1 journal, Future Generation Computer Systems [10].

9.3.2 Contribution 2: Build an FRC architecture that can host important information about fog nodes and ensure concurrency between the DFRs within the consortium

In the next phase of this thesis, we developed a novel FRC that stores extensive data on the fog nodes in the network and we also developed a distributed mechanism to ensure concurrency and consistency between the DFRs within the consortium. The functionality, efficiency, and effectiveness of the FRC in managing fog nodes, synchronizing data across DFRs, ensuring the integrity and consistency of information, and facilitating communication between fog consumers and fog nodes within the simulated network was verified. A detailed explanation of the FRC framework and its implementation algorithms is given in Chapter 5. The content of this chapter is published in the Q1 IoT journal [110].

9.3.3 Contribution 3: An intelligent and distributed context-aware fog node discovery framework using three AI-driven methods

In the next stage of this thesis, we developed the intelligent and distributed FNDE to help the fog consumer identify and locate the nearest fog nodes intelligently based on the location and context awareness of fog consumers and fog nodes. Three AI-driven models, namely KNN, K-d tree and brute force algorithms are applied to discover the nearest optimal fog nodes to the fog consumer. To analyze the performance of these models, accuracy, precision, recall, F1 score, and execution time are measured. The evaluation results show that the k-d tree model performs better than the KNN and brute force models in term of accuracy and time consumption. Applying k-d tree in FNDE highlights the efficiency of the k-d tree in finding the nearest fog nodes, making it a favorable choice for fog consumers seeking faster results. A detailed description of the FNDE framework and the algorithms used in the implementation is given in Chapter 6. This chapter has been published in the Q1 IoT journal [110].

9.3.4 Contribution 4: An intelligent and reliable fog node selection framework using AI-driven algorithms

As part of this research endeavor, we developed the intelligent and reliable FNSE which helps users make informed decisions when selecting trusted fog nodes within a distributed fog environment. To do this, three AI-driven models of the TEE were utilized, namely fuzzy logic, logistic regression, and deep learning for predicting the trust value of fog nodes based on QoS parameters. The performance analysis of these TEE models involved measuring accuracy, precision, recall, F1 score, and execution time. The evaluation outcomes demonstrated that the TEE utilizing the fuzzy logic approach had the most promising and best performance, achieving remarkable accuracy, precision, recall, and F1 score values, thereby affirming its effectiveness in trust prediction. Additionally, the fuzzy logic approach exhibited efficiency by consuming less time, thereby facilitating rapid predictions. This enhanced efficiency ensures that fog node trust values can be accurately and swiftly assessed, contributing to the overall reliability and efficacy of the FNSE. Chapter 7 describes the FNSE framework and its implementation algorithms in detail.

9.3.5 Contribution 5: An intelligent reputation approach for bootstrapping new fog nodes into a fog ecosystem

In this research, we successfully developed an intelligent approach to tackle the challenge of bootstrapping new fog nodes into a fog ecosystem. Our proposed framework, called the BE, is an intelligent and trust-based solution designed specifically to address the cold-start problem in fog computing environments. The primary goal of the BE is to equip fog consumers with the capability to make well-informed and reliable decisions when selecting fog nodes for their applications. To do this, we employed three AI-driven models, namely fuzzy logic, logistic regression, and deep learning, to predict the trust value of new cold-start fog nodes based on QoS parameters. The performance analysis of these models involved measuring crucial metrics such as accuracy, precision, recall, F1 score, and execution time. The evaluation outcomes clearly indicated that the fuzzy logic-based reputation prediction method outperformed the logistic regression and deep learning approaches in terms of accuracy, precision, recall, and F1 score. Notably, the fuzzy logic approach achieved the highest accuracy compared to the other proposed methodologies. The introduction of the BE mechanism addresses the cold-start problem comprehensively, allowing the FNSE to make informed and trustworthy fog node selections. By empowering the FNSE with this intelligent and trust-based capability, fog consumers can now confidently choose fog nodes that align with their specific requirements, thus enhancing the overall reliability and efficiency of the fog computing ecosystem. Chapter 8 provides a detailed explanation of the BE framework and its implementation algorithms.

9.4 Future Research Directions

Despite the comprehensive research conducted on the subject, there are numerous opportunities for future exploration. As a result, our focus will continue in this area, with an emphasis on, although not limited to, the following aspects:

1. **XaaS-FND (where X could be location, context, or content):** In this research, we have designed an intelligent FNDE utilizing location-based context-awareness. Our approach focuses on creating an intelligent single criterion that calculates the proximity between a fog consumer and a fog node. Notably, our method is flexible and can adapt to different proximity quantification parameters. In future work, we envision exploring the development of intelligent multi-criteria-driven node discovery approaches. These approaches will consider diverse parameters that signify proximity, including bandwidth, physical distance, latency, and more. By incorporating multiple criteria, we aim to further enhance the precision and effectiveness of FND, catering to various application scenarios and optimizing the overall performance of the fog computing ecosystem.

2. **Multi-factor based holistic reputation computation:** As part of our research, we successfully created intelligent TEE and BE models, which play a pivotal role in predicting the trust value of fog nodes. This predictive capability assists fog consumers in making informed decisions when selecting reliable fog nodes based on the QoS provided within the fog environment. In our future work, we recognize the significance of additional factors in evaluating the trustworthiness of a fog node. These factors may include assessing the reputation of the fog node provider and considering the quality of experience reported by other fog consumers who have interacted with the same or similar fog nodes. By incorporating these vital factors into our evaluation framework, we aim to further enhance the accuracy and completeness of fog node trust assessments, ultimately empowering fog consumers with more comprehensive information for their decision-making processes.

3. **Platform for objective comparison of fog models:** The research study presented in Chapter 2 highlights that various authors have employed diverse evaluation approaches, including simulations and real-world data, to demonstrate the performance of their models. However, a notable observation is the absence of a single, definitive source of truth or benchmark mechanism that can be universally adopted by different approaches for evaluation. This lack of standardization extends to both the metrics utilized for evaluating different algorithms and the benchmarking of their performance. Within the distributed fog computing literature, platforms and methodologies differ significantly, leading to the absence of a standardized data set for evaluation. As a consequence, biases in the results may arise due to these inconsistencies. To address this challenge, future research should prioritize the development of standardized metrics and benchmarks. These efforts will prove instrumental in facilitating fair and objective comparisons among different approaches, fostering a more cohesive and informed advancement of the field of distributed fog computing. By establishing a common ground for evaluation, researchers can collectively work towards fostering advancements that lead to more robust and effective solutions in the context of fog node discovery, selection, and beyond.

4. **Approaches for securing FRC:** In this thesis, we proposed the architecture and detailed working of FRC, however, we did not focus on the data security mechanisms for FRC. This topic is out of the scope of this research. In future work, we will look at various approaches such as, but not limited to, blockchain for data security of FRC.

Appendix A

Chapter 7 Fuzzy Rules

Table A.1 : Development of Fuzzy Rules for Trust Value Prediction in Experiment Scenario 1

Rule	**IF Response Time**	**AND Availability**	**AND Throughput**	**AND Successability**	**AND Reliability**	**AND WsRF**	**Then Trust vale**
1	High	High	Medium	High	High	High	Highly Trustworthy
2	High	High	Low	High	High	High	Highly Trustworthy
3	High	High	Low	High	Medium	High	Highly Trustworthy
4	High	High	Low	High	Low	High	Highly Trustworthy

Rule	**IF Response Time**	**AND Availability**	**AND Throughput**	**AND Successability**	**AND Reliability**	**AND WsRF**	**Then Trust vale**
5	High	High	Low	Medium	High	High	Highly Trustworthy
6	High	High	Low	Medium	Medium	High	Highly Trustworthy
7	High	High	Low	Medium	Low	High	Highly Trustworthy
8	High	Medium	Low	Medium	High	High	Highly Trustworthy
9	High	Medium	Low	Medium	Medium	High	Highly Trustworthy
10	High	Medium	Low	Medium	Low	High	Highly Trustworthy

Rule	**IF Response Time**	**AND Availability**	**AND Throughput**	**AND Successability**	**AND Reliability**	**AND WsRF**	**Then Trust vale**
11	High	Medium	Low	Low	High	High	Highly Trustworthy
12	High	Medium	Low	Low	Medium	High	Highly Trustworthy
13	High	Medium	Low	Low	Low	High	Highly Trustworthy
14	High	Low	Low	Low	High	High	Highly Trustworthy
15	High	Low	Low	Low	Medium	High	Highly Trustworthy
16	High	Low	Low	Low	Low	High	Highly Trustworthy

Rule	**IF Response Time**	**AND Availability**	**AND Throughput**	**AND Successability**	**AND Reliability**	**AND WsRF**	**Then Trust vale**
17	Medium	High	Medium	High	High	High	Highly Trustworthy
18	Medium	High	Medium	Medium	High	High	Highly Trustworthy
19	Medium	High	Medium	Low	High	High	Highly Trustworthy
20	Medium	High	Medium	Low	Medium	High	Highly Trustworthy
21	Medium	High	Medium	Low	Low	High	Highly Trustworthy
22	Medium	High	Low	High	High	High	Highly Trustworthy

Rule	**IF Response Time**	**AND Availability**	**AND Throughput**	**AND Successability**	**AND Reliability**	**AND WsRF**	**Then Trust vale**
23	Medium	High	Low	High	Medium	High	Highly Trustworthy
24	Medium	High	Low	High	Low	High	Highly Trustworthy
25	Medium	High	Low	Medium	High	High	Highly Trustworthy
26	Medium	High	Low	Medium	Medium	High	Highly Trustworthy
27	Medium	High	Low	Medium	Low	High	Highly Trustworthy
28	Medium	Medium	Low	Medium	High	High	Highly Trustworthy

Rule	**IF Response Time**	**AND Availability**	**AND Throughput**	**AND Success-ability**	**AND Reliability**	**AND WsRF**	**Then Trust vale**
29	Medium	Medium	Low	Medium	Medium	High	Highly Trustworthy
30	Medium	Medium	Low	Medium	Low	High	Highly Trustworthy
31	Medium	Medium	Low	Low	High	High	Highly Trustworthy
32	Medium	Medium	Low	Low	Medium	High	Highly Trustworthy
33	Medium	Medium	Low	Low	Low	High	Highly Trustworthy
34	Medium	Low	Low	Low	High	High	Highly Trustworthy

Rule	**IF Response Time**	**AND Availability**	**AND Throughput**	**AND Successability**	**AND Reliability**	**AND WsRF**	**Then Trust vale**
35	Medium	Low	Low	Low	Medium	High	Highly Trustworthy
36	Medium	Low	Low	Low	Low	High	Highly Trustworthy
37	Low	High	High	High	High	High	Highly Trustworthy
38	Low	High	High	Medium	High	High	Highly Trustworthy
39	Low	High	High	Medium	Medium	High	Highly Trustworthy
40	Low	High	High	Medium	Low	High	Highly Trustworthy

Rule	**IF Response Time**	**AND Availability**	**AND Throughput**	**AND Successability**	**AND Reliability**	**AND WsRF**	**Then Trust vale**
41	Low	High	High	Low	High	High	Highly Trustworthy
42	Low	High	High	Low	Medium	High	Highly Trustworthy
43	Low	High	High	Low	Low	High	Highly Trustworthy
44	Low	High	Medium	High	High	High	Highly Trustworthy
45	Low	High	Medium	High	Medium	High	Highly Trustworthy
46	Low	High	Medium	High	Low	High	Highly Trustworthy

Rule	**IF Response Time**	**AND Availability**	**AND Throughput**	**AND Successability**	**AND Reliability**	**AND WsRF**	**Then Trust vale**
47	Low	High	Medium	Medium	High	High	Highly Trustworthy
48	Low	High	Medium	Medium	Medium	High	Highly Trustworthy
49	Low	High	Medium	Medium	Low	High	Highly Trustworthy
50	Low	High	Medium	Low	High	High	Highly Trustworthy
51	Low	High	Medium	Low	Medium	High	Highly Trustworthy
52	Low	High	Medium	Low	Low	High	Highly Trustworthy

Rule	**IF Response Time**	**AND Availability**	**AND Throughput**	**AND Successability**	**AND Reliability**	**AND WsRF**	**Then Trust vale**
53	Low	High	Low	High	High	High	Highly Trustworthy
54	Low	High	Low	High	Medium	High	Highly Trustworthy
55	Low	High	Low	High	Low	High	Highly Trustworthy
56	Low	High	Low	Medium	High	High	Highly Trustworthy
57	Low	High	Low	Medium	Medium	High	Highly Trustworthy
58	Low	High	Low	Medium	Low	High	Highly Trustworthy

Rule	**IF Response Time**	**AND Availability**	**AND Throughput**	**AND Successability**	**AND Reliability**	**AND WsRF**	**Then Trust vale**
59	Low	High	Low	Low	High	High	Highly Trustworthy
60	Low	Medium	Medium	Medium	High	High	Highly Trustworthy
61	Low	Medium	Medium	Medium	Medium	High	Highly Trustworthy
62	Low	Medium	Medium	Medium	Low	High	Highly Trustworthy
63	Low	Medium	Medium	Low	High	High	Highly Trustworthy
64	Low	Medium	Medium	Low	Medium	High	Highly Trustworthy

Rule	IF Response Time	AND Availability	AND Throughput	AND Successability	AND Reliability	AND WsRF	Then Trust vale
65	Low	Medium	Medium	Low	Low	High	Highly Trustworthy
66	Low	Medium	Low	High	High	High	Highly Trustworthy
67	Low	Medium	Low	High	Medium	High	Highly Trustworthy
68	Low	Medium	Low	High	Low	High	Highly Trustworthy
69	Low	Medium	Low	Medium	High	High	Highly Trustworthy
70	Low	Medium	Low	Medium	Medium	High	Highly Trustworthy

Rule	IF Response Time	AND Availability	AND Throughput	AND Successability	AND Reliability	AND WsRF	Then Trust vale
71	Low	Medium	Low	Medium	Low	High	Highly Trustworthy
72	Low	Medium	Low	Low	High	High	Highly Trustworthy
73	Low	Medium	Low	Low	Medium	High	Highly Trustworthy
74	Low	Medium	Low	Low	Low	High	Highly Trustworthy
75	Low	Low	Low	Low	High	High	Highly Trustworthy
76	Low	Low	Low	Low	Medium	High	Highly Trustworthy

Rule	IF Response Time	AND Availability	AND Throughput	AND Successability	AND Reliability	AND WsRF	Then Trust vale
77	Low	Low	Low	Low	Low	High	Highly Trustworthy
78	High	High	Medium	High	High	Medium	Trustworthy
79	High	High	Low	High	High	Medium	Trustworthy
80	High	High	Low	High	Medium	Medium	Trustworthy
81	High	High	Low	High	Low	Medium	Trustworthy
82	High	High	Low	Medium	High	Medium	Trustworthy
83	High	High	Low	Medium	Medium	Medium	Trustworthy
84	High	High	Low	Medium	Low	Medium	Trustworthy
85	High	Medium	Low	Medium	High	Medium	Trustworthy
86	High	Medium	Low	Medium	Medium	Medium	Trustworthy
87	High	Medium	Low	Medium	Low	Medium	Trustworthy
88	High	Medium	Low	Low	High	Medium	Trustworthy
89	High	Medium	Low	Low	Medium	Medium	Trustworthy
90	High	Medium	Low	Low	Low	Medium	Trustworthy
91	High	Low	Low	Low	High	Medium	Trustworthy
92	High	Low	Low	Low	Medium	Medium	Trustworthy

Rule	**IF Response Time**	**AND Availability**	**AND Throughput**	**AND Successability**	**AND Reliability**	**AND WsRF**	**Then Trust vale**
93	High	Low	Low	Low	Low	Medium	Trustworthy
94	Medium	High	Medium	High	High	Medium	Trustworthy
95	Medium	High	Medium	Medium	High	Medium	Trustworthy
96	Medium	High	Medium	Low	High	Medium	Trustworthy
97	Medium	High	Medium	Low	Medium	Medium	Trustworthy
98	Medium	High	Medium	Low	Low	Medium	Trustworthy
99	Medium	High	Low	High	High	Medium	Trustworthy
100	Medium	High	Low	High	Medium	Medium	Trustworthy
101	Medium	High	Low	High	Low	Medium	Trustworthy
102	Medium	High	Low	Medium	High	Medium	Trustworthy
103	Medium	High	Low	Medium	Medium	Medium	Trustworthy
104	Medium	High	Low	Medium	Low	Medium	Trustworthy
105	Medium	Medium	Low	Medium	High	Medium	Trustworthy
106	Medium	Medium	Low	Medium	Medium	Medium	Trustworthy
107	Medium	Medium	Low	Medium	Low	Medium	Trustworthy
108	Medium	Medium	Low	Low	High	Medium	Trustworthy
109	Medium	Medium	Low	Low	Medium	Medium	Trustworthy
110	Medium	Medium	Low	Low	Low	Medium	Trustworthy

Rule	IF Response Time	AND Availability	AND Throughput	AND Successability	AND Reliability	AND WsRF	Then Trust vale
111	Medium	Low	Low	Low	High	Medium	Trustworthy
112	Medium	Low	Low	Low	Medium	Medium	Trustworthy
113	Medium	Low	Low	Low	Low	Medium	Trustworthy
114	Low	High	High	High	High	Medium	Trustworthy
115	Low	High	High	Medium	High	Medium	Trustworthy
116	Low	High	High	Medium	Medium	Medium	Trustworthy
117	Low	High	High	Medium	Low	Medium	Trustworthy
118	Low	High	High	Low	High	Medium	Trustworthy
119	Low	High	High	Low	Medium	Medium	Trustworthy
120	Low	High	High	Low	Low	Medium	Trustworthy
121	Low	High	Medium	High	High	Medium	Trustworthy
122	Low	High	Medium	High	Medium	Medium	Trustworthy
123	Low	High	Medium	High	Low	Medium	Trustworthy
124	Low	High	Medium	Medium	High	Medium	Trustworthy
125	Low	High	Medium	Medium	Medium	Medium	Trustworthy
126	Low	High	Medium	Medium	Low	Medium	Trustworthy
127	Low	High	Medium	Low	High	Medium	Trustworthy
128	Low	High	Medium	Low	Medium	Medium	Trustworthy

Rule	**IF Response Time**	**AND Availability**	**AND Throughput**	**AND Successability**	**AND Reliability**	**AND WsRF**	**Then Trust vale**
129	Low	High	Medium	Low	Low	Medium	Trustworthy
130	Low	High	Low	High	High	Medium	Trustworthy
131	Low	High	Low	High	Medium	Medium	Trustworthy
132	Low	High	Low	High	Low	Medium	Trustworthy
133	Low	High	Low	Medium	High	Medium	Trustworthy
134	Low	High	Low	Medium	Medium	Medium	Trustworthy
135	Low	High	Low	Medium	Low	Medium	Trustworthy
136	Low	High	Low	Low	High	Medium	Trustworthy
137	Low	Medium	Medium	Medium	High	Medium	Trustworthy
138	Low	Medium	Medium	Medium	Medium	Medium	Trustworthy
139	Low	Medium	Medium	Medium	Low	Medium	Trustworthy
140	Low	Medium	Medium	Low	High	Medium	Trustworthy
141	Low	Medium	Medium	Low	Medium	Medium	Trustworthy
142	Low	Medium	Medium	Low	Low	Medium	Trustworthy
143	Low	Medium	Low	High	High	Medium	Trustworthy
144	Low	Medium	Low	High	Medium	Medium	Trustworthy
145	Low	Medium	Low	High	Low	Medium	Trustworthy
146	Low	Medium	Low	Medium	High	Medium	Trustworthy

Rule	**IF Response Time**	**AND Availability**	**AND Throughput**	**AND Successability**	**AND Reliability**	**AND WsRF**	**Then Trust vale**
147	Low	Medium	Low	Medium	Medium	Medium	Trustworthy
148	Low	Medium	Low	Medium	Low	Medium	Trustworthy
149	Low	Medium	Low	Low	High	Medium	Trustworthy
150	Low	Medium	Low	Low	Medium	Medium	Trustworthy
151	Low	Medium	Low	Low	Low	Medium	Trustworthy
152	Low	Low	Low	Low	High	Medium	Trustworthy
153	Low	Low	Low	Low	Medium	Medium	Trustworthy
154	Low	Low	Low	Low	Low	Medium	Trustworthy
155	High	High	Medium	High	High	Low	Untrustworthy
156	High	High	Low	High	High	Low	Untrustworthy
157	High	High	Low	High	Medium	Low	Untrustworthy
158	High	High	Low	High	Low	Low	Untrustworthy
159	High	High	Low	Medium	High	Low	Untrustworthy
160	High	High	Low	Medium	Medium	Low	Untrustworthy
161	High	High	Low	Medium	Low	Low	Untrustworthy
162	High	Medium	Low	Medium	High	Low	Untrustworthy
163	High	Medium	Low	Medium	Medium	Low	Untrustworthy
164	High	Medium	Low	Medium	Low	Low	Untrustworthy

Rule	IF Response Time	AND Availability	AND Throughput	AND Successability	AND Reliability	AND WsRF	Then Trust vale
165	High	Medium	Low	Low	High	Low	Untrustworthy
166	High	Medium	Low	Low	Medium	Low	Untrustworthy
167	High	Medium	Low	Low	Low	Low	Untrustworthy
168	High	Low	Low	Low	High	Low	Untrustworthy
169	High	Low	Low	Low	Medium	Low	Untrustworthy
170	High	Low	Low	Low	Low	Low	Untrustworthy
171	Medium	High	Medium	High	High	Low	Untrustworthy
172	Medium	High	Medium	Medium	High	Low	Untrustworthy
173	Medium	High	Medium	Low	High	Low	Untrustworthy
174	Medium	High	Medium	Low	Medium	Low	Untrustworthy
175	Medium	High	Medium	Low	Low	Low	Untrustworthy
176	Medium	High	Low	High	High	Low	Untrustworthy
177	Medium	High	Low	High	Medium	Low	Untrustworthy
178	Medium	High	Low	High	Low	Low	Untrustworthy
179	Medium	High	Low	Medium	High	Low	Untrustworthy
180	Medium	High	Low	Medium	Medium	Low	Untrustworthy
181	Medium	High	Low	Medium	Low	Low	Untrustworthy
182	Medium	Medium	Low	Medium	High	Low	Untrustworthy

Rule	IF Response Time	AND Availability	AND Throughput	AND Successability	AND Reliability	AND WsRF	Then Trust vale
183	Medium	Medium	Low	Medium	Medium	Low	Untrustworthy
184	Medium	Medium	Low	Medium	Low	Low	Untrustworthy
185	Medium	Medium	Low	Low	High	Low	Untrustworthy
186	Medium	Medium	Low	Low	Medium	Low	Untrustworthy
187	Medium	Medium	Low	Low	Low	Low	Untrustworthy
188	Medium	Low	Low	Low	High	Low	Untrustworthy
189	Medium	Low	Low	Low	Medium	Low	Untrustworthy
190	Medium	Low	Low	Low	Low	Low	Untrustworthy
191	Low	High	High	High	High	Low	Untrustworthy
192	Low	High	High	Medium	High	Low	Untrustworthy
193	Low	High	High	Medium	Medium	Low	Untrustworthy
194	Low	High	High	Medium	Low	Low	Untrustworthy
195	Low	High	High	Low	High	Low	Untrustworthy
196	Low	High	High	Low	Medium	Low	Untrustworthy
197	Low	High	High	Low	Low	Low	Untrustworthy
198	Low	High	Medium	High	High	Low	Untrustworthy
199	Low	High	Medium	High	Medium	Low	Untrustworthy
200	Low	High	Medium	High	Low	Low	Untrustworthy

Rule	**IF Response Time**	**AND Availability**	**AND Throughput**	**AND Successability**	**AND Reliability**	**AND WsRF**	**Then Trust vale**
201	Low	High	Medium	Medium	High	Low	Untrustworthy
202	Low	High	Medium	Medium	Medium	Low	Untrustworthy
203	Low	High	Medium	Medium	Low	Low	Untrustworthy
204	Low	High	Medium	Low	High	Low	Untrustworthy
205	Low	High	Medium	Low	Medium	Low	Untrustworthy
206	Low	High	Medium	Low	Low	Low	Untrustworthy
207	Low	High	Low	High	High	Low	Untrustworthy
208	Low	High	Low	High	Medium	Low	Untrustworthy
209	Low	High	Low	High	Low	Low	Untrustworthy
210	Low	High	Low	Medium	High	Low	Untrustworthy
211	Low	High	Low	Medium	Medium	Low	Untrustworthy
212	Low	High	Low	Medium	Low	Low	Untrustworthy
213	Low	High	Low	Low	High	Low	Untrustworthy
214	Low	Medium	Medium	Medium	High	Low	Untrustworthy
215	Low	Medium	Medium	Medium	Medium	Low	Untrustworthy
216	Low	Medium	Medium	Medium	Low	Low	Untrustworthy
217	Low	Medium	Medium	Low	High	Low	Untrustworthy
218	Low	Medium	Medium	Low	Medium	Low	Untrustworthy

Rule	**IF Response Time**	**AND Availability**	**AND Throughput**	**AND Successability**	**AND Reliability**	**AND WsRF**	**Then Trust vale**
219	Low	Medium	Medium	Low	Low	Low	Untrustworthy
220	Low	Medium	Low	High	High	Low	Untrustworthy
221	Low	Medium	Low	High	Medium	Low	Untrustworthy
222	Low	Medium	Low	High	Low	Low	Untrustworthy
223	Low	Medium	Low	Medium	High	Low	Untrustworthy
224	Low	Medium	Low	Medium	Medium	Low	Untrustworthy
225	Low	Medium	Low	Medium	Low	Low	Untrustworthy
226	Low	Medium	Low	Low	High	Low	Untrustworthy
227	Low	Medium	Low	Low	Medium	Low	Untrustworthy
228	Low	Medium	Low	Low	Low	Low	Untrustworthy
229	Low	Low	Low	Low	High	Low	Untrustworthy
230	Low	Low	Low	Low	Medium	Low	Untrustworthy
231	Low	Low	Low	Low	Low	Low	Untrustworthy

Table A.2 : Development of Fuzzy Rules for Trust Value Prediction in Experiment Scenario 2

Rule	**IF Response Time**	**AND Availability**	**AND Throughput**	**AND Successability**	**AND WsRF**	**Then Trust vale**
1	Low	High	High	Medium	High	Highly Trustworthy
2	Low	High	Medium	High	High	Highly Trustworthy
3	Low	High	High	High	High	Highly Trustworthy
4	Low	High	Low	High	High	Highly Trustworthy
5	Medium	High	Medium	High	High	Highly Trustworthy
6	Low	High	Medium	Medium	High	Highly Trustworthy

Rule	**IF Response Time**	**AND Availability**	**AND Throughput**	**AND Successability**	**AND WsRF**	**Then Trust vale**
7	High	High	Medium	High	High	Highly Trustworthy
8	Low	High	Low	Medium	High	Highly Trustworthy
9	Medium	High	Medium	Medium	High	Highly Trustworthy
10	Low	Medium	Low	Medium	High	Highly Trustworthy
11	Low	Medium	Medium	Medium	High	Highly Trustworthy
12	Medium	High	Low	High	High	Highly Trustworthy

Rule	**IF Response Time**	**AND Availability**	**AND Throughput**	**AND Successability**	**AND WsRF**	**Then Trust vale**
13	Low	High	High	Low	High	Highly Trustworthy
14	High	High	Low	High	High	Highly Trustworthy
15	Low	High	Low	Medium	Medium	Trustworthy
16	Low	High	Medium	High	Medium	Trustworthy
17	Medium	High	Low	High	Medium	Trustworthy
18	Medium	High	Medium	High	Medium	Trustworthy
19	Low	High	Low	High	Medium	Trustworthy
20	Low	High	Medium	Medium	Medium	Trustworthy
21	Low	Medium	Low	Medium	Medium	Trustworthy
22	Medium	High	Medium	Medium	Medium	Trustworthy
23	High	High	Low	High	Medium	Trustworthy
24	High	High	Low	Medium	Medium	Trustworthy
25	Medium	Medium	Low	Medium	Medium	Trustworthy
26	Low	Medium	Medium	Medium	Medium	Trustworthy

Rule	IF Response Time	AND Availability	AND Throughput	AND Success-ability	AND WsRF	Then Trust vale
27	Low	High	Medium	Low	Medium	Trustworthy
28	Low	Medium	Low	Low	Medium	Trustworthy
29	Medium	High	Low	Medium	Medium	Trustworthy
30	Medium	High	Medium	High	Low	Untrustworthy
31	Low	High	Low	Medium	Low	Untrustworthy
32	High	High	Low	High	Low	Untrustworthy
33	Low	Medium	Low	Medium	Low	Untrustworthy
34	Low	Medium	Medium	Medium	Low	Untrustworthy
35	Medium	High	Low	High	Low	Untrustworthy
36	High	High	Low	Medium	Low	Untrustworthy
37	High	Medium	Low	Medium	Low	Untrustworthy
38	Medium	Medium	Low	Medium	Low	Untrustworthy
39	Medium	High	Low	Medium	Low	Untrustworthy
40	Low	Medium	Low	Low	Low	Untrustworthy
41	Low	Low	Low	Low	Low	Untrustworthy
42	Low	High	Medium	Medium	Low	Untrustworthy
43	Medium	High	Medium	Low	Low	Untrustworthy
44	Medium	Low	Low	Low	Low	Untrustworthy

Rule	IF Response Time	AND Availability	AND Throughput	AND Successability	AND WsRF	Then Trust vale
45	Medium	Medium	Low	Low	Low	Untrustworthy
46	High	Medium	Low	Low	Low	Untrustworthy
47	High	Low	Low	Low	Low	Untrustworthy

Table A.3 : Development of Fuzzy Rules for Trust Value Prediction in Experiment Scenario 3

Rule	IF Response Time	AND Availability	AND Throughput	AND WsRF	Then Trust vale
1	High	High	Medium	High	Highly Trustworthy
2	High	High	Low	High	Highly Trustworthy
3	High	Medium	Low	High	Highly Trustworthy
4	Medium	High	High	High	Highly Trustworthy
5	Medium	Medium	High	High	Highly Trustworthy

Rule	**IF Response Time**	**AND Availability**	**AND Throughput**	**AND WsRF**	**Then Trust vale**
6	Medium	High	Medium	High	Highly Trustworthy
7	Medium	High	Low	High	Highly Trustworthy
8	Medium	Medium	Low	High	Highly Trustworthy
9	Low	High	High	High	Highly Trustworthy
10	Low	Medium	High	High	Highly Trustworthy
11	Low	High	Medium	High	Highly Trustworthy
12	Low	Medium	Medium	High	Highly Trustworthy
13	Low	High	Low	High	Highly Trustworthy
14	Low	Medium	Low	High	Highly Trustworthy
15	High	High	Low	Medium	Trustworthy
16	Medium	High	High	Medium	Trustworthy

Rule	**IF Response Time**	**AND Availability**	**AND Throughput**	**AND WsRF**	**Then Trust vale**
17	Medium	Medium	High	Medium	Trustworthy
18	Medium	High	Medium	Medium	Trustworthy
19	Medium	Medium	Medium	Medium	Trustworthy
20	Medium	High	Low	Medium	Trustworthy
21	Medium	Medium	Low	Medium	Trustworthy
22	Low	High	High	Medium	Trustworthy
23	Low	Medium	High	Medium	Trustworthy
24	Low	High	Medium	Medium	Trustworthy
25	Low	Medium	Medium	Medium	Trustworthy
26	Low	High	Low	Medium	Trustworthy
27	Low	Medium	Low	Medium	Trustworthy
28	High	Low	High	Low	Untrustworthy
29	High	Medium	Medium	Low	Untrustworthy
30	High	Low	Medium	Low	Untrustworthy
31	High	High	Low	Low	Untrustworthy
32	High	Medium	Low	Low	Untrustworthy
33	High	Low	Low	Low	Untrustworthy
34	Medium	Low	High	Low	Untrustworthy

Rule	**IF Response Time**	**AND Availability**	**AND Through-put**	**AND WsRF**	**Then Trust vale**
35	Medium	High	Medium	Low	Untrustworthy
36	Medium	Medium	Medium	Low	Untrustworthy
37	Medium	Low	Medium	Low	Untrustworthy
38	Medium	High	Low	Low	Untrustworthy
39	Medium	Medium	Low	Low	Untrustworthy
40	Medium	Low	Low	Low	Untrustworthy
41	Low	High	Medium	Low	Untrustworthy
42	Low	Medium	Medium	Low	Untrustworthy
43	Low	Low	Medium	Low	Untrustworthy
44	Low	High	Low	Low	Untrustworthy
45	Low	Medium	Low	Low	Untrustworthy
46	Low	Low	Low	Low	Untrustworthy

Table A.4 : Development of Fuzzy Rules for Trust Value Prediction in Experiment Scenario 4

Rule	**IF Response Time**	**AND Availability**	**AND WsRF**	**Then Trust vale**
1	Low	High	High	Highly Trustworthy

Rule	**IF Response Time**	**AND Availability**	**AND WsRF**	**Then Trust vale**
2	Medium	High	High	Highly Trustworthy
3	High	High	High	Highly Trustworthy
4	Low	Medium	High	Highly Trustworthy
5	Low	High	Medium	Trustworthy
6	Medium	High	Medium	Trustworthy
7	Low	Medium	Medium	Trustworthy
8	High	High	Medium	Trustworthy
9	Medium	Medium	Medium	Trustworthy
10	Medium	High	Low	Untrustworthy
11	Low	High	Low	Untrustworthy
12	High	High	Low	Untrustworthy
13	Low	Medium	Low	Untrustworthy
14	Medium	Medium	Low	Untrustworthy
15	High	Medium	Low	Untrustworthy
16	Low	Low	Low	Untrustworthy
17	Medium	Low	Low	Untrustworthy
18	High	Low	Low	Untrustworthy

Bibliography

[1] Luigi Battezzati. Internet of Things Archives — Internet of Things. *Cisco*, (July 2016):1–45, 2016.

[2] Amir Vahid Dastjerdi and Rajkumar Buyya. Fog Computing: Helping the Internet of Things Realize Its Potential. *Computer*, 49(8):112–116, 2016.

[3] Zaigham Mahmood. *Fog Computing: Concepts, Frameworks and Technologies.* Springer International Publishing AG, Cham, 1st ed. 20 edition, 2018.

[4] Ranesh Kumar Naha, Saurabh Garg, Dimitrios Georgakopoulos, Prem Prakash Jayaraman, Longxiang Gao, Yong Xiang, and Rajiv Ranjan. Fog Computing: Survey of Trends, Architectures, Requirements, and Research Directions. *IEEE Access*, 6:47980–48009, 2018.

[5] Avita Katal Ravi Tomar Susheela Dahiya, Niharika Singh, Tanupriya Choudhury. *Fog Computing: Concepts, Frameworks, and Applications.* CRC Press, Milton, 2022.

[6] Simar Preet Singh, Anand Nayyar, Rajesh Kumar, and Anju Sharma. Fog computing: from architecture to edge computing and big data processing. *Journal of Supercomputing*, 75(4):2070–2105, 2019.

[7] Pengfei Hu, Sahraoui Dhelim, Huansheng Ning, and Tie Qiu. Survey on fog computing: architecture, key technologies, applications and open issues. *Journal of Network and Computer Applications*, 98(September):27–42, 2017.

[8] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *MCC'12 - Proceedings of the 1st ACM Mobile Cloud Computing Workshop*, 2012.

[9] Tom H. Luan, Longxiang Gao, Zhi Li, Yang Xiang, Guiyi Wei, and Limin Sun. Fog Computing: Focusing on Mobile Users at the Edge. pages 1–11, 2015.

[10] Afnan Bukhari, Farookh Khadeer Hussain, and Omar K. Hussain. Fog node discovery and selection: A Systematic literature review. *Future Generation Computer Systems*, 135:114–128, 2022.

[11] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In Hans-W. Gellersen, editor, *Handheld and Ubiquitous Computing*, pages 304–307, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[12] Vasileios Karagiannis, Nitin Desai, Stefan Schulte, and Sasikumar Punnekkat. Addressing the Node Discovery Problem in Fog Computing. In Anton Cervin and Yang Yang, editors, *2nd Workshop on Fog Computing and the IoT (Fog-IoT 2020)*, volume 80 of *OpenAccess Series in Informatics (OASIcs)*, pages 5:1–5:10, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[13] Mostafa Haghi Kashani, Ahmad Ahmadzadeh, and Ebrahim Mahdipour. Load balancing mechanisms in fog computing: A systematic review, 2020.

[14] Eva Marin Tordera, Xavi Masip-Bruin, Jordi Garcia-Alminana, Admela Jukan, Guang-Jie Ren, Jiafeng Zhu, and Josep Farre. What is a Fog Node A Tutorial on Current Concepts towards a Common Definition, 2016.

[15] José Santos, Tim Wauters, Bruno Volckaert, and Filip De Turck. Resource Provisioning in Fog Computing: From Theory to Practice. *Sensors*, 19(10), 2019.

[16] Mostafa Ghobaei-Arani, Alireza Souri, and Ali A. Rahmanian. Resource Management Approaches in Fog Computing: a Comprehensive Review. *Journal of Grid Computing*, 18(1):1–42, 2020.

[17] Qiliang Zhu, Baojiang Si, Feifan Yang, and You Ma. Task offloading decision in fog computing system. *China Communications*, 14(11):59–68, 2017.

[18] Md. Muzakkir Hussain and M.M. Sufyan Beg. Code-v: Multi-hop computation offloading in vehicular fog computing. *Future Generation Computer Systems*, 116:86–102, 2021.

[19] Linh-An Phan, Duc-Thang Nguyen, Meonghun Lee, Dae-Heon Park, and Taehong Kim. Dynamic fog-to-fog offloading in SDN-based fog computing systems. *Future Generation Computer Systems*, 117:486–497, 2021.

[20] Carlos Guerrero, Isaac Lera, and Carlos Juiz. Evaluation and efficiency comparison of evolutionary algorithms for service placement optimization in fog architectures. *Future Generation Computer Systems*, 97:131–144, 2019.

[21] Hemant Kumar Apat, Bibhudatta Sahoo, and Prasenjit Maiti. Service Placement in Fog Computing Environment. In *2018 International Conference on Information Technology (ICIT)*, pages 272–277, 2018.

[22] Ahmad Ali, Mansoor Ahmed, Muhammad Imran, and Hasan Ali Khattak. *Security and Privacy Issues in Fog Computing*, chapter 5, pages 105–137. John Wiley & Sons, Ltd, 2020.

[23] Esubalew Alemneh, Sidi-Mohammed Senouci, Philippe Brunet, and Tesfa

Tegegne. A two-way trust management system for fog computing. *Future Generation Computer Systems*, 106:206–220, 2020.

[24] Junwei Zhang, Deyu Li, and Xiaoqin Fan. A customer-centric trust evaluation model for personalized service selection. *Scientific Programming*, 2018, 2018.

[25] Carla Mouradian, Diala Naboulsi, Sami Yangui, Roch H. Glitho, Monique J. Morrow, and Paul A. Polakos. A comprehensive survey on fog computing: State-of-the-art and research challenges. *IEEE Communications Surveys Tutorials*, 20(1):416–464, 2018.

[26] H. Sabireen and V. Neelanarayanan. A Review on Fog Computing: Architecture, Fog with IoT, Algorithms and Research Challenges. *ICT Express*, 7(2):162–176, 2021.

[27] Shanhe Yi, Cheng Li, and Qun Li. A Survey of Fog Computing: Concepts, Applications and Issues. In *Proceedings of the 2015 Workshop on Mobile Big Data*, page 37–42, New York, NY, USA, 2015. Association for Computing Machinery.

[28] Paolo Bellavista, Javier Berrocal, Antonio Corradi, Sajal K. Das, Luca Foschini, and Alessandro Zanni. A survey on fog computing for the Internet of Things. *Pervasive and Mobile Computing*, 52:71–99, 2019.

[29] Dapeng Lan, Amir Taherkordi, Frank Eliassen, and Geir Horn. A Survey on Fog Programming: Concepts, State-of-the-Art, and Research Challenges. In *Proceedings of the 2nd International Workshop on Distributed Fog Services Design*, DFSD '19, page 1–6, New York, NY, USA, 2019. Association for Computing Machinery.

[30] Mandeep Kaur and Rajni Aron. A Systematic Study of Load Balancing Approaches in the Fog Computing Environment. *J. Supercomput.*, 77(8):9202–9247, Aug 2021.

[31] Ashkan Yousefpour, Caleb Fung, Tam Nguyen, Krishna Kadiyala, Fatemeh Jalali, Amirreza Niakanlahiji, Jian Kong, and Jason P. Jue. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture*, 98:289–330, 2019.

[32] Zahra Makki Nayeri, Toktam Ghafarian, and Bahman Javadi. Application placement in Fog computing with AI approach: Taxonomy and a state of the art survey. *Journal of Network and Computer Applications*, 185:103078, 2021.

[33] Mir Salim Ul Islam, Ashok Kumar, and Yu-Chen Hu. Context-aware scheduling in Fog computing: A survey, taxonomy, challenges and future directions. *Journal of Network and Computer Applications*, 180:103008, 2021.

[34] Zeinab Bakhshi, Guillermo Rodriguez-Navas, and Hans Hansson. Dependable Fog Computing: A Systematic Literature Review. In *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 395–403, 2019.

[35] V. Hurbungs, V. Bassoo, and T. P. Fowdur. Fog and edge computing: concepts, tools and focus areas. *International Journal of Information Technology (Singapore)*, 13(2):511–522, 2021.

[36] S. Thiruchadai Pandeeswari and S. Padmavathi. *Fog Based Architectures for IoT: Survey on Motivations, Challenges and Solution Perspectives*, volume 80. Springer International Publishing, 2020.

[37] Gustavo Caiza, Morelva Saeteros, William Oñate, and Marcelo V. Garcia. Fog computing at industrial level, architecture, latency, energy, and security: A review. *Heliyon*, 6(4):e03706, 2020.

[38] Charith Perera, Yongrui Qin, Julio C. Estrella, Stephan Reiff-Marganiec, and Athanasios V. Vasilakos. Fog Computing for Sustainable Smart Cities: A Survey. *ACM Comput. Surv.*, 50(3), jun 2017.

[39] Tauqeer Khalid, Muhammad Abbas Khan Abbasi, Maria Zuraiz, Abdul Nasir Khan, Mazhar Ali, Raja Wasim Ahmad, Joel J.P.C. Rodrigues, and Mudassar Aslam. A survey on privacy and access control schemes in fog computing. *International Journal of Communication Systems*, 34(2), 2021.

[40] Carlo Puliafito, Enzo Mingozzi, Francesco Longo, Antonio Puliafito, and Omer Rana. Fog Computing for the Internet of Things: A Survey. *ACM Trans. Internet Technol.*, 19(2), apr 2019.

[41] Kazi Masum Sadique, Rahim Rahmani, and Paul Johannesson. Fog Computing for Trust in the Internet of Things (IoT): A Systematic Literature Review. In *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)*, pages 1–6, 2020.

[42] Shaheen Parveen, Pawan Singh, and Deepak Arora. *Fog Computing Research Opportunities and Challenges: A Comprehensive Survey*, volume 121. ”Proceedings of First International Conference on Computing, Communications, and Cyber-Security (IC4S 2019)”, 2020.

[43] Jagdeep Singh, Parminder Singh, and Sukhpal Singh Gill. Fog computing: A taxonomy, systematic review, current trends and research challenges. *Journal of Parallel and Distributed Computing*, 157:56–85, 2021.

[44] Sunday Oyinlola Ogundoyin and Ismaila Adeniyi Kamil. Optimization techniques and applications in fog computing: An exhaustive survey. *Swarm and Evolutionary Computation*, 66:100937, 2021.

[45] Nour Mostafa. Cooperative Fog Communications using A Multi-Level Load Balancing. In *2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 45–51, 2019.

[46] Christos Doulkeridis, Nikos Loutas, and Michalis Vazirgiannis. A System Architecture for Context-Aware Service Discovery. *Electron. Notes Theor. Comput. Sci.*, 146(1):101–116, jan 2006.

[47] Quang Duy La, Mao V. Ngo, Thinh Quang Dinh, Tony Q.S. Quek, and Hyundong Shin. Enabling intelligence in fog computing to achieve energy and latency reduction. *Digital Communications and Networks*, 5(1):3–9, 2019.

[48] E. Chang, T.S. Dillon, and F.K. Hussain. Trust and reputation relationships in service-oriented environments. In *Third International Conference on Information Technology and Applications (ICITA'05)*, pages 4–14 vol.1, 2005.

[49] Barbara Kitchenham, O. Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. Systematic literature reviews in software engineering – A systematic literature review. *Information and Software Technology*, 51(1):7–15, 2009.

[50] Sander Soo, Chii Chang, and Satish Narayana Srirama. Proactive Service Discovery in Fog Computing Using Mobile Ad Hoc Social Network in Proximity. In *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom)*

and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pages 561–566, 2016.

[51] Riccardo Venanzi, Burak Kantarci, Luca Foschini, and Paolo Bellavista. MQTT-Driven Sustainable Node Discovery for Internet of Things-Fog Environments. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6, 2018.

[52] Venanzi Riccardo, Kantarci Burak, Foschini Luca, and Bellavista Paolo. MQTT-Driven Node Discovery for Integrated IoT-Fog Settings Revisited: The Impact of Advertiser Dynamicity. In *2018 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, pages 31–39, 2018.

[53] Z. Rejiba, X. Masip-Bruin, A. Jurnet, E. Marin-Tordera, and G. Ren. F2C-Aware: Enabling Discovery in Wi-Fi-Powered Fog-to-Cloud (F2C) Systems. In *2018 6th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, pages 113–116, Los Alamitos, CA, USA, mar 2018. IEEE Computer Society.

[54] Zeineb Rejiba, Xavier Masip-Bruin, and Eva Marín-Tordera. Analyzing the Deployment Challenges of Beacon Stuffing as a Discovery Enabler in Fog-to-Cloud Systems. In *2018 European Conference on Networks and Communications (EuCNC)*, pages 1–276, 2018.

[55] Zeineb Rejiba, Xavier Masip Bruin, and Eva Marín Tordera. A Beacon-assisted direction-aware scanning scheme for 802.11-based discovery in Fog-to-Cloud systems. In *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1–6, 2018.

[56] Zeineb Rejiba, Xavier Masip Bruin, and Eva Marín Tordera. Towards a

context-aware Wi-Fi-based Fog Node discovery scheme using cellular footprints. In *2018 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 1–6, 2018.

[57] Julien Gedeon, Sebastian Zengerle, Sebastian Alles, Florian Brandherm, and Max Mühlhäuser. Sunstone: Navigating the Way Through the Fog. In *2020 IEEE 4th International Conference on Fog and Edge Computing (ICFEC)*, pages 49–58, 2020.

[58] Riccardo Venanzi, Luca Foschini, Paolo Bellavista, Burak Kantarci, and Cesare Stefanelli. Fog-Driven Context-Aware Architecture for Node Discovery and Energy Saving Strategy for Internet of Things Environments. *IEEE Access*, 7:134173–134186, 2019.

[59] José Santos, Tim Wauters, Bruno Volckaert, and Filip De Turck. Towards Dynamic Fog Resource Provisioning for Smart City Applications. In *2018 14th International Conference on Network and Service Management (CNSM)*, pages 290–294, 2018.

[60] Konstantinos Skiadopoulos, Konstantinos Oikonomou, Markos Avlonitis, Konstantinos Giannakis, Dimitrios Kogias, and Ioannis Stavrakakis. Multiple and replicated random walkers analysis for service discovery in fog computing IoT environments. *Ad Hoc Networks*, 93:101893, 2019.

[61] Nitendra Tomar and Rakesh Matam. Optimal Query-Processing-Node Discovery in IoT-Fog Computing Environment. In *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 237–241, 2018.

[62] Saša Pešić, Milenko Tošić, Ognjen Iković, Miloš Radovanović, Mirjana Ivanović, and Dragan Bošković. Bluetooth Low Energy Microlocation Asset

Tracking (BLEMAT) in a Context-Aware Fog Computing System. In *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics*, WIMS '18, New York, NY, USA, 2018. Association for Computing Machinery.

[63] Dominic Henze, Paul Schmiedmayer, and Bernd Bruegge. Fog Horizons – A Theoretical Concept to Enable Dynamic Fog Architectures. In *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*, UCC'19, page 41–50, New York, NY, USA, 2019. Association for Computing Machinery.

[64] Redowan Mahmud, Kotagiri Ramamohanarao, and Rajkumar Buyya. Latency-Aware Application Module Management for Fog Computing Environments. *ACM Trans. Internet Technol.*, 19(1), nov 2018.

[65] Biji Nair and Mary Saira Bhanu Somasundaram. Overload prediction and avoidance for maintaining optimal working condition in a fog node. *Computers & Electrical Engineering*, 77:147–162, 2019.

[66] Issam Jabri, Tesnim Mekki, Abderrezak Rachedi, and Maher Ben Jemaa. Vehicular fog gateways selection on the internet of vehicles: A fuzzy logic with ant colony optimization based approach. *Ad Hoc Networks*, 91:101879, 2019.

[67] Manoj Kumar Mishra, Niranjan Kumar Ray, Amulya Ratna Swain, Ganga Bishnu Mund, and Bhabani Sankar Prasad Mishra. An adaptive model for resource selection and allocation in fog computing environment. *Computers & Electrical Engineering*, 77:217–229, 2019.

[68] Zeineb Rejiba, Xavier Masip-Bruin, and Eva Marín-Tordera. Towards user-centric, switching cost-aware fog node selection strategies. *Future*

Generation Computer Systems, 117:359–368, 2021.

[69] Hani Sami and Azzam Mourad. Dynamic On-Demand Fog Formation Offering On-the-Fly IoT Service Deployment. *IEEE Transactions on Network and Service Management*, 17(2):1026–1039, 2020.

[70] Jie Pan, Yiwen Zhang, Qingren Wang, Dengcheng Yan, and Wenming Zhang. A Novel Fog Node Aggregation Approach for Users in Fog Computing Environment. In *2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, pages 160–167, 2020.

[71] Yuxiao Li, Yazhong Zhang, Yuxiang Liu, Qingmin Meng, and Feng Tian. Fog Node Selection for Low Latency Communication and Anomaly Detection in Fog Networks. In *2019 International Conference on Communications, Information System and Computer Engineering (CISCE)*, pages 276–279, 2019.

[72] Niloy Irtisam, Riad Ahmed, Mohammad Moniruzzaman Akash, Raiyaan Abdullah, Sujan Sarker, Sejuti Rahman, and Lafifa Jamal. Pathfinder: A fog assisted vision-based system for optimal path selection of service robots. In *2020 Joint 9th International Conference on Informatics, Electronics Vision (ICIEV) and 2020 4th International Conference on Imaging, Vision Pattern Recognition (icIVPR)*, pages 1–6, 2020.

[73] Isaac Lera, Carlos Guerrero, and Carlos Juiz. Analyzing the Applicability of a Multi-Criteria Decision Method in Fog Computing Placement Problem. In *2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC)*, pages 13–20, 2019.

[74] Duong Tung Nguyen, Long Bao Le, and Vijay K. Bhargava. A Market-Based Framework for Multi-Resource Allocation in Fog Computing. *IEEE/ACM Transactions on Networking*, 27(3):1151–1164, 2019.

[75] Ganesh Neelakanta Iyer, Vishal Raman, K.S. Aswin, and Bharadwaj Veeravalli. On the strategies for risk aware cloud and fog broker arbitrage mechanisms. In *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, pages 794–799, 2020.

[76] Zexuan Yang, Li Wang, Yinan Ding, and Mei Song. Adverse Selection via Matching in Cooperative Fog Computing. In *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, pages 1–5, 2018.

[77] Taj Rahman, Xuanxia Yao, Gang Tao, Huansheng Ning, and Zhangbing Zhou. Efficient Edge Nodes Reconfiguration and Selection for the Internet of Things. *IEEE Sensors Journal*, 19(12):4672–4679, 2019.

[78] Gaurav Baranwal and Deo Prakash Vidyarthi. FONS: A fog orchestrator node selection model to improve application placement in fog computing. *J. Supercomput.*, 77(9):10562–10589, sep 2021.

[79] Tingting Yang, Rui Wang, Zhengqi Cui, Jie Dong, and Minghua Xia. Multi-attribute selection of maritime heterogenous networks based on SDN and fog computing architecture. In *2018 16th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, pages 1–6, 2018.

[80] Karima Velasquez, David Perez Abreu, Luís Paquete, Marilia Curado, and Edmundo Monteiro. A Rank-based Mechanism for Service Placement in the Fog. In *2020 IFIP Networking Conference (Networking)*, pages 64–72, 2020.

[81] Ramon R. Fontes, Samira Afzal, Samuel H. B. Brito, Mateus A. S. Santos, and Christian Esteve Rothenberg. Mininet-WiFi: Emulating software-defined wireless networks. In *2015 11th International Conference on Network and Service Management (CNSM)*, pages 384–389, 2015.

[82] Ingmar Baumgart, Bernhard Heep, and Stephan Krause. OverSim: A flexible overlay network simulation framework. In *2007 IEEE Global Internet Symposium*, pages 79–84, 2007.

[83] NS-3. NS-3 is a discrete-event network simulator.

[84] Juan Rada-Vilela. The fuzzylite libraries for fuzzy logic control, 2018.

[85] Rajkumar Buyya and Manzur Murshed. GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and Computation: Practice and Experience*, 14(13-15):1175–1220, 2002.

[86] Giuseppe La Torre, Marco Cavallo, Valeria D'Amico, Salvatore Monteleone, and Vincenzo Catania. A context-aware solution to improve web service discovery and user-service interaction. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, pages 180–187, 2016.

[87] Hamed Vahdat Nejad, Chintan Bhatt, Arezoo Tavakolifar, Nooshin Hanafi, Nahid Gholizadeh, Reza Khatooni, and Hossein Behzadian. A survey on context-aware fog computing systems. *Computacion y Sistemas*, 25(1):5–12, 2021.

[88] Feng Zhu, M. Mutka, and L. Ni. Splendor: A secure, private, and location-aware service discovery protocol supporting mobile services. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003. (PerCom 2003).*, pages 235–242, 2003.

[89] Chen Yu, Dezhong Yao, Xi Li, Yan Zhang, Laurence T. Yang, Naixue Xiong, and Hai Jin. Location-Aware Private Service Discovery in Pervasive Computing Environment. *Inf. Sci.*, 230:78–93, may 2013.

[90] Yasir Arfat Malkani and Lachhman Das Dhomeja. Location aware device discovery for physically constrained environments. In *2009 2nd International Conference on Computer, Control and Communication*, pages 1–5, 2009.

[91] Dunlu Peng, Xinglong Jiang, Wenjie Xu, and Kai Duan. WSCache: A cache based content-aware approach of web service discovery. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 6318 LNCS, pages 394–401, 2010.

[92] Kirak Hong, David Lillethun, Umakishore Ramachandran, Beate Ottenwälder, and Boris Koldehofe. Mobile Fog: A Programming Model for Large-Scale Applications on the Internet of Things. In *Proceedings of the Second ACM SIGCOMM Workshop on Mobile Cloud Computing*, MCC '13, page 15–20, New York, NY, USA, 2013. Association for Computing Machinery.

[93] Sami Yangui, Pradeep Ravindran, Ons Bibani, Roch H. Glitho, Nejib Ben Hadj-Alouane, Monique J. Morrow, and Paul A. Polakos. A platform as-a-service for hybrid cloud/fog environments. In *2016 IEEE International*

Symposium on Local and Metropolitan Area Networks (LANMAN), pages 1–7, 2016.

[94] Nam Ky Giang, Michael Blackstock, Rodger Lea, and Victor C.M. Leung. Developing IoT applications in the Fog: A Distributed Dataflow approach. In *2015 5th International Conference on the Internet of Things (IOT)*, pages 155–162, 2015.

[95] Holger Billhardt, Ramón Hermoso, Sascha Ossowski, and Roberto Centeno. Trust-Based Service Provider Selection in Open Environments. In *Proceedings of the 2007 ACM Symposium on Applied Computing*, SAC '07, page 1375–1380, New York, NY, USA, 2007. Association for Computing Machinery.

[96] Jun Li, Xiaolin Zheng, Deren Chen, and William Wei Song. Trust Based Service Selection in Service Oriented Environment. *Int. J. Web Serv. Res.*, 9(3):23–42, jul 2012.

[97] Mingdong Tang, Xiaoling Dai, Jianxun Liu, and Jinjun Chen. Towards a Trust Evaluation Middleware for Cloud Service Selection. *Future Gener. Comput. Syst.*, 74(C):302–312, sep 2017.

[98] Xiaohui Li, Hongxing Liang, and Xing Zhang. Trust based service selection in cloud computing environment. *International Journal of Smart Home*, 10(11), 2016.

[99] Yuchen Pan, Shuai Ding, Wenjuan Fan, Jing Li, and Shanlin Yang. Trust-enhanced cloud service selection model based on QoS analysis. *PLoS ONE*, 10(11), 2015.

[100] Xing Su, Minjie Zhang, and Yi Mu. Trust-Based Group Services Selection in Web-Based Service-Oriented Environments. *World Wide Web*,

19(5):807–832, sep 2016.

[101] Okba Tibermacine, Chouki Tibermacine, and Foudil Cherif. Regression-Based Bootstrapping of Web Service Reputation Measurement. In *2015 IEEE International Conference on Web Services*, pages 377–384, 2015.

[102] Jiwan Seo, Seungjin Choi, and Sangyong Han. The Method of Trust and Reputation Systems Based on Link Prediction and Clustering. In M. Carmen Fernández Gago, Fabio Martinelli, Siani Pearson, and Isaac Agudo, editors, *Trust Management VII - 7th IFIP WG 11.11 International Conference, IFIPTM 2013, Malaga, Spain, June 3-7, 2013. Proceedings*, volume 401 of *IFIP Advances in Information and Communication Technology*, pages 223–230. Springer, 2013.

[103] Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. Facing the cold start problem in recommender systems. *Expert Systems with Applications*, 41(4, Part 2):2065–2073, 2014.

[104] Ari Keränen, Jörg Ott, and Teemu Kärkkäinen. The ONE Simulator for DTN Protocol Evaluation. In *Proceedings of the 2nd International Conference on Simulation Tools and Techniques*, Simutools '09, Brussels, BEL, 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[105] Eli Bressert. Scipy and numpy: an overview for developers. 2012.

[106] Andras Markus and Attila Kertesz. A survey and taxonomy of simulation environments modelling fog computing. *Simulation Modelling Practice and Theory*, 101:102042, 2020. Modeling and Simulation of Fog Computing.

[107] Massimo Villari, Maria Fazio, Schahram Dustdar, Omer Rana, and Rajiv Ranjan. Osmotic computing: A new paradigm for edge/cloud integration.

IEEE Cloud Computing, 3(6):76–83, 2016.

[108] Chintan Bhatt and C. K. Bhensdadia. Fog Computing: Applications, Concepts, and Issues. *Int. J. Grid High Perform. Comput.*, 9(4):105–113, oct 2017.

[109] Ken Peffers, Tuure Tuunanen, Marcus Rothenberger, and Samir Chatterjee. A Design Science Research Methodology for Information Systems Research. *J. Manage. Inf. Syst.*, 24(3):45–77, dec 2007.

[110] Afnan Abdulrahman Bukhari, Farookh Khadeer Hussain, and Omar Khadeer Hussain. Intelligent context-aware fog node discovery. *Internet of Things*, 20(August):100607, 2022.

[111] OMNeT++. OMNeT++ Network Simulation Framework, 2013.

[112] Phu Lai, Qiang He, Mohamed Abdelrazek, Feifei Chen, John Hosking, John Grundy, and Yun Yang. Optimal edge user allocation in edge computing with variable sized vector bin packing. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2018.

[113] T. M. Cover and P. E. Hart. Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory*, 1967.

[114] Mohammad RezaAbbasifard, Bijan Ghahremani, and Hassan Naderi. A Survey on Nearest Neighbor Search Methods. *International Journal of Computer Applications*, 95(25):39–52, 2014.

[115] Olof Enqvist, Fangyuan Jiang, and Fredrik Kahl. A brute-force algorithm for reconstructing a scene from two projections. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2961–2968, 2011.

[116] Vincent Garcia, Eric Debreuve, and Michel Barlaud. Fast k nearest neighbor search using GPU. *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops*, pages 1–6, 2008.

[117] T. F.Michael Raj, P. Sivapragasam, R. Balakrishnan, G. Lalithambal, and S. Ragasubha. QoS based classification using K-Nearest Neighbor algorithm for effective web service selection. *Proceedings of 2015 IEEE International Conference on Electrical, Computer and Communication Technologies, ICECCT 2015*, pages 14–17, 2015.

[118] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998.

[119] Wei Zhang, Xiaohui Chen, Yueqi Liu, and Qian Xi. A Distributed Storage and Computation k-Nearest Neighbor Algorithm Based Cloud-Edge Computing for Cyber-Physical-Social Systems. *IEEE Access*, 8:50118–50130, 2020.

[120] Jon Louis Bentley. Multidimensional Binary Search Trees Used for Associative Searching. *Communications of the ACM*, 18(9):509–517, 1975.

[121] Ingo Wald and Vlastimil Havran. On building fast kd-trees for ray tracing, and on doing that in O(N log N). *RT'06: IEEE Symposium on Interactive Ray Tracing 2006, Proceedings*, pages 61–69, 2006.

[122] Abedalmuhdi Almomany, Walaa R. Ayyad, and Amin Jarrah. Optimized implementation of an improved KNN classification algorithm using Intel FPGA platform: Covid-19 case study. *Journal of King Saud University - Computer and Information Sciences*, 34(6):3815–3827, 2022.

[123] Zhongheng Zhang. Introduction to machine learning: K-nearest neighbors. *Annals of Translational Medicine*, 2016.

[124] Xing Gao and Guilin Li. A KNN Model Based on Manhattan Distance to Identify the SNARE Proteins. *IEEE Access*, 8:112922–112931, 2020.

[125] Https://dataaspirant.com/2016/12/23/k nearest-neighbor-classifier intro/. No Title.

[126] Mohammed Otair. Approximate K-Nearest Neighbour Based Spatial Clustering Using K-D Tree. *International Journal of Database Management Systems*, 5(1):97–108, 2013.

[127] Martin Skrodzki. The k-d tree data structure and a proof for neighborhood computation in expected logarithmic time. pages 1–12, 2019.

[128] G. Varoquaux, L. Buitinck, G. Louppe, O. Grisel, F. Pedregosa, and A. Mueller. Scikit-learn. *GetMobile: Mobile Computing and Communications*, 19(1):29–33, 2015.

[129] Kurnia Saputra, Nazaruddin Nazaruddin, Dalila Husna Yunardi, and Renny Andriyani. Implementation of haversine formula on location based mobile application in syiah kuala university. *Proceedings: CYBERNETICSCOM 2019 - 2019 IEEE International Conference on Cybernetics and Computational Intelligence: Towards a Smart and Human-Centered Cyber World*, pages 40–45, 2019.

[130] Ahmed Shamsul Arefin, Carlos Riveros, Regina Berretta, and Pablo Moscato. GPU-FS-kNN: A Software Tool for Fast and Scalable kNN Computation Using GPUs. *PLoS ONE*, 7(8), 2012.

[131] Eyhab Al-Masri and Qusay H. Mahmoud. Investigating web services on the

world wide web. *Proceeding of the 17th International Conference on World Wide Web 2008, WWW'08*, pages 795–804, 2008.

[132] Sunday Oyinlola Ogundoyin and Ismaila Adeniyi Kamil. A trust management system for fog computing services. *Internet of Things (Netherlands)*, 14:100382, 2021.

[133] Rajganesh Nagarajan, S. Selvamuthukumaran, and Ramkumar Thirunavukarasu. A fuzzy logic based trust evaluation model for the selection of cloud services. *2017 International Conference on Computer Communication and Informatics, ICCCI 2017*, pages 3–7, 2017.

[134] G. Priya and N. Jaisankar. A fuzzy based trust evaluation model for service selection in cloud environment. *International Journal of Grid and High Performance Computing*, 11(4):13–27, 2019.

[135] Sunil Kumar, Sumit Mittal, and Manpreet Singh. Fuzzy Based Trust Management System for Cloud Environment. *Advances in Science and Technology Research Journal*, 10(30):32–37, 2016.

[136] Alagumani Selvaraj and Subashini Sundararajan. Evidence-Based Trust Evaluation System for Cloud Services Using Fuzzy Logic. *International Journal of Fuzzy Systems*, 19(2):329–337, 2017.

[137] Supriya M, Venkataramana L.J, K Sangeeta, and G K Patra. Estimating Trust Value for Cloud Service Providers using Fuzzy Logic. *International Journal of Computer Applications*, 48(19):28–34, 2012.

[138] Surya Nepal, Wanita Sherchan, Jonathon Hunklinger, and Athman Bouguettaya. A fuzzy trust management framework for Service Web. *ICWS 2010 - 2010 IEEE 8th International Conference on Web Services*, pages 321–328, 2010.

[139] Zohra Saoud, Noura Faci, Zakaria Maamar, and Djamal Benslimane. A fuzzy-based credibility model to assess Web services trust under uncertainty. *Journal of Systems and Software*, 122:496–506, 2016.

[140] Hossein Shirgahi, Mehran Mohsenzadeh, and Hamid Haj Seyyed Javadi. A three level fuzzy system for evaluating the trust of single web services. *Journal of Intelligent and Fuzzy Systems*, 32(1):589–611, 2017.

[141] Ebrahim H. Mamdani. Application of Fuzzy Logic to Approximate Reasoning Using Linguistic Synthesis. *IEEE Transactions on Computers*, C-26(12):1182–1191, 1977.

[142] MATLAB. *Version 9.10.0 (R2021a).* The MathWorks Inc., Natick, Massachusett, 2021.

[143] R. Ramesh Feslin Anish Mon Solomon, Godfrey Winster Sathianesan. Logistic Regression Trust–A Trust Model for Internet-of-Things Using Regression Analysis. *Computer Systems Science and Engineering*, 44(2):1125–1142, 2023.

[144] K Prathapchandran and T Janani. A Trust-Based Security Model to Detect Misbehaving Nodes in Internet of Things (IoT) Environment using Logistic Regression. *Journal of Physics: Conference Series*, 1850(1):012031, may 2021.

[145] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[146] H Taud and J F Mas. *Multilayer Perceptron (MLP)*, pages 451–455. Springer International Publishing, Cham, 2018.

[147] Ehsan Khadangi and Alireza Bagheri. Comparing MLP, SVM and KNN for predicting trust between users in Facebook. *Proceedings of the 3rd International Conference on Computer and Knowledge Engineering, ICCKE 2013*, (Iccke):466–470, 2013.

[148] Leo Carlsson. IN DEGREE PROJECT MATERIALS DESIGN AND ENGINEERING 300 , SECOND CYCLE CREDITS , STOCKHOLM SWEDEN 2015 Using Multilayer Perceptrons as means to predict the end-point temperature in an Electric Arc Furnace. 2015.

[149] Barbara Hammer. Neural Smithing – Supervised Learning in Feedforward Artificial Neural Networks, 2001.

[150] Fred L. Van Rossum, Guido and Drake. *Python 3*. CreateSpace, Scotts Valley, CA, 2009.

[151] J Brownlee and Machine Learning Mastery. *Deep Learning with Python: Develop Deep Learning Models on Theano and TensorFlow Using Keras*. Machine Learning Mastery, 2017.

[152] Pasi Luukka. Feature selection using fuzzy entropy measures with similarity classifier. *Expert Systems with Applications*, 38(4):4600–4607, 2011.

[153] Zhen Chen, Limin Shen, Feng Li, and Dianlong You. Your neighbors alleviate cold-start: On geographical neighborhood influence to collaborative web service QoS prediction. *Knowledge-Based Systems*, 138:188–201, 2017.

[154] Zibin Zheng, Yilei Zhang, and Michael R. Lyu. Investigating QoS of real-world web services. *IEEE Transactions on Services Computing*, 7(1):32–39, 2014.

[155] Marin Silic, Goran Delac, and Sinisa Srbljic. Prediction of atomic web services reliability based on K-means clustering. *2013 9th Joint Meeting of*

the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2013 - Proceedings, pages 70–80, 2013.

[156] D L Davies and D W Bouldin. A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(2):224–227, 1979.

[157] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(C):53–65, 1987.

[158] The scikit-image Team. The scikit-fuzzy Documentation. 2016.

[159] Abdulhamit Subasi. *Practical machine learning for data analysis using python* . Academic Press, London, England, 2020.

[160] François Chollet and Others. Keras. \url{https://github.com/keras-team/keras}, 2015.