A Fog Computing Approach for Cognitive, Reliable and Trusted Distributed Systems

Mohammed Al-khafajiy

A thesis submitted in partial fulfilment of the requirements of
Liverpool John Moores University
for the degree of
Doctor of Philosophy

February 2020

Abstract

In the Internet of Things era, a big volume of data is generated/gathered every second from billions of connected devices. The current network paradigm, which relies on centralised data centres (a.k.a. Cloud computing), becomes an impractical solution for IoT data storing and processing due to the long distance between the data source (e.g., sensors) and designated data centres. It worth noting that the *long distance* in this context refers to the physical path and time interval of when data is generated and when it get processed. To explain more, by the time the data reaches a far data centre, the importance of the data can be depreciated. Therefore, the network topologies have evolved to permit data processing and storage at the edge of the network, introducing what so-called *fog Computing*. The later will obviously lead to improvements in quality of service via processing and responding quickly and efficiently to varieties of data processing requests.

Although fog computing is recognized as a promising computing paradigm, it suffers from challenging issues that involve: i) concrete adoption and management of fogs for decentralized data processing. ii) resources allocation in both cloud and fog layers. iii) having a sustainable performance since fog have a limited capacity in comparison with cloud. iv) having a secure and trusted networking environment for fogs to share resources and exchange data securely and efficiently. Hence, the thesis focus is on having a stable performance for fog nodes by enhancing resources management and allocation, along with safety procedures, to aid the IoT-services delivery and cloud computing in the ever growing industry of smart things. The main aspects related to the performance stability of fog computing involves the development of cognitive fog nodes that aim at provide fast and reliable services, efficient resources managements, and trusted networking, and hence ensure the best Quality of Experience, Quality of Service and Quality of Protection to end-users.

Therefore the contribution of this thesis in brief is a novel Fog Resource manAgeMEnt Scheme (FRAMES) which has been proposed to crystallise fog distribution and resource management with an appropriate service's loads distribution and allocation based on the $\mathcal{F}$og-2-$\mathcal{F}$og coordination. Also, a novel COMputIng Trust manageMENT (COMITMENT) which is a software-based approach that is responsible for providing a secure and trusted environment for fog nodes to share their resources and exchange data packets. Both FRAMES and COMITMENT are encapsulated in the proposed Cognitive Fog (CF) computing which aims at making fog able to not only act on the data but also interpret the gathered data in a way that mimics the process of cognition in the human mind. Hence, FRAMES provide CF with elastic resource managements for load balancing and resolving congestion, while the COMITMENT employ trust and recommendations models to avoid malicious fog nodes in the $\mathcal{F}$og-2-$\mathcal{F}$og coordination environment.

The proposed algorithms for FRAMES and COMITMENT have outperformed the competitive benchmark algorithms, namely Random Walks Offloading (RWO) and Nearest Fog Offloading (NFO) in the experiments to verify the validity and performance. The experiments were conducted on the performance (in terms of latency), load balancing among fog nodes and fogs trustworthiness along with detecting malicious events and attacks in the $\mathcal{F}$og-2-$\mathcal{F}$og environment. The performance of the proposed FRAMES's offloading algorithms has the lowest run-time (i.e., latency) against the benchmark algorithms (RWO and NFO) for processing equal-number of packets. Also, COMITMENT's algorithms were able to detect the collaboration requests whether they are secure, malicious or anonymous. The proposed work shows potential in achieving a sustainable fog networking paradigm and highlights significant benefits of fog computing in the computing ecosystem.

List of Publications

This thesis research has been carried out at The Department of Computer Science, Liverpool John Moores University. The main contributions of this thesis are discussed in Chapters 3-6 which are based on the following publications:

Selected Journals Paper

1. **Mohammed Al-khafajiy**, Thar Baker, Hilal Al-Libawy, Zakaria Maamar, Moayad and Yaser Jararweh. "Improving fog computing performance via fog-2-fog collaboration." Future Generation Computer Systems (2019). **(Q1)-(IF:5.7)**.
2. **Mohammed Al-khafajiy**, Thar Baker, Muhammad Asim, Zehua Guo, Rajiv Ranjan, Antonella Longo, Deepak Puthal, and Mark Taylor. "COMITMENT: A Fog Computing Trust Management Approach." Journal of Parallel and Distributed Computing (2020). **(Q2)-(IF:1.8)**.
3. **Mohammed Al-khafajiy**, Thar Baker, Carl Chalmers, Muhammad Asim, Hoshang Kolivand, Muhammad, and Atif Waraich. "Remote health monitoring of elderly through wearable sensors." Multimedia Tools and Applications (2019). **(Q1)-(IF:2.1)**
4. **Mohammed Al-khafajiy**, Hoshang Kolivand, Thar Baker, David Tully, and Atif Waraich. "Smart Hospital Emergency System Via Mobile-based Requesting Services." Multimedia Tools and Applications (2019). **(Q1)-(IF:2.1)**
5. Zakaria Maamar, Thar Baker, Noura Faci, **Mohammed Al-Khafajiy**, Emir, Yacine and Mohamed Sellami. "Weaving cognition into the internet-of-things: Application to water leaks." Cognitive Systems Research (2019). **(Q3)-(IF:1.4)**.

6. Boukadi, Khouloud, Noura Faci, Zakaria Maamar, Emir Ugljanin, Mohamed Sellami, Thar Baker, and **Mohammed Al-Khafajiy**. "Norm-based and commitment driven agentification of the Internet of Things." Internet of Things 6 (2019).
7. **Mohammed Al-khafajiy**, Safa Otoum, Thar Baker, Muhammad Asim, Zakaria Maamar, Moayad Aloqaily, Mark Taylor, Martin Randles "Intelligent Control and Security of Fog Resources in Healthcare Systems via a Cognitive Fog Model" ACM Transactions on Internet Technology. [**Accepted**] **(Q2)-(IF:3.7)**.

Selected Conferences Papers

1. Al-Khafajiy, Mohammed, Thar Baker, Atif Waraich, Dhiya Al-Jumeily, and Abir Hussain. "IoT-Fog Optimal Workload via Fog Offloading." In 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion), pp. 359-364. IEEE, 2018.
2. Al-Khafajiy, Mohammed, Thar Baker, Hilal Al-Libawy, Atif Waraich, Carl Chalmers, and Omar Alfandi. "Fog Computing Framework for Internet of Things Applications." In 2018 11th International Conference on Developments in eSystems Engineering (DeSE), pp. 71-77. IEEE, 2018.
3. Al-Khafajiy, Mohammed, Lee Webster, Thar Baker, and Atif Waraich. "Towards fog driven IoT healthcare: challenges and framework of fog computing in healthcare." In Proceedings of the 2nd International Conference on Future Networks and Distributed Systems, p. 9. ACM, 2018.
4. Maamar, Zakaria, Thar Baker, Noura Faci, Emir Ugljanin, Mohammed Al Khafajiy, and Vanilson Burégio. "Towards a seamless coordination of cloud and fog: illustration through the internet-of-things." In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, pp. 2008-2015. ACM, 2019.
5. Maamar, Zakaria, Thar Baker, Noura Faci, Emir Ugljanin, Yacine Atif, Mohammed Al-Khafajiy, and Mohamed Sellami. "Cognitive computing meets the internet of things." In 13th International Conference on Software Technologies, ICSOFT, Porto, Portugal, July 26-28, 2018, pp. 741-746. SciTePress, 2018.
6. Maamar, Zakaria, Khouloud Boukadi, Emir Ugljanin, Thar Baker, Muhammad Asim, Mohammed Al-Khafajiy, Djamal Benslimane, and Hasna El Alaoui El Abdallaoui.

"Thing Federation as a Service: Foundations and Demonstration." In International Conference on Model and Data Engineering, pp. 184-197. Springer, Cham, 2018.

7. **Mohammed Al-Khafajiy**, Thar Baker, Atif Waraich, Omar Alfandi, and Aseel Hussien, "Enabling high performance fog computing through fog-2-fog coordination model," in 2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA), pp. 1–6, IEEE, 2019.
8. **Mohammed Al-Khafajiy**, Shatha Ghareeb, Rawaa Al-Jumeily, Rusul Almurshedi, Aseel Hussien, Thar Baker, and Yaser Jararweh, "A holisticstudy on emerging iot networking paradigms," in2019 12th International Conference on Developmentsin eSystems Engineering (DeSE), pp. 943–949, IEEE, 2019.

Book Chapter

1. **Mohammed Al-Khafajiy**, Thar Baker, Aseel Hussien, and Alison Cotgrave, "UAV and Fog Computing for IoE-Based Systems: A Case Study on Environment Disasters Prediction and Recovery Plans," in Unmanned Aerial Vehicles in Smart Cities, pp. 133–152, Springer, 2020.

Declaration

The work presented in this thesis was carried out at the Department of Computer Science, Liverpool John Moores University. Unless otherwise stated, it is the original work of the author. While registered as a candidate for the degree of Doctor of Philosophy, for which submission is now made, the author has not been registered as a candidate for any other award. This thesis has not been submitted in whole, or in part, for any other degree.

Mohammed AL-khafajiy
Department of Computer Science
Liverpool John Moores University
3 Byrom St,
Liverpool
L3 3AF
UK

July 1, 2020

Declaration of Authorship

I, Mohammed AL-khafajiy, declare that this thesis titled, 'A Fog Computing Approach for Cognitive, Reliable and Trusted Distributed Systems' and the work presented in it are my own. I confirm that:

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

Signed:

Date: July 1, 2020

Acknowledgements

PhD is tough, but rewarding journey that a person can experience once in a lifetime. I am truly happy that I have overcome the adversities and finally approached near to the end of this journey, my enormous gratitude and thanks go to the Almighty God for every blessing. The PhD journey would not have happened without endless help from people around me.

First and foremost, I am perpetually indebted in thanks and appreciation to my supervisor **Dr. Thar Baker** for his dedicated supervision, helpful and endless support, patience and motivation. His guidance has provided me with great help throughout my PhD and in every aspect of my research work. His kindness and care have encouraged me to stay on track. This work would never have been completed without his support, warm welcome and dedication. His kindness, prudence and work ethic have made me enjoy this journey.

Gratitude and sincere appreciation to my family for their continuous support. I would not have been able to achieve my dreams without thier sacrifice, patience and kindness. I am especially grateful to **my parents (Dhirgham and Ashwaq)** to whom I owe where I am today. They have supported me emotionally and financially, thank you both for giving me the strength to reach for the stars and chase my dream. Immense thanks beyond measure to my wonderful **wife (Rusul)** for her love and constant support, for all the late nights and early mornings, and for keeping me sane over the PhD journey. Thank you for being my muse, my best friend and great companion, who encouraged, and helped me get through this agonizing period in the most positive way. I also want to take the opportunity to thank my **brothers (Hassan and Ali)** and **sister (Mariam)** for their support and encouragement throughout.

Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Introduction

A question of need is a question of taste.

Neil Tennant & Chris Lowe

1.1 Introduction

In the last few years, major advances in Information and Communication Technologies (ICT) have been witnessed. Such advances are anchored to different paradigms, such as Information Centric Network (ICN) (e.g., mainframe), the well known Software Defined Network (SDN), and Data Centre Network (DCN). ICN shifted the inter-networking to a cloud-based computing model, as reported in Cisco Cloud Index (2013-2018) [1]. Since most Internet traffic originates from and/or terminates in the cloud [1, 2, 3, 4, 5, 6], it is predicted that nearly two-thirds of total workloads obtained from traditional IT services (e.g., data aggregation and processing) will be processed on the cloud [1, 7]. Cloud computing enables users to access a variety of configurable facilities such as data storage, processing, infrastructure, and applications [4], providing Everything-as-a-Service (*aaS) in return for a fee. Embracing the clouds, Internet service providers and corporate IT service providers have become more motivated to adopt cloud computing; they can obtain a wide range of services with minimal administration [4, 8]. The inclination to use the clouds coincides with the improvement of the Network Function Virtualisation (NFV) technique which reduces cloud CAPital EXPenditures (CAPEX) and OPerating EXpenditures (OPEX), and improves the flexibility and scalability of an entire network [9].

Within the emerging Internet of Things (IoT), a large number of "smart" devices and objects (e.g., wearable) are, nowadays, connected to the Internet [10], generating high volumes of data every second. In the IoT area, the key word *"thing"* could be everything, referring to anything that can connect to a network and exchange data over this network [11] with other stakeholders (e.g., users, applications, and peers). Cisco Internet Business Solutions Group (IBSG) estimates that approximately 50 billion devices (i.e., things) will be connected to IoT networks by 2022 [12, 13], and thus served by the cloud. Although the cloud can provision efficient data storage and processing facilities, the ever-growing volume of data will result in a burden on the communication bandwidth [1, 2], and "unacceptable" latency [14, 4, 15]. In addition, since the cloud is relatively "far" from IoT things, by the time the data reaches the cloud for storage and/or processing, its importance and freshness could depreciate [16, 17]. It worth noting that we define *service-latency* (to be used in Chapter 5 & 6) as the total time required to deliver a service, which includes the time when

an IoT thing sends a service request, and when it receives the response back including the travelling interval time. To address cloud limitations with a focus on latency in IoT, Cisco came up with the concept of Fog Computing in 2014 [1].

Fog Computing is described as a highly virtualised platform that provides similar cloud facilities, in terms of storage, processing, and communications, but at the edge of the network, "closer" to things compared to the cloud; i.e., between things and clouds [18], intended to allow fast, secure, and reliable services [19, 16]. Fog is not a substitute to cloud but a complements it [1] since both are expected to work together [20, 21]. In general, the fog can support, serve, and facilitate services that are not appropriately served by cloud such as, (i) latency-sensitive services (e.g., healthcare monitoring and online gaming) [14]; (ii) geo-distributed services (e.g., pipeline monitoring) [22]; (iii) mobile services with high speed connectivity (e.g., connected vehicles) [3]; and (iv) large scale distributed control systems (e.g., smart energy distribution and smart traffic lights) [1]. In fog computing, fog-based services are generally owned by different parties for various reasons: (i) the deployment choice that may include the selection of Internet service providers or wireless carriers, (ii) businesses extending their existing cloud-based services to the edge for performance improvement, and (iii) offering spare resources on the local private cloud as fog services to local businesses on lease [23]. The fog-based services can be provided and managed by different providers, which means that the services can be either attached to with services Internet service provider or by independent stockholders such as FogHorn[1]. This flexibility of offering different fog-based services by different providers complicates the performance reliability and trust of service providers in fog computing. Moreover, the main recurring challenging issues of fog computing includes node's heterogeneity that poses substantial designing challenge, in network design, to ensure reliable services throughout, also the trust management among these fog nodes that work at the network edge can poses threats such as Denial-of-Service (DoS) attack, and ultimately resource management and allocation that poses additional provocation due to the limited computing and storage resources available in the fog nodes compared to the cloud.

[1]FogHorn provide edge intelligence solution in the market today, it delivers comprehensive data enrichment and real-time analytics on high volumes, varieties and velocities of streaming sensor data. Can be found at www.foghorn.io

1.2 Motivation: *cognitive, reliable and trusted fog*

Most challenging issues that the fog computing addresses can be associated with Quality of Service (QoS) [24] (e.g., reduce latency and serve large number of users), Quality of Experience (QoE) (e.g., service experience) [3], along with the Quality of Protection (QoP) (e.g., service security and privacy) [25]. Fog can provide elastic resources to large scale processing systems, thus it can work independently (i.e., fog process all requests) and/or federated with cloud (i.e., fog *and* cloud share the processes of requests) [26, 2, 14]. When fogs works independently, we refer to the ($\mathcal{F}$og-2-$\mathcal{F}$og) cooperation as fog-to-fog coordination, while, when fog shares processes with cloud, we refer to the ($\mathcal{F}$og-2-$\mathcal{C}$loud) cooperation as fog-to-cloud collaboration. Simply put, horizontal nodes cooperation is a coordination processes, while the vertical nodes cooperation is a collaboration processes as per Figure 1.1. Despite the appealing benefits of fog computing, some concerns are undermining its adoption. This includes the approach of how to form the $\mathcal{F}$og-2-$\mathcal{C}$loud ($\mathcal{F}2\mathcal{C}$) collaboration and the $\mathcal{F}$og-2-$\mathcal{F}$og ($\mathcal{F}2\mathcal{F}$) coordination.

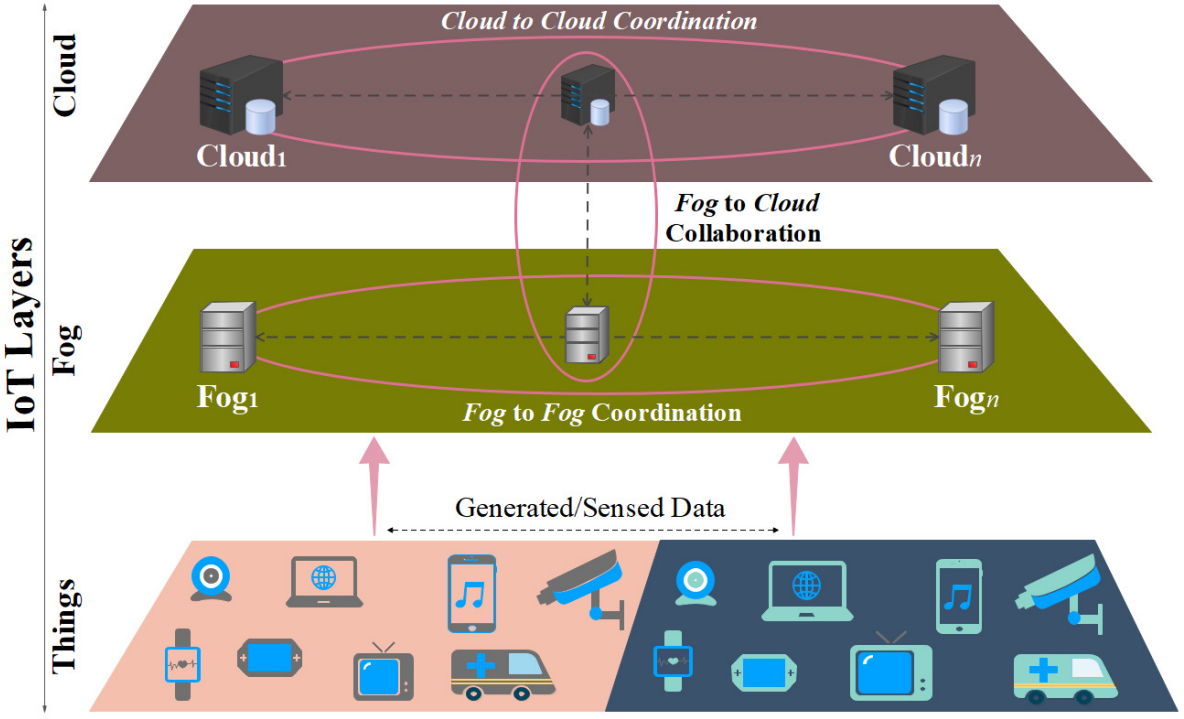


Figure 1.1: IoT Layers and horizontal *versus* vertical nodes cooperation

Fog computing is still an open research area and in its infancy stage, therefore, the motivation of providing a trusted and reliable fog environment for IoT based services comes from the ongoing challenging issues associates with fog computing [14, 27, 28, 20]. Many researchers are focusing on bringing the computing resources to network edges [29, 30]. This will facilitate processing of the data at the edge for time-sensitive applications and services to allow quick responses. Fog nodes are deployed at the edge of the network, and they do not have enough resources and computational power like the cloud [31, 32]. As a result, fog nodes can easily get overloaded with incoming services requests. Also, another noted issue with the cyber-threats is of hostile/open deployment [30, 33, 34]. Hence, there are misbehaving fog nodes that may perform *discriminatory* attacks to ruin the reputation of an IoT service [35]. Thus, avoiding fraudulent or malicious fog nodes for load-balancing and collaboration is still an open challenge. These challenges raise the motivation to develop a fog model that serves as a starting point for the deployment of an efficient and trusted fog computing environment that particularly focus on:

- Cognitive: fog nodes should not only act upon things data, but also direct them to engage in follow-up interactions to achieve certain tasks that boost user's QoS and QoE. Cognitive fog means that it reasons about the surroundings, learns from the past, and adapts to changes, hence fog is featured with components, such as image recognition (see use-case in Chapter 3), that enable a fog node to interpret the surrounding. It worth noting that the cognition feature on fog will not poses extra load on the resources, instead it work inline with FRAMES (In Chapter 5) to manage the load on fog through fog's federation on congestion.
- Reliable: proper network managements for both fog and cloud to promote efficient load balancing among network nodes to address the latency concern of the IoT service request's and things data.
- Trusted: providing a secure/trusted fog environment that allow fogs sharing their resources and exchange data. Fog nodes should make concise decisions that not only includes the choice of best nodes to handle a task efficiently, but also the most trusted fog that could provide best QoS and QoP to end-users. This important to avoid fraudulent and malicious nodes that can disrupt network operations through various attacks (e.g., forgery) which directly effect the reliability of fog computing.

1.3 Research Problem and Questions

Fog computing extends cloud computing to act on IoT data at the edge of the network. This introduces more complexity to the networks, by adding large numbers of devices and service providers, and hence efficient network management/planning is significantly important to design optimal networks. Efficient network management and planning involves the arrangements of the data processing mediums (e.g., fog and cloud computing) in the network. In addition, an appropriate network resources management along a with a criteria to define the selection of specific recipients (i.e., fog, cloud or both) for the IoT data. The efficient network management helps in providing a stable network that guarantees best Quality of Service (QoS) and Quality of Experience (QoE) to the end-users.

Indeed frequent sending of large volumes of data from IoT applications to the cloud leads to both network slowdown and processing congestion, fog computing should be a solution [36]. Hence, a resources management for network components with coordination/collaboration among the nodes on different layers (i.e., fog and cloud layers) is essential. One of the biggest challenges in network management is how to design interaction among fogs/clouds nodes or between fog and cloud in order to accommodate the characteristics of applications because of the huge amount of data generated by these environments. Although fog nodes are placed "closer" to IoT-things so that latency is "taken care" of [14, 37], fog can quickly become congested too due to fog's limited resources. This occurs when the number of service requests, soliciting their services, exceed the fog's computational capabilities [14, 3]. It's obvious that when the service's traffic increases on fog nodes, the potential of having the newly arrived request waiting will be high, and hence having latency. OpenFog [38] reported that, although fog computing provides extensive peer-to-peer interconnection for communication purposes with the clouds, its nodes run in silos, where no collaboration capability, for job processing, is available. Therefore, a proper resource management is required to unlock the silos and free them from the historical stovepipes working pattern. In fact, poor resource management causes latency and inefficiency for IoT-services [6, 26, 39].

In this thesis a comprehensive resources management for fog and cloud computing is presented. The main research problem and questions investigated in this thesis can be summarized as follow:

Problem Statement: despite the appealing features of fog computing, there is a lack of; *(i)* a clear characterisation for fog computing [40, 38], *(ii)* concrete solutions for network resources managements (both fog and cloud nodes) [41, 16], *(iii)* secure and trusted fog network [42, 43]. Hence, there is no solid and stable deployment of fog computing to aid the heavy processes of IoT-things [14, 44, 45, 46, 47].

Research Questions (RQ): according to the problem statement, this thesis is concerned with the following research questions:

RQ1: What is fog computing cognition? How can fog node be cognitive and what are the activities of a cognitive fog? to achieve better QoS and QoE with fog computing.

RQ2: How to provision the computing and networking resources simultaneously in fog/cloud IoT architecture? What criteria are used for selecting data recipients? for better QoS.

RQ3: How to identify computation intensive tasks and point out the congested fog nodes in the network to trigger task's offloading and node's resource sharing, thus allow better QoS and QoE using fog computing.

RQ4: Find an efficiently approach to distribute network workload evenly, also scheduling tasks among the fog nodes with respect to their resources capabilities in $f$og-2-$f$og computing environment for better QoS and QoE.

RQ5: How to ensure a fair participation for fog nodes to avoid idleness? as the traffic can significantly vary depending on node's geo-location in fog environment for better QoS.

RQ6: How to periodically analyse the trust among fog nodes during the run-time? How to identify malicious behaviours to self-adapt and take actions? for better QoE and QoP.

Scope of the thesis: the focus of this thesis is to have a stable performance for fog computing to aid the IoT-services and cloud computing in the ever growing industry of smart things. Aspects related to the performance stability of fog computing involves the development of cognitive fog nodes, reliable resources management and trusted networking, and hence ensure best Quality of Service (QoS), Quality of Experience (QoE), and the Quality of Protection (QoP) to the end-users.

1.4 Thesis Aim and Objectives

This thesis focuses on providing a holistic fog computing approach for a cognitive, reliable and trusted distributed IoT service that ensure best QoS, QoE and QoP. Hence, the main aim and objectives of this thesis are as follows:

Research Aim: design and develop a comprehensive solution to tackle the challenging issues of deploying fog computing in the IoT network. This solution addresses the current limitations of; *(i)* network resources management by efficient resource provisioning algorithm to ensure the Quality of Service (QoS) provided, *(ii)* services reliability and availability in cases of high-traffic and network node's congestion, and hence to ensure the Quality of Experience (QoE) provided. Finally, *(iii)* security and privacy through evolving trusted network environments for nodes, to share resources and user's data, hence avoiding malicious events and attacks to ensure the Quality of Protection (QoP) provided. The solution for these challenges will be integrated in our proposed Cognitive Fog (CF).

Research Objectives (RO): to tackle the research questions and achieve the research aims, this thesis identified few objectives that are essential to develop the desired solution. The objectives are applied iterative to guide the research process, and hence the following research objectives have been identified:

RO1: Review the relevant state-of-the-art research on fog/cloud computing to obtain a systematic understanding and identify the gaps in the area. [to resolve all RQs]

RO2: Identify the key characteristics of fog computing along with the main challenging issues that deter the deployment of fog computing within the IoT network, to create criteria for using fog/cloud and avoid network congestion.[help resolve RQ1, RQ2]

RO3: Investigate the functional and non-functional requirements of fog nodes. Also, investigate the barriers that might impede fog in IoT network. [help resolve RQ3, RQ4]

RO4: Design and develop a comprehensive solution that manages the network resources and ensures the level of security and trust within the network. [help resolve RQ5, RQ6]

RO5: Evaluating the developed solution to ensure that the network could achieve the desired performance, and thus the thesis aim. [evaluate all RQs]

1.5 Research Methodology

Designing and developing a comprehensive solution to tackle the fog computing challenges will need to follow a clear and effective methodology. Therefore, to resolve the thesis aim and achieve research objectives, a conventional Design Science Research Methodology (DSRM) inspired by [48, 49] is adopted as an optimal methodology to fulfill the research aim. The methodology is applied iteratively to guide the research process. The following methodology steps will be used to approach the desired outcome.

1. Problem identification: a survey on the state-of-the-art research studies has been conducted to acquire a full knowledge about fog computing and its challenges. This helps in identifying the research landscape and analysing the gap in research. This step involved some critical thinking of the main solution to aid fog computing challenges and the modelling strategies developed before, hence to justify the value of the new solution.

2. Objectives of the solution: identifying the objectives was driven by the identified problem, hence this required accurate knowledge about the state of the problem, its current solutions, and their efficacy. The proposed solution aims at providing a complete design and implementation to help with the deployment of fog computing in the IoT network. Therefore, the objective is to provide; (i) a cognitive model for fog nodes, and (ii) resources management that evolves two main algorithms:

    - A load balancing algorithm to monitor network resources, active processes, and the incoming services requests volume. Hence, it can monitor the performance and congestion to promote load balancing and offloading to address any latency concerns.
    - Trust and recommendation algorithm and model to help networked fog nodes make the right decision for selecting the appropriate nodes to collaborate with during task offloading. Hence, this process includes assessing the trustworthiness level of the nominated nodes to ensure that the QoP, QoS and QoE provided by the hosting node can be met.

3. Design and development: during this phase, agile methodology is adopted, thus the proposed solution was designed and implemented in parts so that intensive testing and evaluation could be carried out on each part, hence meeting the desired objective. Therefore, the design and development of the solution for fog computing were carried out in Chapter 3-6. Chapter 3 presents the design principles and the preparation of developing cognitive fog nodes. Chapter 4 presents the foundation for networked IoT-nodes collaborations (i.e., fog and cloud) and the criteria for selecting data recipients. Chapter 5 presents the fog coordination model which will allow nodes to outsource their resources to enable load balancing and resources management. Chapter 6 presents an approach to impart useful prognostic information on nodes trustworthiness.

4. Demonstration: in this thesis, multiple healthcare related scenarios are adopted for demonstrating the usefulness of the research and highlight the advantages of using a reliable fog computing in latency sensitive applications/systems. Also, a mathematical proof and simulation is provided to demonstration both network resources managements efficiency and trustworthiness assessments model.

5. Evaluation: to evaluate the performance of the proposed solution to show the feasibility, a test-bed and simulation has been implemented for different part of the solution to intensively test the efficiency. This involves assessing the effectiveness of the solution compared to current efforts available in the relevant literature.

6. Communication: main thesis contributions have been discussed and published in peer-reviewed scholarly publications. A total of 14 publications have been disseminated from the presented work; 6 of which are published in high-quality impacted journals, and the rest are published in high-quality conferences proceedings that are relevant to the scope of this research.

1.6 Contributions to Knowledge

Precise design and development of the proposed solution could potentially lead to a sustainable IoT fog computing based networking paradigm, and thus benefits the computing ecosystem. This thesis made the following key contributions (summarised in Figure 1.2), ordered by their appearance in the thesis.

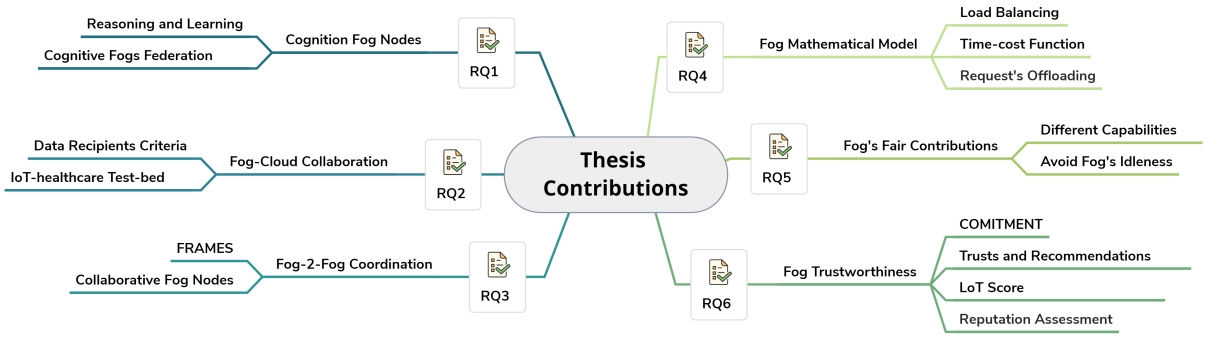


Figure 1.2: Contributions to knowledge summarised in a mind-map

Note: 2 partials fulfill 1 RQ, for example contributions 1 & 2 fulfill RQ1:

1. A novel approach for developing cognition fog computing, this by empowering nodes with reasoning, learning, and adaptation capabilities so that it would weave these fog nodes into service provisioning models. Cognitive fog advocates that fog can interpret the gathered/received data in a way that mimics the process of cognition in the human mind. The core concepts and design of cognitive fog are discussed in Chapter 3. [Partially addresses **RQ1**].

2. Cognitive fog federation, this is about gathering multiple nodes to perform/achieve a specific task efficiently in a certain situation. Fog nodes become members of a federation based on their capabilities, hence federation can be a planned federation or an ad-hoc federation. In planned federations, all nodes are known to each other from the design time to assist, or take benefit from, each other. On the contrary, in ad-hoc federations, the nodes are formed on the fly, so they can communicate with each other based on a need. Both types of federations are discussed in Chapter 3. [Partially addresses **RQ1**].

3. A collaboration model of fog and cloud with a set of criteria for selecting data recipients. These criteria define to whom data of things should be sent (cloud, fog, or both) and in what order (cloud then fog or fog then cloud or both concurrently). This is supported by a healthcare driven IoT case study deployed on a test-bed to demonstrate fog-cloud collaboration. The objective is to assist engineers who are in-charge of developing IoT applications to know what is best for their system. The fog-cloud collaboration and criteria for selecting data recipients are discussed in Chapter 4. [Addresses **RQ2**].

4. A novel Fog Resource manAgeMEnt Scheme (FRAMES) that promotes load balancing to address the latency concern of service request's received from things. This is based on the load distribution algorithm in the $\mathcal{F}$og-2-$\mathcal{F}$og coordination model that achieves a optimal workload among the collaborative nodes. Ensuring the fair participation of fog nodes while maintaining efficient workload distribution can be difficult task, however this can be manged by bounding node's participation by their capabilities (in term of resources) and not time-of-operation or tasks loads. Hence, this approach will allow fairness in term of service delivery, obviously participation level and fairness can predetermined according to the level of QoS and QoE intended to achieve. The load distribution model consider not only the queue length of a node, but also the node's capabilities (i.e., CPU frequency) and their performance with different request types, such as, heavy-request (e.g., from sensor) and light-request (e.g., from CCTV). FRAMES is discussed in Chapter 5. [Addresses **RQ3**, Partially **RQ4**].

5. A mathematical model that backs the decision of load balancing among fog nodes. This investigates the time delay issue and the requests offloading opportunities in the $\mathcal{F}$og-2-$\mathcal{F}$og coordination model. Hence, a time-cost function is developed to compute the time-cost for a service to be processed in multi-nodes based on the number of participant nodes and network conditions. The mathematical model is discussed in Chapter 5. [Addresses **RQ5**, Partially **RQ4**].

6. A novel Fog COMputIng Trust manageMENT (COMITMENT) approach to impart useful prognostic information on networked nodes trustworthiness. Thus, providing a secure and trusted networking environment for nodes to share their resources and

exchange data securely and efficiently. The core concepts and design of COMITMENT are discussed in Chapter 5. [Partially addresses **RQ6**].

7. A novel trust and recommendation model and algorithm that helps nodes make the right decision for selecting the appropriate nodes to collaborate with in the $\mathcal{F}$og-2-$\mathcal{F}$og coordination environment. This is to provide support during the offloading processes to avoid malicious nodes and attacks. Therefore, this process includes assessing the trustworthiness level of the nominated nodes to ensure that the QoP, QoE and QoS are met by the hosting node before a coordination is formed. The trust and recommendation model is discussed in Chapter 5. [Partially addresses **RQ6**].

1.7 Thesis Organisation

The structure of the thesis chapters is shown in Figure 1.3. It worth noting that the work presented in this thesis is derived from several publications published during the PhD journey. The rest of the thesis is structured as follows:

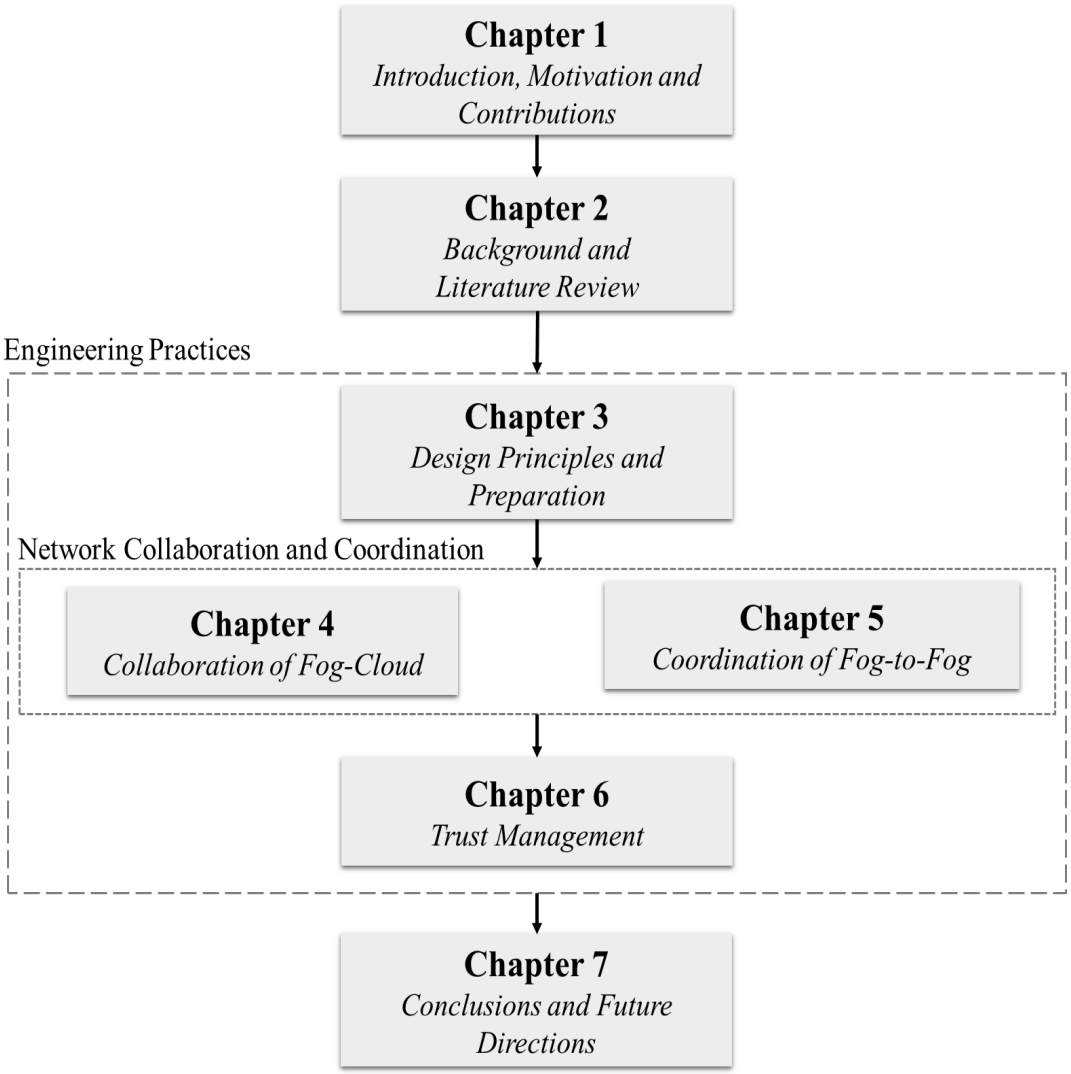


Figure 1.3: Thesis Structure

Chapter 2: provides a background on the basic concepts of IoT, cloud computing and fog computing along with a comparison between the two computing paradigms and their existence in the IoT era. Then, the chapter reports on a systematic literature review on fog/cloud challenges with critique for some of the recent research efforts.

Chapter 3: proposes a holistic fog computing architecture and design principles, also the concept of cognitive fog is also presented in this chapter. The design principles of fog nodes are highlighted based on the main requirements of fog in term of networks communications and geo-location along with their functional and non-functional requirements.

Chapter 4: proposes a fog-cloud collaboration model that assists organizations wishing to ride the IoT wave in determining where data should be sent (cloud, fog, or cloud and fog concurrently) and in what order (cloud, fog, or cloud and fog concurrently). Hence, a set of data-recipient selection criteria, such as frequency, sensitivity, freshness and volume, have been proposed to ensure a smooth collaboration.

Chapter 5: proposes a novel Fog Resource manAgeMEnt Scheme (FRAMES) to justify fog distributions and management with an appropriate service's load distribution and allocation. Also, the mathematical model that backs the decision of load balancing among fog nodes.

Chapter 6: proposes Fog COMputIng Trust manageMENT (COMITMENT), which is a software-based solution that is responsible for providing a secure and trusted environment for nodes to share their resources and exchange data packets. Also, a trust and recommendation model that helps nodes make the right decision for selecting the appropriate nodes during coordinations.

Chapter 7: concludes and summarises the thesis outcomes and contributions. Also, potential future work and possible extensions are also discussed.

CHAPTER 2

Background and Literature Review

We know next to nothing about virtually everything. It is not necessary to know the origin of the universe; it is necessary to want to know. Civilization depends not on any particular knowledge, but on the disposition to crave knowledge.

George Will

2.1 Introduction

Due to the massive volume of data generated from Internet of Things (IoT) applications, IoT becomes a source of big data. Currently, IoT data are backed by cloud computing, where data is processed by big data-systems in a powerful data-centres. However, with the increase of data velocity and volume, the distant cloud computing may not be able to satisfy the ultra-low latency requirements for IoT applications [14, 7, 4], such as patient monitoring applications in [50, 40]. Therefore, the fog computing paradigm emerged to support the cloud by providing data processing and analysis at the edge of the network where IoT nodes are located. This chapter introduced a background on the basic concepts of IoT, cloud computing and fog computing along with a comparison between the two computing paradigms and their existence in the IoT era.

The survey was conducted based on studying a systematic literature review by using search terms like "fog vs cloud", "fog and cloud", "fog resource managements", "fog offloading", "cognitive fog", "load balancing", "fog congestion" and other compensations of these terms. The number of publications initially identified were over 250, which has been filtered to the most relevant researches. Hence, examined more than 120 relevant studies/researches from multiple research databases, such as IEEE, Web of Science and Elsevier libraries. Research criteria were then developed (presented in Section 2.5.1) from the analysis of the primary studies. These criteria were then used to classify and analyse current research. Also, a cross analysis performed to derive the gaps and directions for further research, has been conducted in this thesis to fill the gaps.

2.2 Background

This section provides a brief background on the Internet of Things (IoT) and its main data processing mediums (i.e., fog and cloud). In addition, it analyses cloud computing and fog computing in terms of similarities and differences along with appropriateness for certain types of applications. Finally, last subsection discusses the co-existence of fog and cloud in the IoT paradigm.

2.2.1 Internet of Things (IoT)

There are no doubts that the Internet has impacted people's lives and organizations' practices. Acting as a reliable communication middleware, the Internet permits the connection of different hardware and software components to the extent that location is no longer an obstacle to information availability and service accessibility. The latest Information and Communications Technology (ICT) developments, introduced the IoT, targets convenience by ensuring that things in people's and organizations' surroundings are accessible and responsive to their requests. Smart-home is a good example of this convenience where things like white appliances take actions on behalf of the home's occupants.

Different forms of computing contribute to IoT functioning including mobile, however, the abundant literature about IoT (e.g., [51, 52]) does not help propose a unique definition of what is IoT. On the one hand, Barnaghi and Sheth provide a good overview of IoT requirements and challenges [52]. Requirements include quality, latency, trust, availability, reliability, and continuity that should impact efficient access and use of IoT data and services. While the challenges resulting from today's IoT ecosystems featuring billions of dynamic things and thus, making existing search, discovery, and access techniques and solutions inappropriate for IoT data and services. On the other hand, Abdmeziem et al. discuss IoT characteristics and enabling technologies [51]. Characteristics include distribution, interoperability, scalability, resource scarcity, and security, along with enabling technologies include sensing, communication, and actuating. These technologies are mapped onto a three-layer IoT architecture consisting of perception, network, and application, respectively. Each layer could have any of the three main IoT components which enables seamless ubiquitous Computing [53]: (*i*) Hardware that is made-up of data sources (e.g., sensors, actuators) along with the embedded communication components and protocols. (*ii*) Middleware to support the on demand storage and computing tools for data analytic. This includes various types of data processing mediums (e.g., fog and cloud computing) that can serve the IoT needs. (*iii*) Data presentation through easy to understand visualization and interpretation tools which can be widely accessed on different platforms and which can be designed for different applications.

A comprehensive guide about applications, protocols, and best practices in the IoT is released by the DZone group in 2017 [54]. The guide covers various aspects relevant to IoT such as privacy, big data, monitoring, context, and architecture. Some terms worth mentioning in the guide are: (*i*) consensual IoT meaning that all IoT providers need to respect and take all measures in their power to protect users' privacy and safety, (*ii*) ubiquitous computing meaning that the next generation of IoT systems will require a middleware protocol capable of managing heterogeneous devices, supporting scalability, ensuring privacy and security, and encouraging utility, and (*iii*) context meaning that approaching users' attention should be at the right time with the right messaging. Although the IoT is getting a significant attention in both research and industry, many challenging issues still need to be addressed in both technological side (e.g., communication and processing) and the social side (e.g., security and privacy) before the IoT can be widely accepted.

2.2.2 Cloud Computing

Cloud computing within the definition of The National Institute of Standards and Technology (NIST) [55] is *"a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction"* [55]. Cloud computing brings together already existing technologies such as virtualisation, grid computing and utility-based pricing [56] to meet industrial demands. The cloud is composed of five characteristics, three service models and four different deployment models [55]. The five main cloud characteristics are:

- On-demand self-service: a consumer can independently provision computing services that may require server time and network storage automatically. Without the involvement of human interaction with the service provider [57].
- Network access: services and capabilities offered are available via standard mechanisms that promote and allow the use of heterogeneous client platforms [58].
- Resource pooling: the providers computing and storage resources are combined to allow services to be used by multiple consumers, by using, a multi-tenant model with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. In terms of location, the consumer commonly has no knowledge or control over the location of the provided resources but may be able to specify location at a higher level of abstraction [59].
- Rapid elasticity: capabilities can be elastically provisioned and released in some cases automatically to scale rapidly outward and inward commensurate with demand [58].
- Measured service: cloud systems automatically control and optimise resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing and bandwidth). Resource usage can be monitored, controlled and reported. Therefore providing transparency for both the provider and consumer of the utilised service [60].

Cloud computing employs a three service-driven business model [58]. In other words, hardware and platform resources are provided as services on an on-demand basis. Concep-

tually, every layer of the cloud architecture can be implemented as a service. However, in practice, clouds offer services that can be grouped into three categories [61]: Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS) which are described as follow:

1. Software as a Service (SaaS): provides on-demand applications over the internet.
2. Platform as a Service (PaaS): this service provides platform layer resources that include operating system support, frameworks for software development.
3. Infrastructure as a Service (IaaS): refers to on-demand provisioning of infrastructural resources, usually in terms of virtual machines.

Due to different consumer demands such as high reliability and businesses aiming to reduce operation costs by using the cloud, there are four different deployment models that all have their own benefits and drawbacks. The four deployment models are as follows:

- Private cloud: known as internal clouds, this is designed for exclusive use by one organisation. The management of a private cloud can be done by the organisation itself or by an external provider. By deploying a private cloud, the organisation can benefit from the best degree of control over performance, security and reliability [62].
- Community cloud: this deployment model infrastructure is provisioned for exclusive use by a specific community of users or organisations. They are able to share a cloud as the users have similar requirements (e.g., mission, compliance considerations and security). It can be owned, managed/operated by one or more organisations [60, 62].
- Public cloud: in this model service providers offer their resources as services to the general public rather than organisations. Public clouds have multiple benefits to the service providers, such as no initial capital investment on infrastructure and shifting of risks to infrastructure providers [62].
- Hybrid cloud: this model is a combination of public and private cloud models that tries to benefit by reducing the limitations of the individual approaches. Part of the service infrastructure is operated within private clouds while the remaining sections are run in public clouds [59, 62].

2.2.3 Fog Computing

Fog computing, also call fog networking or fogging, is described as a highly virtualised platform that provides application services, and network services at the edge of the network, closer to the IoT things. Fog nodes act as middleware and are placed between things and cloud layers [18]. Fog computing is similar to cloud computing, it offers a range of application services, such as, data processing and data analysis with closer distance. However, fog is expected to deliver these services faster, more securely, and more reliably than clouds [19, 16] due to its proximity. The fog layer is composed of large scale geo-distributed fog nodes, which are deployed at the edge of networks [1]. Each fog node is equipped with on-board computational resources, data storage, along with network communication facilities to bridges things and cloud within the IoT network [14].

It is worth noting that fog computing is not a substitute for cloud computing but is a complement it [1] which introduces to lower bandwidth burden along with reducing transmission and processing delays. Fog computing only offers the ability to extend the storage, networking and computing capabilities of the cloud with a better positioning within the network in relation to the end-devices [50, 39, 31]. In general, fog computing can support, serve, and facilitate services that are not appropriately served by the cloud due to cloud proximity, such as, (*i*) services that are latency sensitive (e.g., online gaming) [14]; (*ii*) geo-distributed services (e.g., pipeline monitoring) [22]; (*iii*) services that requires mobility support with high speed connectivity (e.g., connected vehicles) [3]; and finally, (*iv*) services in large scale distributed control systems (e.g., smart grid) [1]. Figure 2.1 shows an IoT-based healthcare example where fog computing is adapted to interact with the different types of sensors and actuators, such as camera, wearable, environment sensors. On the thing side, fogs directly communicate with things to accumulate the data via wireless communication connections interfaces (e.g., Zigbee, LoRa). On the other side, the fogs are interconnected with clouds to leverage the rich functions and services of the cloud.

The fog nodes can work independently or in collaboration with cloud node. Since fog nodes have on-board computational power resources and data services facilities (e.g., data processing and aggregations), fog can independently provide predefined/dedicated services without cloud assistance [1, 63, 64, 65]. For example, fog nodes can independently monitor

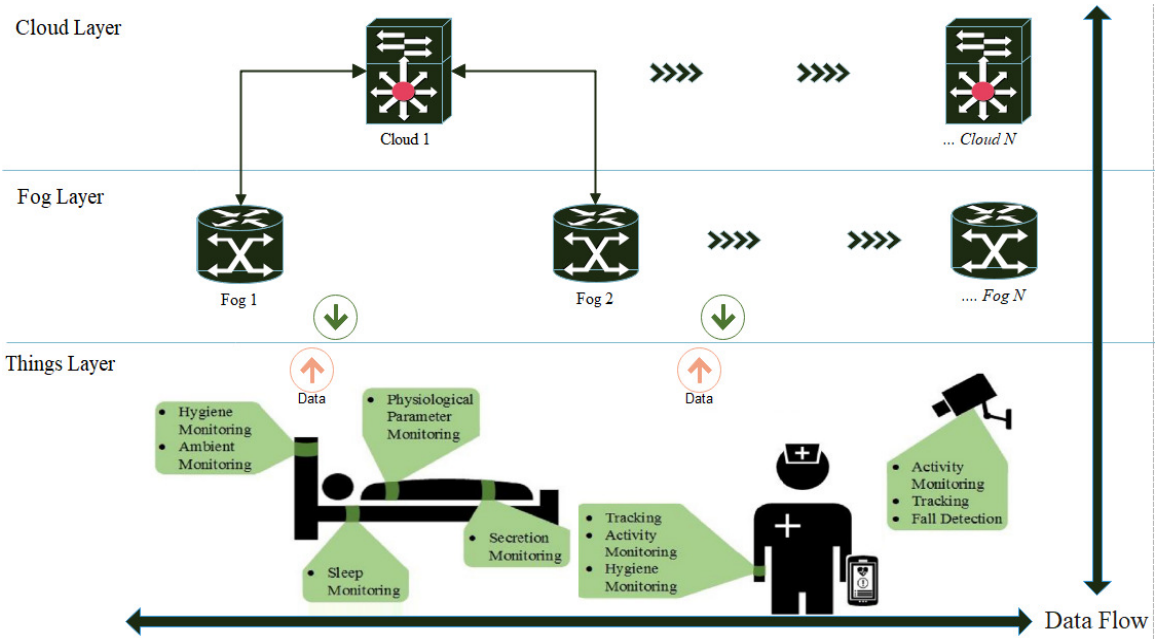


Figure 2.1: IoT-Fog based Healthcare Example

and analyse real-time data from a pressure sensor and then initiating actions like opening/closing a valve in response to a pressure reading from the sensor. However, fog has limited hardware capabilities compared to cloud, therefore it may get congested very easily when the data traffic is high and exceeds fog capacity. Hence it may need cloud assistance or fog's coordination to process the tasks. Its obvious that when the service traffic increases on fog nodes, the potential of having the newly arrived services requests waiting is high, thus having latency. The high traffic is associated with fog node capacity and its resource managements, therefore, the better the fog resource management, the lower is the latency.

Despite the appealing benefits of fog computing, some concerns are undermining its adoption. These include how to specify $\mathcal{C}$loud-2-$\mathcal{F}$og ($\mathcal{C}2\mathcal{F}$) collaboration and $\mathcal{F}$og-2-$\mathcal{F}$og ($\mathcal{F}2\mathcal{F}$) coordination. Thus, since fog is being introduced to address the problem of mobility and latency for delay-sensitive applications, most issues within the fog layer can be categorised under the umbrella of Quality of Service (QoS) [24] (e.g., reduce latency, mobility and privacy) and Quality of Experience (QoE) (e.g., service experience) [3]. It is worth noting that the definition of *service latency* in this thesis is the total time required to deliver a service, which covers the time when an IoT thing sends a service request and receives the response back. This includes the travelling interval time and the processing time.

2.2.4 Cloud *versus* Fog Computing in IoT

Since 2000, cloud has become a popular geo-distributed operation model for organizations. It differs from its predecessor models (i.e., grid and cluster) not only in terms of architecture, networking, and middleware, but in terms of consolidating hardware and software resources into co-located server farms known as data-centres. Cloud data-centres facilitate the delivery of computing, storage, and networking to organizations as the 5^{th} utility (after water, electricity, gas, and telephony) using a pay-per-use pricing model. This leads to minimizing operational, acquisition, and maintenance costs. Moreover, the technical complexity of managing the clouds is hidden away from organizations. Despite the bright side of cloud, it does not, unfortunately, suit all application types, especially IoT-based services and application. For instance, latency-critical and data-privacy sensitive applications cannot be hosted on the cloud due to reasons such as: (i) high-latency added by network connections to datacentres [66] and (ii) multi-hops/nodes between end-users and datacentres raise the probability of attacks. The centralized nature of cloud leading to a high communication cost and power consumption are extra reasons for the cloud unsuitability [66].

In conjunction with the IoT boom, fog has been introduced and become a hot topic in recent years. It was first introduced by Satyanarayanan et al. in 2009 [67] and generalized by Cisco Systems in 2014 [68] as a new ICT-based operation model that would make computing, storage, and networking facilities "close" to where data is captured and/or located. The extension from cloud to fog is not trivial due to their subtle similarities and differences. Cisco[1] provides Table 2.1 to illustrate how cloud and fog would handle the characteristics of certain applications differently. For instance, real-time applications that ask for almost immediate action and high data protection, would discard cloud as an operation model. On the contrary, fog would offer better support to mobile applications compared to cloud. A thorough discussion of fog's feasibility is presented by Varghese et al. [69]. The authors mention that by 2020, 43 trillion gigabytes of data will be generated and thus, need to be processed in cloud data-centers. However, this operation model cannot be sustained for a long time due to frequency and latency of communication between these devices and

[1]Cisco blog on IoT, from Cloud to Fog Computing
`https://blogs.cisco.com/perspectives/iot-from-cloud-to-fog-computing`

Table 2.1: Appropriateness of cloud/fog to respond to system's characteristics (Cisco)

#	Characteristics	Cloud	Fog
1	Latency	High	Low
2	Delay jitter	High	Low
3	Location of service	Within the Internet	At network edge
4	Distance client to server	Multiple hops	One hop
5	Security	Undefined	Can be defined
6	Attack on the enroute	High probability	Very low probability
7	Location awareness	No	Yes
8	Geo-distribution	Centralized	Distributed
9	No. of server nodes	Few	Very large
10	Support for mobility	Limited	Supported
11	Real-time interaction	Supported	Supported
12	Last-mile connectivity type	Leased line	Wireless

geographically distributed data-centres. Fog comes to the rescue of cloud by processing data closer to its source so, that, network traffic is reduced and both quality-of-service (QoS) and quality-of-experience (QoE) are expected to improve. According to Taivalsaari and Mikkonen, fog harnesses a network's edges for computation to pre-process thing data and trigger alert requests locally on the basis of pre-defined criteria [66]. Table 2.2 also provides more details about cloud versus fog as an operation model for applications.

Table 2.2: Analysis of cloud *versus* fog

Cloud	Fog
Data/applications are processed/run over the cloud, which is time consuming for large data.	Data/applications are processed/run over scattered network edges, so this could consume less time.
High demand of bandwidth, as a result of sending every bit of data over communication channels.	Less demand of bandwidth due to access points located next to data.
Slow response time and scalability problems due to cloud servers located at remote places.	Possibility of reducing response time and scalability due to edge servers located at close places.

2.3 Cloud and Fog Co-Existence in IoT

Different reasons motivate the adoption of fog over cloud such as avoiding the risk of network slowdown and storage congestion [36], when large volumes of data are regularly transferred. In the following, we discuss some works that expose these reasons.

Aazam et al. refer to the amount of data that the IoT will generate requiring additional support to applications deployed on the clouds [70]. This support is exemplified by first, pre-processing and trimming data before sending it to the clouds and second, having a smart gateway accompanied with a smart network or fog to regulate data flows from IoT devices to the clouds. There two forms of communication arising between the smart gateway and IoT thing: (*i*) direct (*aka* one-hop) when the communication is of a smaller magnitude and IoT things are not very diverse and (*ii*) indirect (*aka* multi-hops) through base stations and sink nodes when IoT things are diverse, more widely spread, and their data is more heterogeneous.

Lewis et al. advocate cyber-foraging that "*is the leverage of external resource-rich surrogates to augment the capabilities of resource-limited mobile devices*" [67]. This leverage could be applied to IoT things that could offload their computation and data-management duties to cloudlets instead of maintaining continuous remote connection with the clouds [71]. Cyber-foraging is very appropriate for military personas (e.g., soldiers) who make extensive use of IoT things (e.g., handled tablets) during ground operations. These personas use multiple applications like speech and image recognition and situational awareness that are computation intensive and take a heavy toll on the devices' batteries and computing resources. Lewis et al. suggest partitioning military-critical applications into a very thin client that will run on the IoT device and a computation intensive server that will run on the cloudlet which is a fog devices that provide data staging and data filtering.

Yannuzzi et al. discuss the key ingredients of a successful IoT recipe [72]. These ingredients are fog computing, cloud computing, and more fog computing, and allow the provision of an alternative to having storage and computing capabilities always confined in data-centres. Applications that exhibit features of mobility, reliable control and actuation, and scalability should benefit from Yannuzzi et al.'s proposed recipe. Despite the fact

that the authors raise the question of where storage and computing resources should be appropriately located, they do not provide any criteria that would address this.

Petri et al. analyze and manage data in a multi-layered cloud configuration [73]. Although data are expected to flow from the lowest level, consisting of sensors, to the highest level, consisting of data-centres, through an intermediate level, consisting of gateways, the authors observe that complying with this flow is not a must for certain applications like IoT. To this end, data analysis is usually performed at 3 layers, in-situ (i.e., sensor), in-transit (i.e., gateway), and data-drop (i.e., cloud). The authors observe that only a subset of data is sometimes needed for analysis at the cloud level. Thus, they propose a coordination mechanism as an objective function that minimizes and/or maximizes some QoS constraints over each layer. These constraints depend on criteria like type of sensing (for a multi-purpose sensor), number of concurrent streams processed (for a gateway), and execution time per application (for a cloud).

Wen et al. discuss the challenges that hamper the orchestration of IoT services from a fog perspective, such as service selection and placement, parallel computation, late calibration, dynamic orchestration with run-time QoS, to cite just a few [74]. The authors develop a fog orchestrator that consists of a resource manager, planner, and status monitor. On top of this, the orchestrator interacts with a cloud data-centre and other fog nodes in charge of locally storing and/or computing data of the IoT.

Chekired et al. discuss the role of fog computing in sustaining Industry 4.0 growth [75]. To achieve this role, they propose a fog-based multi-tier architecture and a scheduling model for data processing. Through this architecture and model, the authors aim at minimizing communication and data processing delay in Industry 4.0 systems by organizing fog servers hierarchically and optimally aggregating and offloading data peak loads in lower-tier servers to higher-tier servers.

Despite the co-existence of cloud and fog in the afore-mentioned approaches, fog largely dominates cloud in term of appropriateness for IoT applications. The coordination (Fog-2-Fog or Cloud-2-Cloud) and collaboration (Fog-2-Cloud) models proposes in this thesis is one step forward towards a holistic approach for a cognitive, reliable and trusted fog-cloud models to support IoT applications.

2.4 IoT and Data Processing Mediums Challenges

In the IoT, things provide a vast amount of data every second, which requires special treatments by the available ICT and data processing mediums to be converted into a useful knowledge. In order to generate useful data out of raw data, a proper data filtering and aggregating should be supported by the ICT with a special care for data security and privacy which is set/identified according to the application's requirements. ICT currently adopts fog and/or cloud in most IoT-based application as data processing mediums. However, due to the heterogeneity of data, IoT-nodes (i.e., thing, fog and cloud) and communications protocols, frequent challenging issues arise, such as resource management. This section summarise the key technical challenges and issues associate with IoT, fog and cloud computing.

2.4.1 IoT Challenges

IoT enables many new features and opportunities for industry (e.g., smart grid) or directly to end-users (smart-homes). However, the IoT itself lacks some theory, architecture standards and technologies that can integrate the virtual world and the real physical world in a unified framework [76, 77]. The main challenge issues of IoT are summarised below:

- Standard Challenge: since the IoT does not have well-defined standards that govern the interaction and the development of the IoT-systems, there are many issues related to IoT-systems management and development [78], such as connectivity and security. Standards could play an important role in forming the IoT, thus they will allow IoT-systems developments to follow some minimum requirements that could satisfy both QoS and QoP. Generally, standards should be publicly available and free to use.

- Data management: the IoT sensors and applications provide vast amounts of data every millisecond, hence these data need immediate attention to serve their storage and processing needs. These data consist not only of traditional discrete-data, but also a continuous stream of raw-data that are generated from digital-sensors [79], thus entering the era of big-data. For example, industrial/environment sensors that stream different data, such as location, movement, temperature, humidity etc. The

current adopted architecture and network topology of today's data-centres are not prepared to handle or deal with the heterogeneous nature of the IoT data [79, 80].

- Architecture Design: since IoT encompasses an extremely wide range of technologies, communication protocols and heterogeneous devices, the network and architecture of the IoT become very challenging to identify, manage and standardise. In the IoT, data integration over different environments is a tough process and should be enabled by modular interoperable components [76]. Thus, IoT infrastructure development requires systems to: (*i*) accumulate data from different sources, (*ii*) determinate and analyse data of relevant features and relationships to interpret the data, (*iii*) provide support for decision making. Therefore, a single architecture cannot be a blueprint for all IoT systems/applications [76, 81]. Hence, this stresses the need for a heterogeneous IoT architectures, this should be flexible and adaptable, following preset IoT standards and not restricted to a single solution.

- Technical challenges: the IoT technologies can bring complex technical issues in both things-design and networking protocols [82]. The things-design has lots of requirements so they can be widely accepted, however, there are two main requirements that are essentials when producing any thing-device: (*i*) extremely low-power consumption during both active and sleep modes (*ii*) ultra low-cost in terms of production and delivery to consumers. In addition, things should be a bandwidth-saver as the bandwidth in IoT is critical and can vary from kbps to mbps from sensing simple raw-data to capturing video streaming [76]. In term of networking protocols between things and things with data-centres, these should be simple and fixable and more importantly they should be low-cost and provide reliable connectivity [76].

- Security and Privacy: the IoT-things generate and exchange enormous amount of security-critical data along with privacy-sensitive information, hence are appealing targets of various attacks [83, 84, 85]. Cyberattacks on IoT-things are very critical since they may cause physical damage and even threaten human lives [83]. A recent study from HP Inc revealed that 70% of the most commonly used IoT devices contain serious vulnerabilities to attacks [86, 87]. This poses new challenges on the design and implementation of secure embedded systems that typically must provide multiple

security features and safety functions. Current IoT applications are insufficiently developed with regards to fulfilling all the desired security requirements and endure security and privacy risks [83, 44]. Protecting and avoiding attacks on IoT will require a complete cybersecurity framework. This framework should be able to cover all abstraction-layers of the heterogeneous IoT across platform boundaries [83, 42].

2.4.2 Cloud Challenges In IoT

Integrating cloud computing with IoT can provide several benefits, also it can foster the development and improvement of many IoT-systems. Cloud can help in managing the connected IoT-things (i.e., sensors) remotely, thus making the generated data globally accessible. However, complex IoT-systems, such as patient's monitoring, could raise several challenging issues for cloud computing [88, 89]. The main recurring challenging issues of cloud computing in the IoT are twofold:

- Proximity: the current cloud networking paradigm relies on powerful centralised servers that are located somewhere in the world. Also, the traditional traffic-management is mainly based on a centralised control mechanism. This poses heavy loads on the traffic management server and causes a long response delay [3]. The long-distance between the data-source (i.e., IoT-things) and designated cloud servers makes cloud an impractical solution for IoT data processing, especially for delay sensitive IoT-applications. This is because by the time the data reaches a far cloud servers, the importance of IoT-things data would be depreciated [17, 39]. This prompts the need to evolve the network in order to permit the data processing at the edge of the network

- Performance and Reliability: cloud adoption in the IoT could feature many application and services, especially those that requires heavy tasks processing, such as video processing. However, adopting cloud computing in mission critical applications would raise some reliability concerns [89], especially in the context of moving things, such as smart-vehicles that often experience networking/communications being intermittent or unreliable when they are in motion [3]. This will seriously affect the usability and user-experience, and thus the QoS provided by the cloud. It worth noting that cloud's main issue is to have a stable and acceptable networking performance throughout as

timeliness may be heavily impacted the performance [89, 90], for example peak and off-peak time performance.

2.4.3 Fog Challenges In IoT

Although Fog computing is a promising network paradigm to serve IoT applications/systems, there are a number of challenging issues that need significant attention. Despite the fact that there are a number of research projects that have been conducted on fog computing, there are on-going research challenges and opportunities still open to discuss. The main recurring challenging issues of fog computing in the IoT are threefold:

- Heterogeneity: in IoT-based applications, the bottom most layer within the IoT (thing layer) can have multiple different devices such as smart-phones, autonomous cars, wristbands and other IoT smart objects. The heterogeneity issue emerge at this point due to the heterogeneous data-formats [91], which limits the data aggregation processes and thus could directly impact the QoS and QoE can be provided to the end users if data could not processed in time due to its heterogeneous nature. Dealings with various data formats and different communication protocols for managing unstructured data becomes a major issue. Heterogeneity becomes an substantial designing factor to be considered during the design phase of an $IoT\text{-}Fog$ based system architecture [92]. Therefore, this raises the issue of how fogs can handle various data formats and network protocols from highly dissimilar sources of data.

- Resource Management: when IoT layers (a.k.a, things, fogs, and clouds) are integrated into one network, the management of the resources becomes a primary concern [93]. Resource discovery and sharing are critical factors for IoT applications, as it could affect the services and QoS directly. Due to the dynamic nature of IoT nodes in terms of communication and data acquisition, significant challenges arise. Even when considering fog without the aid of the cloud [63, 64, 65], resource management can be challenging. This is due to the limited computing and storage resources available in the fog compared to the cloud. Fog resource management is not been widely studied in most existing researches studies [94, 26]. Thus, the question of *"How to balance the load in Fog layer?"* towards delay minimising, services availability, cost-effectiveness,

and power-reduction is an open research challenge. Understanding the nature of fog in the way it deals with data and the mobility of things may be beneficial for resource management and task scheduling within fog to allow best QoS. In resource management, offloading can be a solution to balance the fog's workload, however, it still experiences some issues [1, 39, 95, 96, 97], for example, the question of "when to offload a task?" in a way that can allow efficient resource management while insuring best QoS is still an open research challenge. Offloading refers to the transfer of tasks from one entity to another, such as one fog to another or to cloud. For example a fog node transferring the load to another node that's experiencing less load. What makes offloading challenging is how tasks should be offloaded, and what reasons should be applicable to make the decision to offload the task(s), hence achieving minimal delay. To the best of our knowledge, there are a number of research studies tackling the challenges associated with task scheduling [26] at the fog layer. However, most proposed research so far permits distributing jobs over participant Fog nodes regardless of the current workload on nodes. In other words, they have not appreciated the possible unbalanced situation among fog nodes in terms of traffic and workloads [26]. In fact, most of the proposed algorithms focus on reduce the task blocking probability, hence, they cause such unbalanced loads, among participant nodes. This stresses the need for algorithms and framework that support offloading [98, 99] and load redistribution [100] activities at the fog layer. Offloading could be detrimental to latency-sensitive systems if carried out in an unsuitable manner. If the offloading of the tasks causes more delay, it could reduce the QoS and QoE.

- Security (Threats and Attacks): security in fog computing is also a changing issue which can directly impacts the QoP and QoE to end users, threats and attacks are mainly because of fog's geo-distribution and positioning within the network. Working at the network edge could present threats that do not exist within an organised cloud architecture. One of these threats could be a Denial-of-Service (DoS) attack, in which the attacker can relay and alter the communication between two parties [101], thus affecting the QoP. For example, in a healthcare system, the attack could compromise a gateway that is in between a patient monitoring sensor and a fog node that processes patient's data, hence, a major consequence regarding the patient's health occurs if

the attacker altered the data that being processed. Not only gateways are abused for attacks, but fog node themselves can also be attacked and manipulate them to become malicious fog nodes. A malicious fog nodes can disrupt network operations through various attacks, the following attacks [102, 103, 104] are considered since they can directly effect the reliability of fog computing.

1. Forgery:- malicious fog nodes may forge their identities and fabricate fake data to mislead other fog nodes and IoT services. This type of node burdens the network resources by excessively consuming network bandwidth, storage and computational power by running a fake services and fabricating large amounts of faked data.
2. Tampering:- malicious tampering with fog nodes degrades fog efficiency by delaying, modifying or droping the transmitted data. Detecting such malicious fog nodes is difficult as transmission failure or delay may be caused by other factors, such as unstable channel conditions or weak network signal, and not due to tampering with the fog.
3. Spam and Jamming:- this attack burdens the network with unwanted content and data by generating large amount of bogus data to jam the network channels and the fog's resources. Such attacks are generated and spread by malicious fogs to consume network and fog's resources so that the fog become unavailable for other services and processes.
4. Impersonation: A malicious fog pretends to be a legitimate fog node to provides fog's services, but then it provide fake or phishing services to users and breaches users privacy.
5. Denial of Service (DoS):- malicious attacks to disrupt the fog's services and make them unavailable to the intended users, by flooding the target fog nodes with superfluous service requests. This attack consumes network resources to prevent the requests from legitimate users from being fulfilled. Fog nodes are highly vulnerable to DoS attacks compared to the cloud due to the fog's limited resources.

2.5 Related Works

The Research and Development (R&D) community has found that the employment of fog computing at the edge of the network can provide a low latency, location awareness and many more features that can improve the Quality of Services (QoS) [14, 4, 3, 38, 96, 105]. However, there is a lack of concrete solutions supporting the development and adoption of this computing paradigm [39, 40, 106]. Although Fog computing is still in its conceptual stage, there is some related research which needs to be regarded. The following subsections are: (*i*) research criteria section to define the research criteria that are used to critique the recent efforts, (*ii*) existing work limitations section summarises some of the main research on fog computing, and (*ii*) gap analysis that highlights the contributions of this thesis.

2.5.1 Research Criteria

Despite the growing interest in fog computing for IoT-enabled applications and systems, there does not appear to be an established approach for a concrete solution supporting all fog computing features. Recent efforts on fog computing has been studied with regard to the following criteria:

- Heterogeneity criterion: fog nodes are heterogeneous, hence fog nodes can significantly vary in their processing, storage and communication capabilities. Therefore, during the design phase of IoT-fog based system, this criterion should be taken into account in order to specify the limitation and capabilities of fog nodes, hence the system will be able to decide/pick the right resources for job deployment to allow best QoS.

- Cognition criterion: fog nodes should not only act upon things data, but also direct them to engage in follow-up interactions that should lead to achieving certain tasks. Hence, fog should be able to interpret gathered data so that it can learn from their process experiences according to different situations/scenarios and improve when performing repeated processes. It worth noting that the cognitive capabilities and learning activities are not bound by the types/formats of the data, hence fog can learn from any data as long as the data can be interpreted (i.e., unencrypted) by the fog to allow fog identifying patterns and similarities for future processes for best QoE.

- Scalability criterion: an elastic fog system is essential in the era of the IoT because fog computing is expected to cover vast number of IoT things which are significantly growing over time. Thus, fog should be able to make decisions whether to scale up/down or scale out upon the number of connected devices/Things. Therefore, IoT-fog based systems should have an elastic scalability manner to insure fog reliability at such large IoT scales, hence having best QoS and QoE.

- Federation criterion: fog nodes are distributed over several geographical location based on things locations, also they can be provided/operated by different parties. Therefore, cooperation among fog nodes is essential in order to ensure QoS leverage (e.g., reduce latency) for IoT Things. Therefore, federation among fog nodes is an essential criterion for fog nodes to help with in-sourcing and out-sourcing resources/processes to other nodes which may help to deliver best QoS and QoE.

- Security and Privacy criterion: the flexibility of offering different fog-based services by different providers complicates the trust situation between fog nodes and end-users. Cryptographic-based techniques can prevent external attacks from expose user's privacy, however, they are not useful when fog nodes are already authenticated and part of a networks using legitimate identities. Therefore, fog nodes should be supported by some enhanced security features (e.g., on the fly authentication) to mitigate the risk of breaching user-privacy, thus avoiding malicious fogs/attacks and having best QoP.

2.5.2 Existing Work Limitations

The benefits of adopting the fog computing paradigm has attracted researchers and organizations in different research disciplinary. Since the emergence of fog computing by CISCO in 2012 [26], researches studies related to the fog paradigm have been actively conducted over the last few years. In early efforts, studies were primarily conducted on emphasizing the importance of the fog and its services along side fog usability, while today the focus of research has subsequently shifted toward fog framework, resources management and security. This section summarises some of the main state-of-the-art research/contributions on fog computing, these research studies have been grouped according to their field and studied based on the criteria in Section 2.5.1.

2.5.2.1 Fog Computing Resource Management

Beate et al. [107] propose a job placement and migration approach for providers of infrastructure that incorporate cloud and fog. The approach ensures end-to-end latency restrictions and reduces network usage by planning load migration ahead of time. The authors also discuss how the application knowledge of Complex Event Processing (CEP) can reduce the required bandwidth of Virtual Machines (VMs) during their migration. However, the presented approach does not consider offloading high load among fog nodes; hence they assumed the fog nodes are able to perform computationally intensive tasks, which might not always be the case thus impacting the QoS provided by fog. According to our criteria, the authors indicate that the approach can be employed at large scale in the real-world, so the scalability is met as the presented approach can scale well with the increment of jobs. However, presented approach seem to overlooking heterogeneity of both devices and data which may impact QoE and the federation among fog nodes to allow better job processing and migration which can directly impact the QoS and QoE.

Agarwal et al. [108] proposed a resource allocation in a fog context. They proposed 3-layer architecture, client, fog, and cloud allowing the distribution of workload between the cloud and fog nodes. In fact, the authors check whether enough processing is available on the fog node so that, all or some tasks are executed or even postponed. Tasks could also be directed to cloud nodes. The limitation of this work is making an assumption that

every fog and cloud nodes will have a manger to manage the collaboration and performance of the node. Thus, this approach is not well discussed to prove the proper execution of distributed tasks. According to our criteria, Agarwal et al.'s work tackles the heterogeneity challenge well but neither scalability nor federation challenges were tackled, this limit the deployment of fog computing and obviously impact the both QoS and QoE.

Heil et al. [109] propose a context-aware federation approach for IoT devices to support user access, and connect and locate arbitrary devices according to their functionalities. The proposed approach utilizes the concept of Federated Devices Assemblies (FDX) for integrating real-word IoT devices into service federations. According to our criteria, Heil et al.'s things federation tackles heterogeneity and federations of things well in away that can improve the QoS, but this federations have not discussed on fog level, also the interaction among "things" is not monitored and facilitated with an approach to prevent the attacks and malicious behavior that can impact network's QoP. Mathlouthi et al. [110] present an approach which enables the composition of federated cloud based System of Systems (SoS) to work co-operatively in order to achieve common goals. An SoS constitutes several complex, heterogeneous, and autonomous systems deployed on heterogeneous cloud environments. In this work, the presented functional and non-functional requirements for IoT are considered to obtain best SoS composition and maintain the overall QoS. However, according to our criteria, Mathlouthi et al.'s SoS composition approach tackles the heterogeneity and federations of the devices very well, hence developed approach can deliver reasonable QoS. Nevertheless, neither the scalability nor cognition criteria are not tackled, also malicious behavior within the network is not tracked/monitored for better QoE and QoP.

Sun et al. [111] propose a Cloud-of-Things and Edge Computing (CoTEC) scheme for traffic management in multi-domain networks. CoTEC directs the traffic flow through service nodes. The authors assign a critical egress point for each traffic flow in the CoTEC network using multiple egress routers to optimize the traffic flow; this is known as Egress-Topology (ET). Therefore, the proposed ET incorporates traditional multi-topology routing in the CoTEC network to address the inconsistencies between service overlay routing and border gateway protocol policies. Furthermore, Sun et al. introduce several programmable nodes that can be configured to ease the ongoing traffic on the network and realign services

among other nodes in multi-domain networks. In regards to the above challenges, the federation is only met with edge-cloud, however the congestion and its implication in clouds or edges is not discussed, this can be a serious issue that can impact both QoS and QoE. Moreover, they seems to overlooking the heterogeneity (i.e., device's capacity) and scalability of the network. Also, in edge-cloud communications, the proposed approach does not monitored in term of malicious behaviors within the network, hence the QoP may impacted.

Lina et al. [112] propose a fog computing based resource allocation policy using Priced Timed Petri-Nets (PTPNs). A simulation was developed to evaluate the proposed resource allocation strategy using parallel machines and Linux cluster. The outcomes were that the proposed resource allocation policy can provide efficient resource selection for autonomous task scheduling and improve the use of fog resources. The limitation of this work is the small-scale context related to online shopping, and the process of resource allocation is not automated calling for user assistance, hence this propsed work is not scalable. While the authors meet the federation criteria in term of considering different forms communications among nodes, the heterogeneity and cognitivity are not met.

Al-Turjman et al. [113] propose a cognitive caching approach for the future fog that focuses on the data exchange in Information Centric Sensor Networks (ICSNs). It depends on functional parameters, such as the age of information and data fidelity, to assign a value to the cached data while retaining the most valuable one in the cache for a prolonged time period. This enables the significant availability of the most valuable and difficult to retrieve data in the ICSNs. According to our criteria, Al-Turjman et al. overlooked the issues of heterogeneity in term of device's capacity, also how the proposed work can cop with expanding of the network, thus the scalability left unmet. Jalali et al. [114] propose a cognitive IoT gateway based approach supported by cognitive analytic and machine learning to improve the performance of IoT-enabled applications. The proposed approach enables the IoT devices to automatically learn and decide whether and when to run an application on the cloud or on the fog. Jalali et al. does not discuss how the proposed approach and the cognitive capabilities can handle/impacted with the encrypted data traffic on the gateway, which may affect services reliability, hence the QoS. Also, authors seems overlooked the challenges of scalability when the network expanding.

2.5.2.2 Fog Security: Trust and Privacy

In recent years, trust-based security solutions have been the focus of both industry and academia. Trust can help in detecting and isolating those malicious entities which are part of a network using legal identities. Moreover, trust plays an important role in nurturing the relation between different fog nodes in terms of maintaining user privacy and information security [115]. Ideally, fog clients are expected to connect to any arbitrary fog node to avail its services such as computation, storage and processing, with a belief that the provided information is not to be misused. The integration of trust management in fog computing will assist fog nodes to select the most secure and trustworthy fog nodes in the vicinity according to their needs and requirements. For achieving this, all the participating fog nodes should have a certain threshold of trust in each other. However, the development of a trust management mechanism for fog nodes is tricky due to its decentralised architecture, which may pose some challenges. The main issue with the decentralized architecture is that it makes collection and management of evidence and behaviour difficult which is required for the evaluation of the trustworthiness of distributed fog nodes [116].

Abedin et al. [117] propose resource sharing among fog nodes by defining a utility metric for these nodes that accounts for the communication benefits in case resources are shared among them. First, the authors determine an organised list of preferences that pairs fog nodes with each node. Then, each fog node requests pairing with its preferred fog nodes. At the reception side, depending on the preference and benefits of the previously received requests, a target fog node decides either to accept or to reject the request. The limitation of this work is that the pairing decisions are made based on communication cost without considering time and location. The authors do not also take the QoS, in term of latency and bandwidth, into the consideration as part of the resource sharing decision. Abedin et al. considers the heterogeneity of nodes, as the resource limitation of fog nodes (e.g., CPU) has been considered during load allocation. However, the evaluation is conducted over a small scale, making the scalability limited and hence, not met. In addition, federation is not relevant for this context, since the developed algorithm targets a single fog domain.

Fog-based trust management is in its inception as there have been very few reported works on the topic of the trust mechanism in fog computing. Alrawais et al. [118], carried

out a survey for finding the current security issues and challenges in IoT and propose a fog-based security mechanism to improve the distribution of certification revocation information between IoT devices. Wang et al. [119] propsed a concept of fog-based hierarchical trust-based mechanism for SDN, which has two distinctive features that are based on trust in the network structure, and the trust between cloud service providers (CSPs) and sensor service providers (SSPs). They focused on the packet loss rate, route failure rate and forwarding delay only. According to our criteria this work did not investigated the affect of nodes heterogeneity nor the scalability with network's expanding in term of size and functionality, which can be a major drawback. The proposed trust-based mechanism for SDN may have discussed how to achieve better QoS and QoP, but not the QoE. Elmisery et al. [120] proposed a fog-based middleware where trust between a fog node and the cloud is calculated in a decentralized fashion using entropy definition. The entropy may help in improving the QoP among nodes, but not the QoE and QoS as it can be a time consuming, also this approach will not scale well with network expansions. The authors in [121] proposed a fuzzy trust-based model that take into account experience and plausibility for securing vehicular networks. To ensure the correctness of information collected from authorized vehicles, a series of security checks are performed. Moreover, a fog -based facility is used to evaluate the level of accuracy of event's location. The limitation of this work is that the pairing decisions are made based on communication cost without considering time of processing nor federation, hence the heterogeneity and federation criterias are overlooked.

Many trust-based models have been reviewed thoroughly in the literature [43, 122, 123]. Reputation is considered as an important parameter for the evaluation of trustworthiness. Hence, there are many mechanisms which employ this procedure for evaluating the trustworthiness in mobile ad-hoc networks [124] and vehicular ad-hoc network [125], delay-sensitive networks [126] and mobile crowd sensing [127]. However, they seem to have overlooked the heterogeneity of nodes and scalability. Kai Hwang with his team presented the idea off trust in clouds, in which he suggested to combining security-based data centres, data access and virtual clusters driven by reputation systems [128]. The work of [122] represents a trust mechanism using a points based technique for protecting against unauthorized entry. For securing data transmission between two devices, trust was used in the gateway devices.

Jiang et al. [129] proposed an Efficient Distributed Trust Model (EDTM) for things. They randomly calculated direct trust values and recommendation trust values by evaluating the number of packets received by the sensor node. This approach is helpful in identifying different types of attacks. However, it is susceptible to processing and communication overheads, also they have overlooked the heterogeneity and scalability criteria. The work of [43] integrates the cloud and edge computing trust evaluation mechanisms which resulted in the considerable reduced resource usage for the evaluation of trust and increased IoT-cloud services efficiency. In this approach, they employed mean trust value, calculated on the basis of observed values obtained from the interacting devices. This may lead to communication overhead in the network.

The realization of offloading among fog nodes achieve resource efficiency and avoid bottlenecks, and overload [130]. There exist several mechanisms in the literature that focuses on the issue of offloading requests in a fog computing environment. However, they do not consider trust as a primary metric when it comes to offloading requests among fog nodes [131]. Bonomi et al. [132] proposed a fog computing module that brings the fog computing power and resources closer to the mobile users through an offloading policy. The policy takes into account execution, energy and other expenses. Fricker et al [133] proposed an analytical model to analyze a simple offloading strategy under heavy load for data centres in fog computing. The model considered forwarding requests with a certain probability to neighbouring data centres when the originally intended data centres is overloaded. Moreover, requests can be blocked/rejected based on whether it can offload the arriving requests to other data centres. Zhang et al. [97] proposed an analytical framework to support fair offloading among multiple fog nodes while maintaining low delay. It selects fog nodes to offload tasks based on a fairness metric and rules that minimize the task delay. Massri et al. [106] presented a collaborative fog-to-fog communication algorithm that allows fog nodes to communicate and coordinate with each other to process IoT job requests. However, most effort seems to have overlooked the heterogeneity of devices, thus treating node with same capacity which is a major drawback that can impact the QoS and QoE. Also, the scalability with IoT network expansion and federation among nodes for better QoS are overlooked. Nevertheless, none of these works have investigated, or invested in, the ideal location of fog in the network to provide cognitive capabilities and learning activities within fog nodes.

2.5.3 Gap Analysis

Despite the growing interest in fog computing for IoT-enabled applications, there are not, to the best of our knowledge, dedicated works that particularly examine a systematic framework for fog nodes coordination. Such a framework should be elastic in a way that can meet the intelligence communication among fog nodes, along with tackling the heterogeneity, scalability, and security criteria in Section 2.5.1. Several efforts tackle the challenging criteria of fog computing, however, most proposed approaches in the existing literature only pay attention to one criterion and overlook the other criteria, and this significant issue as other criteria may be in conflict with what has been proposed. Thus, a concrete solution for fog computing is still absent, and hence we identify the research gaps.

Many research studies have proposed theoretical solutions that have not yet been evaluated, especially those that are related to fog cognition and security, or they overlook some other criteria. Hence, from a closer analysis of the literature, we found that the engineering approach of fog computing is not considered in different aspects of architectural stability for different systems that have different types of tasks on fog nodes, such as *heavy-tasks* and *light-tasks*. Therefore, this thesis serves as a starting point to jointly handle the criteria in Section 2.5.1. Table 2.3 shows a comparative analysis of the work proposed in this thesis with other research. The comparison is based on main objectives and criteria set for fog computing scope, such as QoS, scalability and security, along with appropriate fog resource management and availability.

Table 2.3: Comparative analysis with main recent research studies

Research	Scope and Research Objectives								
	QoS	Latency	Security	Availability	Scalability	SSLA	QoE	Resource Plan	Cognition
Deng et al. [63]	✓	✓	–	–	✓	–	✓	✓	–
Al-khafajiy et al. [29]	✓	✓	–	✓	–	–	–	✓	–
He et al. [134]	✓	–	✓	–	–	–	–	–	–
Yannuzzi et al. [72]	✓	–	–	✓	✓	–	–	–	–
Chen and Hao [135]	✓	–	–	–	–	–	–	✓	–
Giang et al. [136]	✓	–	✓	–	✓	–	✓	✓	–
Al-Turjman et al. [113]	–	✓	✓	–	–	–	–	–	✓
Sarkar et al. [137]	✓	✓	–	–	–	–	✓	–	–
Skarlat et al. [138]	✓	–	–	✓	✓	–	–	✓	✓
Gupta et al. [139]	✓	✓	–	–	✓	–	✓	–	–
Shen et al. [140]	–	–	✓	–	✓	–	–	–	–
Wen et al. [141]	✓	–	–	–	–	–	–	✓	–
Liu et al. [142]	✓	–	–	✓	✓	–	–	✓	–
Bhardwaj et al. [143]	–	–	✓	–	–	–	–	✓	–
Wang et al. [144]	✓	✓	–	✓	✓	–	–	✓	–
Jalali et al. [114]	–	–	–	–	–	–	–	✓	✓
Vallati et al. [145]	✓	–	–	–	–	–	✓	–	–
Azimi et al. [146]	✓	–	–	✓	✓	–	–	✓	–
Markakis et al. [147]	✓	–	✓	–	–	–	–	✓	–
Chen and Xu [148]	✓	–	–	–	✓	–	✓	–	–
Ni et al. [149]	–	✓	✓	✓	–	–	–	–	–
Lina et al. [112]	✓	✓	–	✓	–	–	–	✓	–
Sun et al. [111]	–	✓	–	✓	✓	–	–	✓	✓
Agarwal et al. [108]	–	✓	–	–	✓	–	–	✓	–
Beate et al. [107]	–	✓	–	–	–	–	–	✓	–
Heil et al. [109]	✓	–	–	✓	–	–	–	✓	–
Abedin et al. [117]	–	–	✓	–	✓	–	–	✓	–
Elmisery et al. [120]	–	–	✓	–	–	–	✓	–	–
Jiang et al. [129]	✓	–	✓	–	–	–	✓	–	–
Proposed CF	✓	✓	✓	✓	✓	✓	✓	✓	✓

2.6 Chapter Summary

This chapter introduced a background on the basic concepts of IoT, cloud computing and fog computing along with a comparison between the two computing paradigms and their existence in the IoT era. Then, the chapter reported on a systematic literature review on fog/cloud challenges with a critique of some of the recent research efforts. After studying the literature, this chapter provided a gap analysis to highlight the research opportunity that forms the contribution of this thesis. The literature survey aims to provide a comprehensive overview of the current state-of-the-art and connect knowledge on fog, cloud computing, and more on fog since it is the main focus of this thesis.

The outcome of the background study and systematic literature review revealed that the emerge of IoT, on one hand, brought significant advantages to our lifestyle by utilising the network of sensors and smart objects. However, one the other hand, brought unavoidable challenges that vary from network congestion and resource managements to security and privacy challenges imparted by the heterogeneous nodes in the IoT network. Moreover, The emerge of fog computing with IoT deems to offer valuable services to help in network managements the lowering the response time to IoT application in comparison with cloud computing. However, there were no concrete solutions on fog computing can be adopted, manged, and what services fog nodes can provides. Therefore, next chapter proposes a full fog computing design principles and preparations for a concrete adoption of fog computing.

CHAPTER 3

Design Principles and Preparation

True stability results when presumed order and presumed disorder are balanced.

Tom Robbins

3.1 Introduction

In most recent IoT systems, there is a bridging point, called *fog computing*, between IoT things and the Internet (i.e., cloud). This fog layer often just performs basic functions such as translating between the protocols used in the Internet and the smart objects that are deployed on the Internet, such as healthcare wearable, along with providing other basic data storage and processing services (e.g., data filtering and aggregations). Fog computing offers the ability to extend not only storage capabilities but also networking and computing capabilities of the cloud to the edge of the network. The better positioning of fog nodes within the network in relation to fog connectivity with end-devices boosts its functionality and abilities, especially for systems that require data synchronisation with low latency (i.e., real-time systems), such as patient monitoring systems.

Fog nodes have the advantage of obtaining beneficial knowledge and constructive control over both the things network and the data transmitted over the network due to its strategic position within the network. This enables fog nodes to not only act on the data but also make intelligent decisions. Therefore, in this chapter, we first highlight the adopted network architecture of fog computing, secondly discuss the design principle and requirements of fog computing, and thirdly propose the concept of Cognitive Fog (CF) that forms the fog node functional and non-functional requirements which are used throughout this thesis. Briefly, the architecture of the fog computing network will give insight into how the network components are connected and the integration of different technologies within the network topology. The design principles highlight the main requirements of fog nodes based on networks communications and geo-location of the fog nodes along with their functional and non-functional requirements. While the concept of CF will discuss the opportunities of fog nodes within the network, in terms of what can be achieved by employing fog, that's includes the processes of not only acting on the gathered data but also learn from them and make decisions.

3.2 Fog Networks Architecture

Before we dive into the Cognitive Fog principles and requirements, we highlight the adapted fog computing architecture[1]. This architecture is similar to other large-scale computing architectures (e.g., cloud computing) which have either application specific architecture or application agnostic architecture. However, there is no standards architecture for the systems that are using fog computing [1, 2, 50, 150] as a data processing mediums. To elaborate this more, as reported by the National Institute of Standards and Technology (NIST) [151] there is no consensus exists on distinction architecture (distinct from cloud computing) and clear adoption of fog computing, hence no standards architecture on how fog computing can be adopted. The general fog computing architecture adopted in this thesis is in-line with the architectures presented in [1, 2, 4, 14, 39] which were initially introduced by CISCO [41]. Understanding the fog computing architecture topology helps obtain a better insight into the main functionality and benefits of using fog computing, also the advantages of its locations within the networks. Figure 3.1 (recall from Chapter 1) shows the IoT fog architecture composed of *things*, fog, and *cloud* layers. The main layers in IoT-fog architecture are: *thing*, fog, and *cloud*. The bottom-most layer (*thing*) comprises of end-devices, gateways and sensors. The middle layer (fog) is where the fog nodes reside along with the core network. The top-most layer (*cloud*) is where the cloud components are located for historical data storage and big data processing. Below is a description of the three layers.

Thing layer: also called *perception* layer, this is the closest layer to the users and physical sensor's surrounding/environment. This layer involves the connected IoT sensors, such as ambient sensors, heart-rate and blood-oxygen sensors and so on, this to ensures data availability by hosting the networked devices and enable data sensing and sharing. These sensors and devices are widely distributed over this layer and their main responsibility is to sense the featured data from the physical surrounding objects or events and transmitting these sensed/generated data to the upper layer (e.g., fog layer) for processing. Each device/sensor has a communication protocol, such as IEEE 802.15.4, WiFi, Bluetooth, MQTT, and so on. This communication protocol is essential for the things so they can transmit the generated/sensed data to other layers in the form of a data-processing request.

[1]It worth noting that this architecture is used throughout this thesis unless otherwise specified

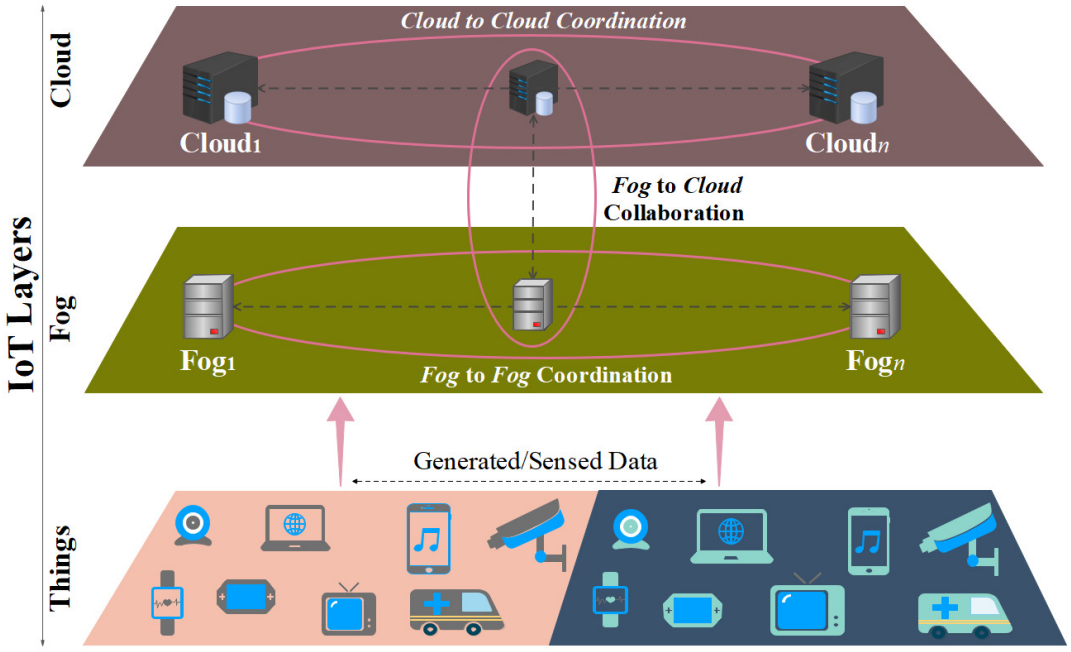


Figure 3.1: IoT fog architecture composed of *things*, *fog*, and *cloud* layers

Fog layer: this is located at the edge of the thing's layer. This layer contains a large number of distributed fog nodes [39, 46] that should ideally be located "next" to data sources (i.e., things). The fog node can be any device with: i) communication protocol, ii) processing and storage capacities, and iii) physically located close to the data sources. This generally include, but is not limited to, access-points, gateways, routers, switchers, base-stations, or specifically developed fog nodes [46]. This layer refines and processes the data that are generated/sensed at the things layer. The fog nodes are deployed at the edge of networks [1] to ease and make fast the process of data acquisition and processing. Thus, each fog node is equipped with on-board computational resources, data storage, alongside network communication facilities to bridge things and cloud within the IoT network [14]. Fog has the potential to reduce the amount of things' data transmitted to the cloud layer by acting on these data. The real-time analysis and latency sensitive applications can be accomplished in the fog layer. The fog nodes communicate with things to accumulate and process data conveniently and independently [1]. Moreover, fog nodes are also connected with cloud, hence if no sufficient resources are available at the fog layer, its responsible for interaction and cooperation with the cloud to obtain more powerful computing for rich-functions and services as well as more storage capabilities.

Cloud layer: this layer provides a global or centralised monitoring and control over the network. This layer consists of multiple high performance servers which enable omnipresent, convenient, and proper network access to shared resources, such as storage, rich-functions and service over the network. This layer has powerful computing and storage capabilities to support extensive computation analysis and permanently storage of an enormous amount of data. Cloud data centres are ideal for for big data processing and advanced machine learning activities, such as training and analysis.

3.2.1 Network topology

IoT network with fog computing carries out the service's communication, computation and storage at the edge of the network, is the most basic characteristic of fog computing and the most significant advantage compared with other traditional computing models. Therefore, stranded network topology has evolved to have the fog layer as the main point of communication to the thing's service requests. In the following, we define an IoT network as:

$$IoT = \{T, F, C, L\}$$

where:

- T is a set of things $\{t_1, t_2, .., t_n\}$; t_n is a 3-tuple format $\langle n, t_y, d\rangle$ where n is a thing identifier (e.g., IP address), t_y is a thing's type according to the packet's payload size [2] generated from the thing (e.g., *heavy-packet* and *light-packet*), and d denotes the service request's destination fog, from the t_n to the nearest fog node (i.e., the first fog node that receives a service request from t_n) within the fog layer, and this is subject to change according to the shortest distance between t_n and the fog node in the fog domain, It worth noting that the shortest distance is identified during the design time of the network, and in case of mobile things, things will have to periodically run a protocol to check the closest fog nodes coverage (similar to access point).

- F is a set of fog nodes $\{f_1, f_2, .., f_i\}$; f_i is a 4-tuple format $\langle i, \ell, s, \hbar, r\rangle$ where i is a fog identifier (e.g., IP address), ℓ denotes a fog node location, s and $\hbar$ refer to services (e.g., image processing) provided by the fog node and hardware capabil-

[2]The payload size of 1024 Bytes can be transported without any fragmentation through a normal not constrained network; otherwise, it is fragmented into lighter tasks [152]

ity (i.e., CPU frequency) of the fog node, respectively, and r is a set of all "reachable fogs" from f_i.

- C is a set of cloud nodes, each c_i is defined using a 3-tuple format $\langle i, \ell, s \rangle$ where i is the cloud identifier, ℓ denotes the cloud location, and s denotes the cloud services (e.g., processing and storage).

- L is set of communication links among the thing, fog and cloud layers, such that L is:
  $L \subseteq \{\langle n, \grave{n}, q \rangle | (n, \grave{n}) \in (T,T)(T,F)(F,F)(T,C)(F,C)(C,C)(C,F)(F,T)(C,T) \wedge (q \in Q)\}$

  This means, L is a sub-/set of available links between $Thing \longleftrightarrow Fog \longleftrightarrow Cloud$. Each link is associated with its q from Q set, which refers to the QoS properties (e.g., upload $b_{\uparrow}$ and download $b_{\downarrow}$ bandwidth).

3.2.2 IoT services requests workflow

The standardised approach in which service requests are made in the IoT systems (with a fog layer) is as follows: t_n generating, sensing and/or gathering data periodically from the surrounding environments and sending it to either the fog layer or the cloud layer for processing and/or manipulation. In the fog layer, f_i can serve t_n's request instantly or offload it to another fog node (e.g., $f_{i \in r}$) in the same domain if f_i is congested and may delay processing t_n's request. To this end, f_i (or $f_{i \in r}$) responds back to t_n and reports to cloud c_i for data archiving. Similarly, when packets are sent to c_i, it will be processed at this level and a response goes back to t_n. It is worth noting that the importance of the fog layer location (i.e., in-between thing and cloud layers) makes fogs more accessible/reachable for both things and clouds. Therefore, fog can be used/operated horizontally (i.e., $\mathcal{F}$og-2-$\mathcal{F}$og) and vertically (i.e., $Thing \longleftrightarrow Fog \longleftrightarrow cloud$) in the network to provide the desired IoT services with high QoS and QoE. However, thesis main focus is on processing service requests dispatched from the things to fogs, in which the latter can adapt to different workloads by evolving the propose fog cognition features and the $\mathcal{F}$og-2-$\mathcal{F}$og coordination for efficient data processing.

3.3 Design Principles and Requirements

Understating the fog computing design principles and requirements is essential to gain the most efficient services of fog computing. The following subsections will highlight the main design principles of fog nodes based on networks communications and geo-location where nodes are planted. In addition, the fog computing functional and non-functional requirements are presented as performance requirements (i.e., functional requirements) and security requirements (i.e., general non-functional requirements). It is worth noting that these requirements and the design principles are adopted and taken into account when implementing the fog system in the following chapters.

3.3.1 Design Principles

In this section the design principles of fog computing are discussed. These are general fog node design principles that have been proposed according to the requirements that needs to be satisfied by the practical constraint of fog nodes, such as nodes resource managements, communications, traffic management as well as fog services related constraints. Fog computing involves the on-board components of data-processing and/or analytic, networks communication channels as well as software applications running on distributed nodes [47]. Therefore, when designing a fog computing network, a set of aspects and facilities that manages the networking, storage and services needs to be considered.

The fog manages the cooperation between data-centres and end-devices (i.e., things) for data storing and processing. In addition, it supports user mobility, resource and nodes heterogeneity as well as distributed data analytics to address the requirements of widely distributed applications that need low-latency [39]. Hence, fog nodes generally use the sense-process-actuate and stream-processing programming models [47]. In such model, things (e.g., sensors) stream data to the IoT networks, applications/services that are running on fog nodes will subscribe to process the data, and the resulted/refined data are translated into actions sent to actuators or to storage in the cloud for future uses. Fog nodes are dynamically discover and use Application Program Interfaces (APIs) to build new complex functionalities [47] and/or separate their current services to improve the scalability. Moreover, fog node's resource management is a big concern when designing the fog networks.

Hence, the resource management processes uses information from the resource monitoring service to track the state of available fog nodes and/or clouds to identify the best candidates to process incoming tasks. With multi tenant applications, the resource-management components prioritize the tasks of the various participating users or programs. Therefore, there are four main designing principles that should be taken into account when designing fog networks and services:

1. Fog nodes hardware components: since fog nodes can be anything, such as access-points, routers, base-stations, or specifically developed fog device, the hardware components can significantly vary. Nevertheless, in principle each fog node should have some requisite components such as a CPU for data processing, RAM for temporarily data storing, and disk for longtime data storage. In addition, each node should have a pre-defined software that features or operates the hardware components such as operating system with a resources managements software.

2. Fog nodes Locality and Geo-distribution: fog nodes normally act as a bridging point between the data sources (e.g., sensors) and data processing mediums (e.g., cloud, local servers, or fog itself). since the things (e.g., sensors) are widely spread over different geographical locations, the fog computing services should also have equal spread to provide efficient functional ties to these things and be able to serve end-users with reliable services. Hence, this will require a large number of planted fog nodes available in the networks compared to the number of cloud servers.

3. Fog nodes communications: the communication links of fog nodes are mainly with things and cloud. Thus, in fog nodes communication a machine-to-machine (M2M) standard is used, such as MQ Telemetry Transport (MQTT) and the Constrained Application Protocol (CoAP). In addition, the adoption of a Software Defined Networking (SDN) to help with the efficient management of heterogeneous fog networks.

4. Fog resources planning: resource planning becomes a critical issue in fog computing due to the vast amount of data that things provide every second. Hence, a proper strategy for efficient resources planning needs to be in place when designing a fog network. This includes the processes of estimating the correct number of fog nodes

which need to be installed in the network. In addition, some resource sharing features to allow the fog nodes to share the load and reduce the traffic load from a congested fog node.

3.3.2 Fog Performance Requirements

The most basic requirement/characteristic and the most significant advantage of fog computing is that the service's capabilities are in close proximity to end devices/users in comparison with other traditional computing paradigms. Fog computing aims at supporting things functions by performing the tasks of computation, communication and storage at the edge of the network to address requirements of services that are latency sensitive with a wide and dense geographical distribution. Furthermore, the main performance requirements/characteristics of fog computing are listed as follows:

1. Low latency and real time interactions: the good location of fog nodes within the network most significantly reduces the time of data-movement across the network. Also, provides high quality localised services supported by endpoints. Hence, it enables low latency that meets the demand of real-time interactions with things (e.g., sensors), especially for latency sensitive or time sensitive applications and services.

2. Service Availability: fog computing services availability means that the services must be available when required. Unexpected situations, such as service crashes, would significantly affect service availability. Hence, the fog should be able to tolerate any attacks that aim to crash the fog services or divert them. It is worth noting that the service distribution among fogs helps in enhancing services availability.

3. Scalability: this is a very essential requirement when designing fog networks as its connected with both big data and the geo-distribution of both fog and things nodes. Thus, the network scalability is the ability of fog computing to handle the growing number of service requests (i.e., tasks) sent from the things in both processing and storage capacities. Also, the potential of fog to be easily enlarge in order to accommodate the continuous growth.

4. Save bandwidth: is one of the significant advantages of fog computing. Since fog

allows the data processing and storing at the edge it reduces the amount of the data transmission over the the network, hence reducing bandwidth. Also, in some services the decisions are completed within the fog layer, rather than completed by the cloud, therefore fog computing will save the bandwidth effectively compared to the cloud. This bandwidth saving advantage is more and more becoming significantly effective with the increasing in the amount of data in the network.

5. Support for mobility: this for both things mobility and fog mobility. The support of mobility is by providing adequate communication technologies to ensure the continuous data sharing and processing. An example of a mobile thing is robots, while the mobile fog is smart vehicular.

3.3.3 Fog Security Requirements

A number of security measures and requirements need to be taken into account when designing a fog computing network in order to enable a secure fog networks that provide a secure environment for the running services and applications. This secure fog computing environment will enable fog nodes to securely outsource services, resources as well as data sharing across fog nodes. Therefore, the following security requirements should be fulfilled when designing a fog computing network. These security requirements are defined as Requirements of Protection (RoP) which are a set of security requirements that includes all the security factors required to deliver the desired services securely and efficiently. Thus, RoP defines and measures the Quality of Protection (QoP) among fogs. It is worth nothing that the following RoP are mainly focused on the security and trustworthiness within the fog layer. Hence, the more RoP are met, the better is the QoP.

1. Location and identity: fog responses to any collaboration requests from other fogs should be based on an authentication process, such as fog's identity and location. The fog should be trusted by verifying the identity of fog nodes within the fog domain and identifying whether the provided fog location is real or fake before it an access the desired services.

2. Service integrity: since the transmitted service packets among fog nodes can be changed during the transmission time by malicious fogs, the packets must be checked

so that they completely match what was sent initially (such as packet authentication from source). It is worth noting that the fog might be legitimate for collaboration, however the service packets could contain fabricated data, and thus, the bigger the distance between collaborating fogs, the higher is the risk of packet attacks. Hence, the packets that are generated in a closer-distance and short-time are more reliable than packets arriving from long-distance and generated a long-time ago.

3. Confidentiality: the confidentiality in the fog-2-fog collaboration refers to data confidentiality. Since data packets are shared among fogs, the data may contain sensitive information, such as personal details (e.g., bank details), therefore, such confidentiality can be achievable by adopting public or symmetric key encryption to assure the security of the communications. Thus, the encryption of data prior to sharing is required to keep data secret and unreadable for distrusted or malicious fogs, and only trusted fogs can have the correct decryption key for the shared data.

4. Trusted fog: the fogs trust each other based on past experiences obtained upon the fog's coordinations. The ability of selecting the trusted fogs in a domain will helps in providing the desired fog's services with high quality, hence both Quality of Experience (QoE) and Quality of Protection (QoP) will be fulfilled. Moreover, the trust between fogs is:

    - Dynamic: the trust between fogs is dynamic and not static, so that fog_a trusts fog_b at a specific timestamp (e.g., t_1), however fog_a may distrust fog_b at t_2 due to two reasons; i) fog networks topology is continuously changing by adding or removing nodes from the fog domain. ii) fogs within the domain may alter their behaviour due to malicious attacks (e.g., Denial of Service). Therefore, periodic trust assessment is essential.
    - Subjective: fog nodes may have different security measures for different types of processing so they meet the QoP. For example, fog_a can trust fog_b to carry out processes for traffic data, however, fog_b is not trusted enough for fog_a to process healthcare related data.
    - Asymmetric and not transitive: each fog node has its own RoP that defines its QoP. Hence, the RoP properties that one fog adopts can vary from one

fog to another, so that, if fog_a finds fog_b is trustworthy, it is not necessarily that fog_b finds fog_a is trustworthy. Similarly, the trust is not transitive, for example, if fog_a trust fog_b and fog_b trust fog_c, it is not necessarily true that fog_a trusts fog_c.

3.4 Cognitive Fog Model

Fog nodes have the advantage of obtaining beneficial knowledge and constructive control over both the things network and data transmitted over the IoT network due to their strategic position within the network (i.e., in between things and cloud layers). This enables fog nodes to not only act on the data but also make intelligent decisions, for follow-up processes when required. The core concepts of Cognitive Fog (CF) and fog federations (so that CF can assist each other) is elaborated in this section. Before we dive into the details, we highlight the key definition of cognitive fog and fog nodes federation.

Cognitive fog advocates for fogs that can interpret the gathered/received data from things, hence CF learns and matches patterns in a way that mimics the process of cognition in the human mind [153]. CF can learn from their process experiences according to different situations/scenarios, then get better when performing the repeated processes. Therefore, fogs employ algorithms such as pattern recognition and data mining to boost the abilities rapidly and achieve better experiences on the repeated processes. In this thesis, the context of CF takes the same concepts of cognitive computing which can be defined according to DARPA definition of cognitive system, which is a system that can *"reason, use represented knowledge, learn from experience, accumulate knowledge, explain itself, accept direction, be aware of its own behavior and capabilities as well as respond in a robust manner to surprises"* [22, 153]

Fog nodes federations is about gathering multiple fog nodes to perform/achieve a specific task in a certain situation or scenario. Fogs become members of a federation because of their capabilities that permit them to satisfy the needs and requirements of the situation assigned to this federation for handling. Hence, fogs are to be described and discovered for federation and then selected for a particular federations according to *planned* and *ad-hoc* federations [22] based upon trustworthiness assessment for the fog nodes in achieving the

desired tasks. The *planned* and *ad-hoc* federations are as follows:

- Planned federation is formed at design-time, all its fog participants are already identified and ready to act according to a task's needs and requirements.
- Ad-hoc federation is formed at run-time, fogs are joined together according to certain occasions where each fog can empower the federation with various types of processing and controls that enhance the processes.

3.4.1 Fog Cognition

To allow fog to be cognitive, so that, it reasons about the surroundings, learns from the past, and adapts to changes, the fog nodes are featured with functionalities, such as pattern recognition, image recognition, and emotional intelligence that enable the Cognitive Fog (CF) to not only respond to events, but also interpret the surrounding activities in order to invoke further services/processes based on fog's judgments/interpretation that boost the cognition features of fog, hence improve the QoE ans QoS. In addition, CF features a computation/processing capability for task processing needs, resources for storage needs and communication abilities for networking and interactions. The operations over the CF run or interact with four connected worlds as per Fig. 3.2. The data world that is featuring the both raw-data (i.e., plan data from sensors) and filtered data (i.e., processed data), the process world featuring the processing models that acts on the data, the fog world featuring the CF processes and controls over all connected worlds, and finally the things world that contains the things nodes that are planted at the user level and controlled by CF to adapt as the environment changes, based on its sensed/gathered data.

CF either acts upon a things data or direct things to engage in continuous interactions that should, ideally, lead to achieving certain tasks, such as directing traffic upon congestion or accidents. Each CF has a number of functional and non-functional requirements that either permit service accessibility and allow CF to participate in the service's request processing and decision making processes or just to step-out the processes. Functional restrictions influence CF involvements in active processes in the process world due to limited availability (e.g., busy) and/or security restrictions (based on the trusts/recommendations

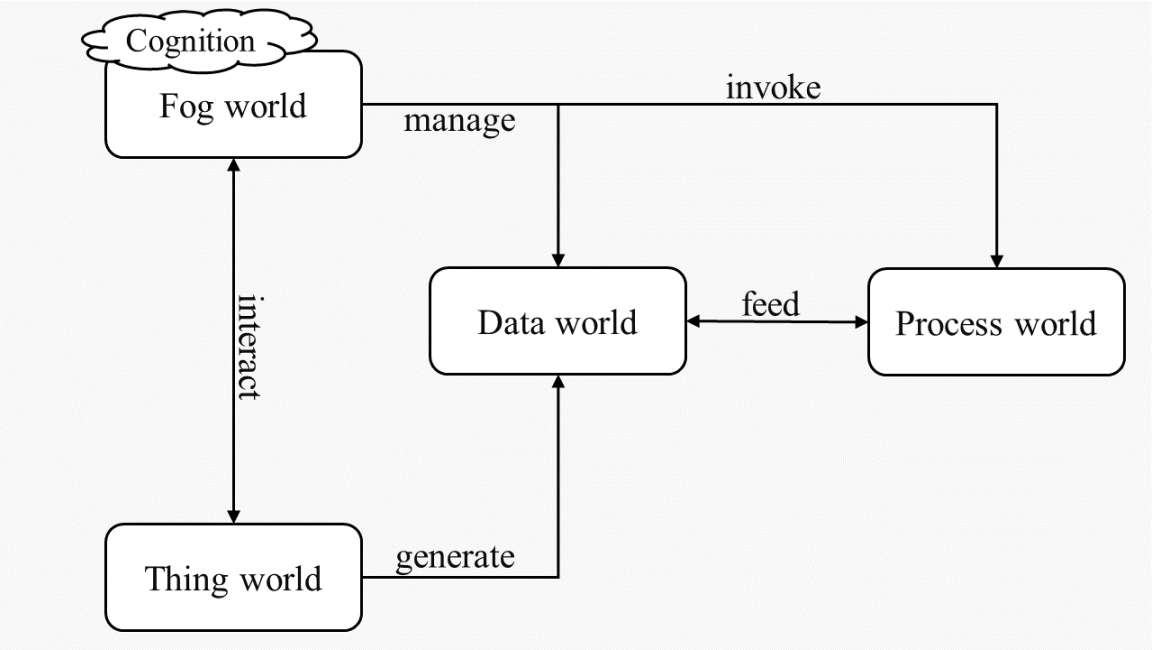


Figure 3.2: Interactions of the Cognitive Fogs

approach presented in Chapter 6). While the Non-functional restrictions, influence CF involvements in the active processes in the process world due to processing or storage capabilities and/or bandwidth limitations which associate with the type of data packets (two types of data packets are considered, light and heavy packets, more in Chapter 5). Therefore, participation considers a CF's functional and non-functional restrictions that, in fact, reflect this CF's current/active participation in other ongoing (under-execution) process which also influence CF participations with other CF processes during the planned or the ad-hoc federations (Discussed in Section 3.4.2).

The cognition anchored defined as a three stage cycle as per Fig. 3.3. In the first stage, the reasoning stage, all cognition activities are taken place to assesses the surroundings according to the received data from the data and things worlds. Thereafter and prior to any decision from the CF, it will check its functional and non-functional restrictions for any processes and/or participations within the new context that may impact the CF performance. In the second stage, the CF relies on both the thing's data and data in the data world to make some decisions and reasoning that could lead to CF participations in new process as well as some adjusting in its behaviours, such as executing additional processes (e.g., pattern and/or image recognition) to identify certain activity, then ideally, to redirect the

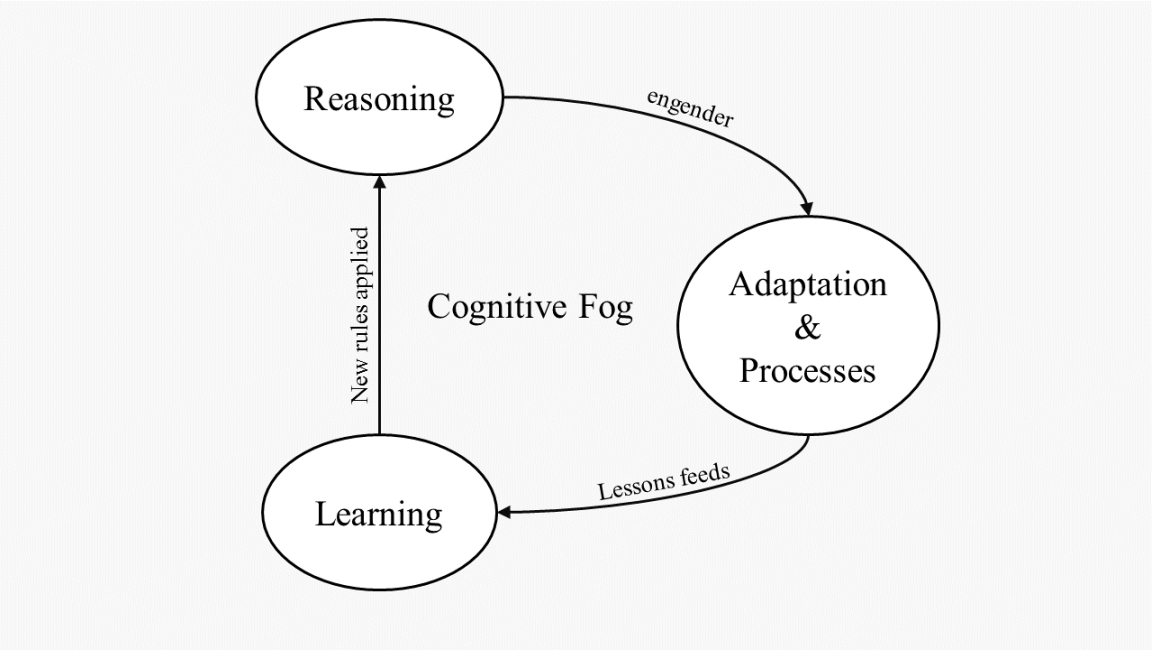


Figure 3.3: Cognition of the Cognitive Fog as a 3 stage cycle

connected things accordingly. In the third stage, the lessons learned during the adaptation and participation of CF will feeds into a learning process, such as making new rules/notes for its functional and non-functional, for example. To this end, all learning outcomes will will feeds into the reasoning stage that applies on the CF in it's future interactions.

3.4.2 Fog Federation

In order to model CF federations, understanding the insight of CF design is an essential step. In the proposed model, recall that each CF consists of set of 4-tuple, hence CF composed of $\text{CF} = \{i, t, c, l\}$. Where i denotes to CF unique identifier, such as, the IP address or a unique ID, t denotes to the type of CF (e.g., refers to the type of processes/jobs that CF is capable of), to elaborate more, since different CF can be equipped with different cognition features, t is use to distinguish CFs based on their functionalities/jobs, for example healthcare CF, Traffic CF, etc. While c denotes the total capability of the CF node, such as fog hardware limitations (e.g., CPU frequency), and finally l denotes the actual geographical location where the CF is installed. Thus, these CF tuples are used to define each CF in the network, prior to or during any federation. Fig. 3.4 shows both types of federations.

In planned federations ($\mathcal{PF}$), all CFs are known to each other as they are initially (i.e.,

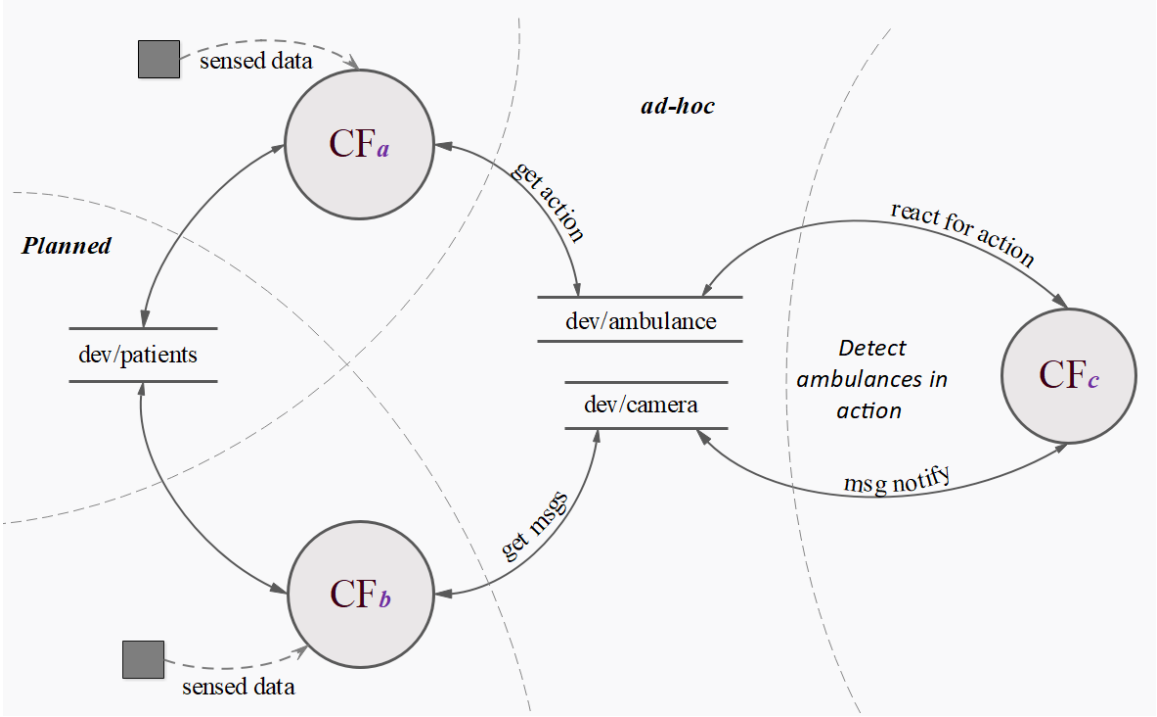


Figure 3.4: Planned and Ad-hoc federations

during design time) designed to assist, or take benefit from, each other. Thus, CF in a particular geographical area, having the same t (i.e., same type of processes/jobs) and l (e.g., within the same network domain), are designed to communicate with each other to deliver a single task. We can formulate the $\mathcal{PF}$ as:

$$\mathcal{PF} = \{\mathcal{CF}_1^{ts_1}, \mathcal{CF}_2^{ts_2}, ..., \mathcal{CF}_n^{ts_n}\} \tag{3.1}$$

Where ts refers to the tasks required from CF during the federation. For instance, the roadside of a highway supplied with a set of CFs to perform road monitoring task, such as traffic and accidents (known from data provided from things planted along the way). The CFs are connected to each other at the design stage, thus in this scenario, the planned federations occur when one or more CF has gone down for whatever reasons, active CF will federate to cover the role of the failure CF. In $\mathcal{PF}$, CF are usually connected to perform a specific task (e.g., road monitoring) and not multi-tasks.

On the contrary, in ad-hoc federations ($\mathcal{AP}$), the CFs are communicating with each

other based on a need (i.e., formed on the fly) and usually to perform different types of tasks (i.e., multi-tasks are achieved from the federation) according to a specific situation, for example, multiple CF can form a federation to detect and react upon a patient illness. Hence, within the $\mathcal{AP}$, multiple CF could perform one or more tasks according to the federation's outcome and its requirements, therefore, $\mathcal{AP}$ can be formulated as a 2-dimensional matrix of unlimited possibilities of $\mathcal{CF}$ communications according to tasks, as follows:

$$\mathcal{AF} = \begin{pmatrix} \mathcal{CF}_{1,1}^{ts_1} & \mathcal{CF}_{1,2}^{ts_1} & \cdots & \mathcal{CF}_{1,n}^{ts} \\ \mathcal{CF}_{2,1}^{ts_2} & \mathcal{CF}_{2,2}^{ts_2} & \cdots & \mathcal{CF}_{2,n}^{ts} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{CF}_{x,1}^{ts} & \mathcal{CF}_{x,2}^{ts} & \cdots & \mathcal{CF}_{x,n}^{ts} \end{pmatrix} \tag{3.2}$$

So that, one row could refer to the multi CF collaborating to achieve one task ($\mathcal{CF}_{1,1}^{ts_1}$ & $\mathcal{CF}_{1,2}^{ts_1}$ & $\cdots$ & $\mathcal{CF}_{1,n}^{ts}$), while CF on another row is working to achieve another task, and so on. Eventually, the total CFs in the federation (i.e., all rows and columns) will achieve multiple tasks.

3.5 Case Study and Testbed Setup

This section demonstrate the practicality of the proposed cognitive fog along with its two types of fog nodes federations, planned and ad-hoc federations. The work presented is based on a motivating healthcare case study. The testbed setup and implementation of the cognitive fog take the form of a feasibility study. Therefore, two main stages are included: i) the installation and experiment's configurations that shows the connectivity of different components, and ii) the performance evaluation upon the execution of the cognitive fog testbed.

3.5.1 Case Study - Patients Monitoring

Improving the efficiency of healthcare and biomedical systems is one of the considerable goals of modern society. The case study proposed in this chapter is about using fog computing in healthcare. Moreover, cognitive fog is used to monitor the health and activities of elderly people in care-homes premises, especially the people who need special care or continuous assisting and monitoring.

Consider an IoT healthcare system to monitor human symptoms data for patients with chronic diseases. This system is offered by a healthcare organisation to its patients in care-homes premises. The system is modelled to support real-time monitoring for patient activities. Thus, the system consists of smart healthcare wearables (e.g., heartrate, temperature and oxygen sensors), cognitive fogs, cloud and dashboard for caregiver and doctors to monitor the patient's symptoms. The fogs are responsible for getting real-time data from wearables and making a primary compute/processing on the gathered data for diseases detection. While the cloud data-centre is for data storage and future analysis (i.e., non-real-time processes) including the machine learning (ML) segment for data training and analysis activities.

The IT division experts install a few CFs according to the care-home size, with at least two CFs at any given location of their care-home premises. The reason for having at least two at each premises is to make sure that a backup fog node is always available in case one fog node went down. Also, in case one fog node is busy with the data processes of a

patient and more data is received from another patient, a planned or an ad-hoc federation is formed to handle the extra load from the congested fog node. Moreover, for patient's symptoms monitoring, we have focused on monitoring the pulse rate (i.e., heartrate) with either abnormal pulse racing or pulse dropping of a patient. To this end, two possible cases have been considered:

1. First time pulse rate is racing/dropping: CF will analyse the received data from the pulse sensor to detect/check for any abnormal racing or dropping in the patients pulse. For any suspicious situation, a planned CFs federation is formed, with respect to the rescue of CF non-functional requirements, to investigate the patient's status and make a decision based on federated CFs experiences with such situations. Once a decision is taken, the caregiver will be notified through the dashboard. Thereafter, every CF will log the set of learned lessons which could be used in the future federation of similar or repeated processes.

2. Recurrent abnormality detected: on a similar or repeated situation, the installed CFs will learn (i.e., make note of repeated actions) the conduct taken by the caregiver on such pulse racing/dropping situations, so that CF can automate the processes and take the action more quickly and on behalf of the caregiver, such as request an ambulance and notify the in-charge doctor(s) about the patient's status. In such a scenario, an ad-hoc CF federation is formed after selecting the necessary CFs (i.e., according to their functional requirements) with respect to their non-functional requirements to run multiple processes at the same time. For example, care-home CF communicating with CF of the nearest hospital to send an ambulance to the care-home address, also communicating with roadside CF to clear the way for the ambulance in advance to avoid traffic delays and congestion. Once the case is over, the ad-hoc CFs federation becomes a planned federation that could be initiated in the future, this should be titled under one task that has a similar or repeated situations that happen along with similar functional and non-functional requirements.

Our proposal is that CFs for health monitoring would reason about sensed data, such as pulse abnormality, time detected and the case of pulse racing or dropping, so ideally, CF will be able to act accordingly and form the appropriate federations to solve the issues.

3.5.2 Testbed and experiment configurations

Our CF testbed is shown in Figure. 3.5. The testbed was assembled using four CFs and three things (i.e., sensors) nodes. We assume that the CFs are located in these locations, i) in care-home premise (CF_1 & CF_2) where patients are normally based, ii) in hospital and/or A&E department (CF_3), and iii) on-street fog (CF_4) (located on roadside between the care-home premises and the hospital connected to traffic-light and CCTV things nodes). Moreover, the CF_1, CF_2 and CF_3 are a Raspberry Pi (RPi) device with a Quad Core 1.2GHz CPU and 1GB RAM. However, each with different functionality, according to our case-study, CF_1 and CF_2 are connected to a pulse sensor (SEN-11574) to measure heart-rate and temperature/humidity sensor DHT22 (AM2302) sensor, these are used as patient's sensors. While CF_3, is used for hospital processes, such as dispatching ambulance and contacting doctors according to the data received from CF_1 and/or CF_2. Finally, CF_4 composed of a Lenovo Ideapad laptop with i5 1.8GHz CPU and 8GB RAM connected to the Internet over Ethernet cable and fitted with an HD Lenovo EasyCamera Webcam (as a thing node) with 0.92MP resolution. In addition, CF_4 is connected to a traffic light node (as a thing node) which has two LED diodes (Green and Red) wired through the breadboard to the RPi.

The interactions between CFs themselves, and CFs with IoT things (i.e., sensors) are over publish/subscript protocol, that is Message Queuing Telemetry Transport (MQTT) protocol is been used, however, any other similar protocol can be used in this scenario (e.g., Kafka, RabbitMQ, etc). Moreover, via the subscribed topic, which is a UTF-8 string that the MQTT broker uses to decide on which client can receive which message, the subscribers of a specific topic will receive useful data in real-time. For example, the traffic light receives signals through the *"CF/traffic"* topic, upon which it changes to green or red.

During the ad-hoc federation, a CF will be responsible for communicating with the camera thing and the traffic thing to clear the way for an ambulance travelling from/to the hospital. Therefore, to detect ambulances, we developed an in-house Python image recognition program that processes RGB images using an Open Source Computer Vision (OpenCV) library. Upon ambulance detection by the CF, according to the live frames from the camera, it will send an alert to the traffic-light, to stop or redirect the traffic, over the MQTT protocol via *"CF/traffic"* topic to set the traffic-light sign.

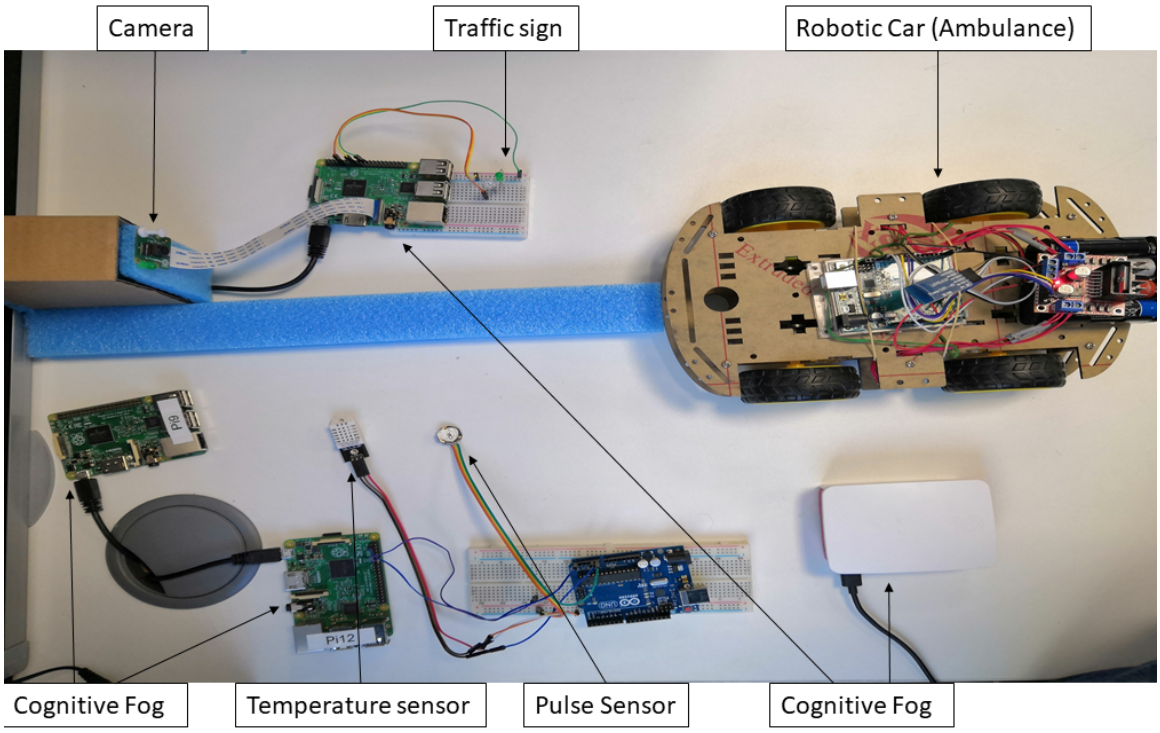


Figure 3.5: Cognitive Fog testbed

3.5.3 Performance evaluation

In our experiment, we employ the four CFs as follows: CF_1 and CF_2 is for interacting with the patient's things (i.e., the pulse and temperature sensors) as well as interpret the sensed data. The CF_3 is for alerting the hospital's A&E about the patient's situation, command for ambulance driver to go to the patient's address (supplied from CF_1 or CF_2). While, the CF_4 is for interacting and controlling things planted on the roadside (i.e., camera and traffic-light). The camera is for broadcasting live images of the road from/to the care-home and hospital, while the traffic-light node is regulating the access of the ambulance. For evaluation needs, two simulation scenarios were carried. It worth noting that these results are not comparatively evaluated, at this point, as these are just testbed setup evaluation results for the sake of preparation for next chapters. The scenarios are as follow:

Scenario $_1$: we considered a $\mathcal{PF}$ of two CFs, namely CF_1 and CF_2, upon needs after detecting an abnormality in the patient's pulse, thus experience of multi CFs is required to make a decision. The $\mathcal{PF}$ evaluated in terms of time-delay and efficiency

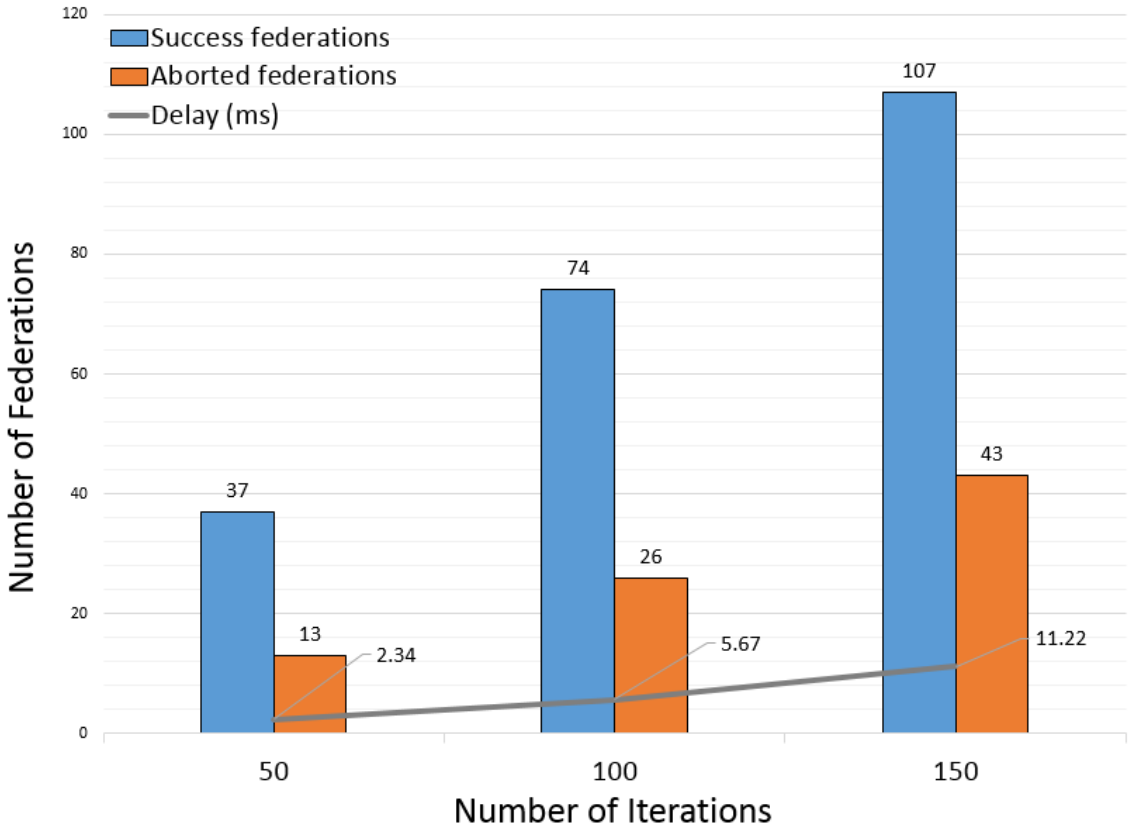


Figure 3.6: The Execution time for Planned Federations

in forming a such federation, therefore, we measured the total time required to form $\mathcal{PF}$ between CF_1 and CF_2 when the pulse sensor provides a reading that looks abnormal (i.e., 60Bpm $\geqslant$ pulse $\geqslant$ 100Bpm as in [105]). CF_1 interpret the sensed data from both pulse and temperature sensors to reason the measured data, thereafter, upon suspected values or abnormality, CF_1 will seek an assist from CF_2, forming a $\mathcal{PF}$ to make a decision for either alerting the caregiver or not. During the same execution life-cycle, we change the payload of sensed data and experience a different set of data across a number of iterations which have been grouped into 50, 100 and 150 iterations. The objective was to observe how the test-bed behaves with respect to the number of detected abnormalities and the time taken to make a decision including the time required to exchange number of messages between both CFs. Figure. 3.6 reports the performance results of the $\mathcal{PF}$ within the three iterations. It worth noting that the aborted federation in Figure 3.6 is due to some non-functional requirements.

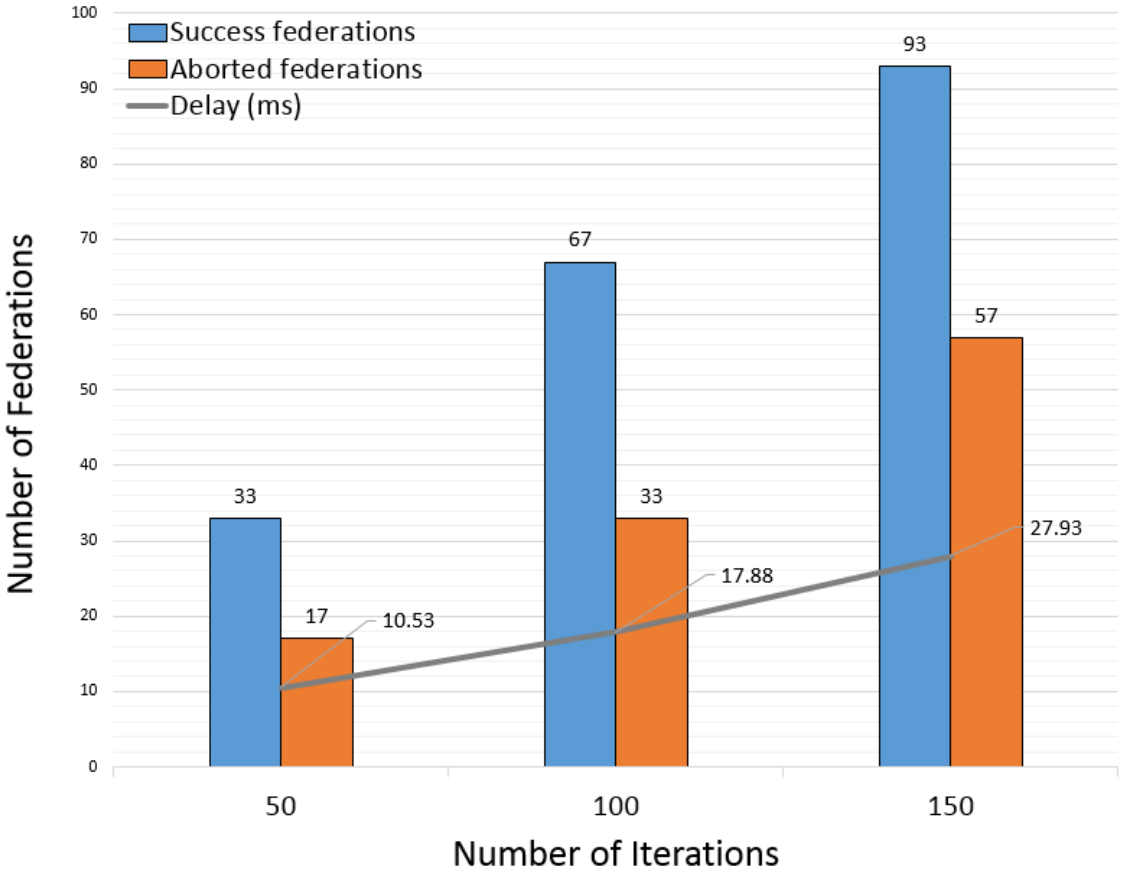


Figure 3.7: The Execution time for Ad-hoc Federations

Scenario $_2$: we expanded Scenario $_1$: to include all four CFs, namely CF_1, CF_2, CF_3 and CF_4. In this scenario both $\mathcal{PF}$ and $\mathcal{AF}$ are formed according to following: i) CF_1 detect an abnormality, in patient's pulse, and through a $\mathcal{PF}$ with CF_2 makes decision for requesting ambulance. ii) CF_1 will search for nearest hospital and communicate with its CF, in this case CF_3, and form an $\mathcal{AF}$. To this end, CF_3 will inform the doctor and send out an ambulance to the patient. iii) CF_3 will also form an $\mathcal{AF}$ with CF_4 to clear the path for the ambulance, upon detecting the ambulance via the camera thing, through controlling the traffic-light signals. The $\mathcal{AF}$ is evaluated in terms of time-delay and efficiency in forming the federations, thus, we measured the total time required to form an $\mathcal{AF}$ among all CFs. Figure. 3.7 reports the performance results of the $\mathcal{AF}$ within three iterations (50, 100 and 150 iterations). It is worth noting that the time-delay (in millisecond) for $\mathcal{AF}$ is higher due to the multi-tasks required from the federation, also, the aborted federation is due to some

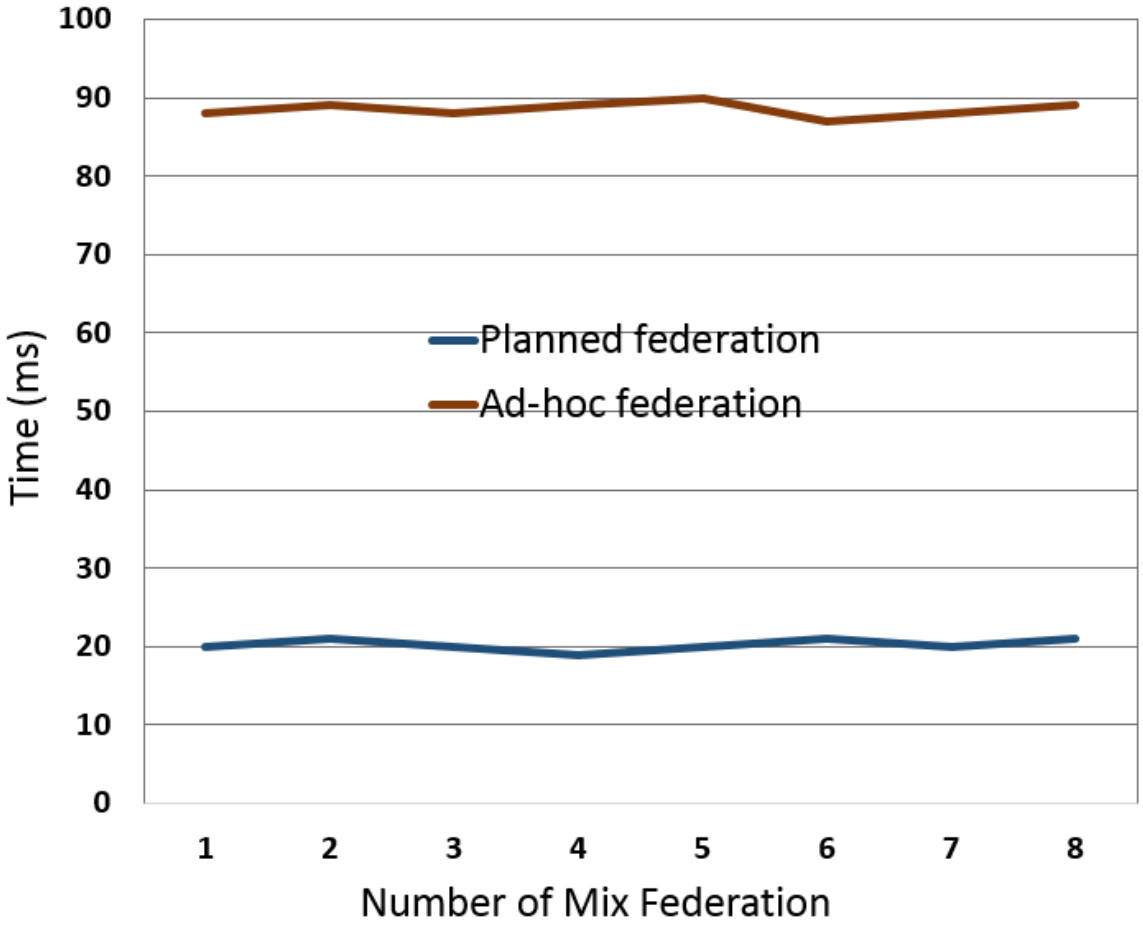


Figure 3.8: Execution time related to $\mathcal{PF}$ versus $\mathcal{AF}$ federations

non-functional requirements. Within this scenario, we checked how the test-bed behaves when the $\mathcal{PF}$ of things (i.e., pulse sensor) is merged with an $\mathcal{AF}$ federation to evaluate the execution/process time required to perform a collaboration. Figure. 3.8 illustrates the results showing cases of execution time related to $\mathcal{PF}$ versus $\mathcal{AF}$ federations; it took between 85ms and 90ms to execute an $\mathcal{AF}$ and between 18ms and 22ms to execute $\mathcal{PF}$.

3.6 Chapter Summary

In this chapter, a holistic fog computing architecture and design principles are presented. The architecture of the fog computing network give an insight on how the network components are connected and the integration of different technologies within the network topology. The IoT fog architecture is composed of three main layers; things, fog, and cloud layers. Things is the bottom-most layer, comprising of end-devices, gateways and sensors. Fog is the middle layer where the fog nodes reside along with the core network. Cloud is the top-most layer where the cloud components are located for historical data storage and big data processing. The topology of fog computing carries out the service's communication, computation and storage at the edge of the IoT network, thus this is the most basic characteristic of fog computing and the most significant advantage compared with other traditional computing models.

The design principles of fog nodes are highlighted based on the main requirements of fog in term of networks communications and geo-location of the fog where nodes are planted, along with their functional and non-functional requirements. The four main designing principles of fog nodes are; hardware components, locality and geo-distribution, communications, and resources management. The fog computing functional and non-functional requirements fit under the umbrella of performance requirements (i.e., functional requirements) and general security requirements (i.e., non-functional requirements). Hence, this chapter have addresses part of: i) RO2 as the key characteristics of fog computing are identifies, also highlighted the main challenging issues that deter the deployment of fog computing within the IoT network, ii) RO3 as the functional and non-functional requirements of fog computing were investigated in details, also highlighted the barriers that might impede fog in the IoT network. Hence, this chapter fulfill RQ1 for cognitive capabilities and RQ2 for fog computing criteria and requirements.

The concept of cognitive fog (CF) is also presented in this chapter. The cognitive fog discussed the opportunities of fog nodes within the network, in terms of what can be achieved by employing a fog that not only act on the gathered data but also learns from them and makes decisions. CF advocates fogs to interpret the data so it can learn from their process experiences according to different situations/scenarios, then fogs can get better

when performing the repeated processes. The operations of CF runs over four connected worlds; data world featuring both raw and filtered data, processes world featuring processing models, fog world featuring the CF processes and controls, and finally the things world that is controlled by the CF to adapt with environment changes. Moreover, one of the important characteristics of CF is node federations, which is about gathering multiple fog nodes to perform/achieve a specific task in a certain situation. There are two types of federations; planned and ad-hoc federations. Planned federations are formed at the design-time, while ad-hoc federations are formed at run-time. The performance of the developed CF test-bed shows that fog can perform better on repeated processes. After exploring fog computing design requirements and setting the preparations of adopting fog nodes, the next chapter discusses the fog/cloud coherence in IoT and proposes set of criteria that defines where data of things should be sent (cloud, fog, or both) and in what order (cloud then fog or fog then cloud or both concurrently).

CHAPTER 4

Collaboration Model of Fog and Cloud

Essentially, all models are wrong,
but some are useful.

George E. P. Box

4.1 Introduction

Today's Information and communications technology (ICT) requires a different way of approaching the huge volume of data that needs to be transferred securely, processed rapidly, and used properly. Although the trend is to shift data and computation from organizations to the cloud as this operation model offers many benefits, some organizations have been reluctant to adopt it. According to a Logicworks survey, 78% of IT decision makers believe that vendor lock-in prevents their organisation from maximising the benefits of cloud resources. This led the majority of ITs decision makers to choose not to fully invest in cloud because they value long-term vendor flexibility over long-term cloud success" [154]. Cloud is even a major concern with IoT practitioners. Exchanging data back and forth between things and the cloud could turn out to be time consuming, be subject to alterations and misuses, and heavily depend on network availability and reliability [72]. Storage and/or computation could better serve the IoT industry when it happens "next" to where data is collected minimizing its transfer and avoiding its exposure to unnecessary risks. This is the essence of fog computing. However, rather than treating cloud and fog as antagonists, this chapter discusses how they can work hand-in-hand through a fog-cloud seamless collaboration of the operations that each and both can handle.

Fog-cloud collaboration has become doable because of the recent advances in storage, networking, and processing capabilities of fog devices [155]. The objective is to assist engineers who are in-charge of developing IoT applications, to define where data of things should be sent (i.e., cloud, fog, or both) based on the intrinsic characteristics of these applications. These characteristics are presented in this chapter and vary from data latency to sensitivity and freshness. The contributions of this chapter includes; (*i*) a collaboration model of fog and cloud, (*ii*) a set of criteria that defines where data of things should be sent (cloud, fog, or both) and in what order (cloud then fog or fog then cloud or both concurrently), and (*iii*) a demonstration of fog-cloud collaboration through a healthcare-driven IoT case study deployed on a testbed.

4.2 Collaboration Model of Fog and Cloud

The collaboration model of fog and cloud computing consists of two main parts. The first part presents an overview of the proposed collaboration model between fog and cloud computing, the approach for fog-cloud collaboration on data processing. The second part defines the criteria that guide the collaboration for fog-cloud. Thus, this includes the selection of specific recipients (i.e., cloud and fog nodes) to which sensed and actuated data that things generate are sent, also a specific fog-cloud collaboration configuration that defines where things should send their data. The fog-cloud collaboration model adopts distributed based architecture for both fog and cloud nodes. This collaboration of fog-cloud is generally about delivering/achieving the best QoS and QoE to end users, such as reducing service time and avoiding delays for real-time systems.

4.2.1 Foundations of fog-cloud collaboration model

The foundations of the fog-cloud collaboration model is discussed in this section. Figure. 4.1 presents the proposed approach for supporting the fog-cloud collaboration model, the first tier *(thing-fog-cloud interactions)* means that the data of the IoT ecosystem can be sent to fog, cloud, or both depending on the IoT system's requirements/criteria like those discussed in Section 4.3, similarly, the second tier *(cloud-fog interactions)* of Figure 4.1 means that the data can be shared between both fog and cloud based on needs. It is clear that fog nodes have just the same attributes as cloud nodes have; storage capacity, computing power, and networking capabilities that vary according to the nodes specification and the needs and requirements of the IoT-enabled systems. The approach features an IoT ecosystem in which things are expected to feed "appropriate" recipients (i.e., cloud, fog, or both) with data.

The fog-cloud model can provide an elastic computational resources for large scale processing systems, thus fog and cloud can work either independently (i.e., fog↔fog or cloud↔cloud) or collaborated (i.e., fog↔cloud). This chapter focus on the fog-to-cloud collaborations (fog-to-fog next chapter), the fog and cloud will aid each other to serve the end-user; hence they can improve the ability to handle big-data acquisition, aggregation, reducing data transportation as well as balancing the computation power used for data processing. It is worth noting that even in fog-to-fog coordination, when fog nodes work

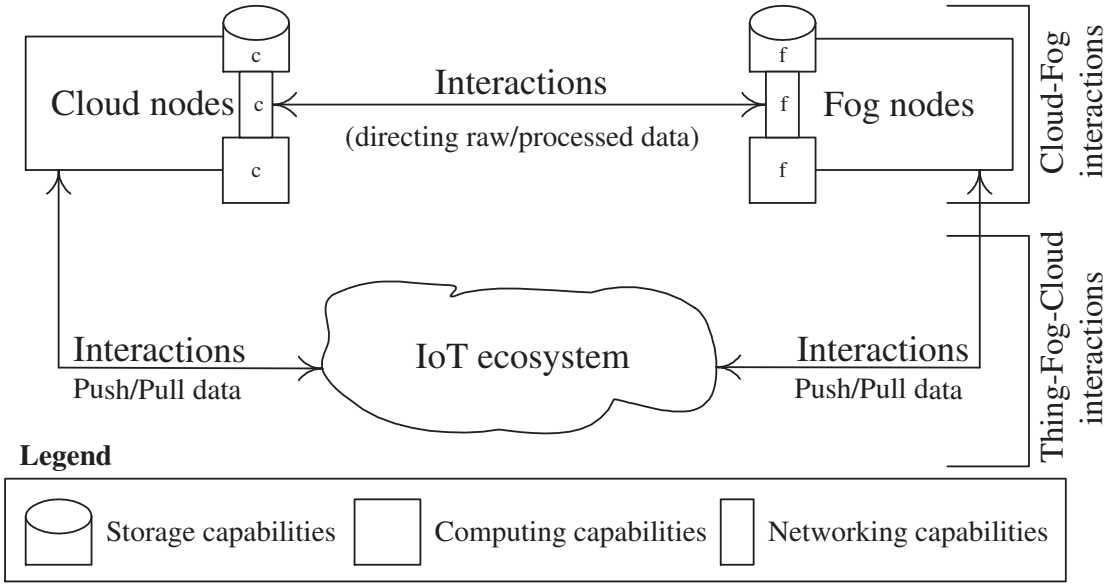


Figure 4.1: Representation of the fog-cloud collaboration model

together to achieve one task, cloud can be used to aid in relevant decision-making that requires cloud computing knowledge and/or capabilities. Delay sensitive applications may gain priority over others to make use of the fog-cloud model over applications/systems that are not time sensitive. The decision-making in fog-cloud interactions and where data can be processed may also be influenced by the demand and location of these fog-cloud resources, for example, fog-cloud resources in busy area versus quiet area. According to Figure. 4.1, two main categories of interactions are shown and discussed below:

- The first category, known as TFC for Thing-Fog-Cloud, involves the ecosystem and cloud/fog nodes that consists of pushing and/or pulling (on either a continuous basis or a regular basis) raw data from things to cloud/fog nodes.

- The second category, known as CF for Cloud-Fog, involves cloud and fog nodes and consists of directing either raw or processed data from cloud to fog or vice-versa. Rather than sending similar data to both cloud and fog, things could send data to either fog or cloud that will relay these data to the other partner. More details about this option are provided in Section 4.3.

The IoT ecosystem also features networks of things that support the exchange of additional data (not-initially requested but could be deemed relevant) from things to cloud/fog nodes.

4.3 Criteria for Selecting Data Recipients

This section defines the criteria adopted for selecting thing's data recipients, whether its fog, cloud or both. The objective is to assist engineers, who are in-charge of developing IoT systems, in selecting/knowing where data generated by things should be sent (i.e., cloud, fog, or both cloud and fog), also whether the collaboration is required or one recipient can handle the desired service/tasks. This can also be influenced by the characteristics and requirements of the IoT applications/systems in term of performance, reliability and privacy. The assumption made in support of these criteria is that cloud nodes are physically far from things and that fog nodes are physically closer to the things. To achieve this objective, an exhaustive set of criteria is proposed, which allow us to look at data transfer from different perspectives (e.g., location, time, and application's needs). These criteria are as follow:

1. **Proximity** refers to how "close" things are from cloud/fog nodes, service type, storage and/or computational facilities. Continuous transfer of large volumes of data could be time consuming and require a certain level of bandwidth. Thus, if things are in fixed location, interactions with fog/cloud nodes in terms of duration and bandwidth could be estimated/predicted. However, this does not apply to mobile things since network coverage varies and is dependent on many physical factors. The proximity criterion is appropriate when considering data transmission, such as $\mathsf{Thing} \rightarrow \mathsf{Cloud}$ and $\mathsf{Thing} \rightarrow \mathsf{Fog}$.

2. **Frequency** criterion refers to the rate of data transfer from things to fog/cloud nodes. The frequency could be regular with fixed frequency (e.g., every 2 hours) or continuous with different frequency (e.g., every time patient's blood pressure drops). Frequency may be set according to the proximity criterion.

3. **Sensitivity** criterion refers to the nature of data exchanged between things and fog/cloud nodes. Highly-sensitive data should not be exposed longer than is required on the networks and during the data processing among all things, fog and cloud nodes.

4. **Time and Freshness (velocity)** criterion refers to how important is the data delivery between things and fog/cloud, hence this is according to the requirements of the

IoT system on how recent is the data, whether it should be in real-time, near real-time or at anytime. The delay that results from withholding/processing data at the thing, fog and cloud levels until its transfer to the end-user. This may also include the delay for holding data at the thing level in preparation for transferring the collected data (i.e., real-time vs batch processing) to either fog, cloud, or fog/cloud nodes.

5. **Volume** criterion refers to the amount of data that things produce and send to fog/cloud nodes. For instance, if the fog can handle up-to a set image size in pixels, so when an image with a bigger size is captured, this might be directed to the cloud to take care of it; otherwise, the image can be partitioned into sub-graphs, in which case the fog can work in a collaborative mode to perform the processing.

6. **Criticality** criterion refers to the demands that fog/cloud express with regard to data of things. This may combine both data integrity and availability. Low demands could lead to ignoring certain data, while high demands could lead to high traffic and network congestion. Therefore, selecting appropriate data recipient is crucial.

7. **Variety** criterion refers to the different types of data being generated, for example, personal data can be anything like texts, emails, photos, videos, sensor-data etc., hence, such types of data can be split in three categories: structured, semi-structured and unstructured data. Structured data has a fixed format and size, semi-structured data has a structure but does not obey the formal structure of data models (e.g., relational databases), while the unstructured data does not have any format and poses challenges for processing and analysing activities. Therefore, according to data needs and requirements, the recipient can be identified; either fog, cloud, or fog-cloud.

8. **Veracity** criterion refers to the biases, noise and abnormality in data. This refers to the data that is being collected and stored as well as mined meaningful to the problem being analysed, which can be critical for the cognition capabilities. Veracity can be more challenging in comparison with above criteria, hence careful selection of data recipient is essential to prevent sloppy data from accumulating in the network, fog as first hop can help in such case. Also an approach/procedure may required to allow only the clean data in the processes of feeding and learning of the cognition capabilities, hence fog-cloud may collaboration for such scenario.

Table 4.1: Data-recipient selection criteria *versus* interaction forms (HR:Highly Recommended, R:Recommended, NR:Not Recommended, N/A:Not Applicable)

Criterion	Features	T→C	T→F	T→C|F	T→C→F	T→F→C
Frequency	Continuous stream	NR	HR	N/A	NR	R
	Regular stream					
	Short gaps	NR	HR	N/A	NR	HR
	Long gaps	R	R	R	R	R
Sensitivity	High	NR	HR	N/A	NR	HR
	Low	R	R	R	R	R
Freshness	Highly important	NR	HR	N/A	NR	R
	Lowly important	R	R	R	R	R
Time	Real-time	NR	HR	N/A	NR	HR
	Near real-time	R	HR	HR	R	HR
	Batch-processing	HR	NR	N/A	R	NR
Volume	High amount	HR	NR	N/A	NR	R
	Low amount	NR	HR	N/A	NR	R
Criticality	Highly important	HR	HR	HR	HR	R
	Lowly important	NR	HR	N/A	NR	HR

In Table 4.1, we analyze the role of the aforementioned criteria in recommending a certain form of interaction between thing, clouds, and fog nodes. The specialization of TFC and FC interactions, mentioned in Section 4.2.1, leads to five interaction forms classified into one-hop and two-hops interactions. The interaction forms described below, the notations T, C, and F refer to Thing, Cloud, Fog, respectively, → refers to flow, and | refers to concurrently.

1. One-hop with {T→C, T→F, T→C|F}, assuming that processing data at cloud nodes is different from processing data at fog nodes in term of speed, privacy etc. Example of T→C includes batch processing of CCTV data, example of T→F includes processing CCTV frames as being captured in real-time, while the example of T→C|F includes processing CCTV data with no time-sensitivity (i.e., delay is acceptable).

2. Two-hops with {T→F→C} this includes pre-processing data at fog nodes prior to sending the new data to cloud nodes, for example, process patient's vital sensor data in real-time at the fog, then notify the cloud with a new record for the patient to be used in the future (i.e., keeping patient history). While, {T→C→F} includes the pre-processing data at clouds prior to sending the new data to fogs, this could be a rare, but can be used for scenarios where some authentication processes is required before assigning a fog service, for example authenticating a doctor trying to access patient's healthcare service that run on a fog.

Table 4.2: Cloud and Fog Computing Characteristics (Cisco)

#	Characteristics	Cloud	Fog
1	Latency	High	Low
2	Delay jitter	High	Low
3	Location of service	Within the Internet	At network edge
4	Distance client to server	Multiple hops	One hop
5	Security	Undefined	Can be defined
6	Attack on the enroute	High probability	Very low probability
7	Location awareness	No	Yes
8	Geo-distribution	Centralized	Distributed
9	No. of server nodes	Few	Very large
10	Support for mobility	Limited	Supported
11	Real-time interaction	Supported	Supported
12	Last-mile connectivity type	Leased line	Wireless

Cisco[1] provides Table 4.2 to illustrate how cloud and fog would handle the characteristics of certain applications. For instance, real-time applications that ask for almost-immediate action and high data-protection, would discard cloud as an operation model. Contrarily, fog would offer better support to mobile applications compared to cloud.

[1]Cisco blog on IoT, from Cloud to Fog Computing
https://blogs.cisco.com/perspectives/iot-from-cloud-to-fog-computing

Establishing correspondences between Table 4.2's characteristics and Table 4.1's suggestions of how to proceed with data (i.e., 5 interaction forms), yields into the following points:

- Frequency criterion is dependent on the data stream between things and cloud/fog nodes. If the stream is continuous (non-stop), then it is highly recommended to involve fog nodes in all interactions so, that, direct data-transfer to cloud nodes is avoided as per Table 4.1, rows 1&2 (i.e., low-latency and low-delay jitter). If the data stream is regular, recommendations will depend on how short *versus* long the gaps are during data transfer. For example for regular stream with long gaps any interaction can be recommended "R" since there is enough time for data to go into any interactions. Generally these interactions can also be influenced by the type of data and requirement of the applications/systems.

- Sensitivity criterion is about the protection measures that need to be put in place during data exchange between things and cloud/fog nodes. If the data is highly sensitive, then it is highly recommended to involve fog nodes in all interactions so that protection is ensured as per Table 4.1, rows 4&5, otherwise, data could be sent to cloud and fog nodes. Security can be defined along with very-low-probability of attack enroute by malicious nodes in the network.

- Freshness criterion is about the data quality to maintain during the exchange between things and cloud/fog nodes. If the data needs to be highly fresh, then it is highly recommended to involve fog nodes in all interactions as per Table 4.1, rows 6&7. This can be subject to being aware of the location of fog nodes and their support to real-time interactions should be provided. It is worth noting that data freshness is different from data frequency, having high frequency does not reflect the freshness of data. Freshness reflects the new and useful data for the system/application, while high frequency is probably just redundant data.

- Time criterion is about how soon data is made available for processing. If it is real-time processing, then it is highly recommended to send data to fog nodes as per Table 4.1, row 8-10. If it is near real-time (i.e., minutes are acceptable) then it can be sent to cloud and/or fog nodes. Otherwise, cloud is ideal for data batch-processing. In batch processing, cloud nodes are always preferred over fog nodes due to the limited capabilities of fog nodes. More details on fog congestion can be found in Chapter 5.

- Volume criterion is about the space constraint over the amount of data collected or produced by things. In other words, it is the constraint of the amount of data collected/produced by things and the correspondent/equivalent space required on fog/cloud nodes to handle these data. If this amount is big, it is highly recommended to send data directly to cloud nodes. Otherwise, data could be sent to a fog node(s) and then to cloud. In case of a large amount of data and where the data is divisible, then data could be sent over to multiple fog nodes as per Table 4.1, row 11&12 (i.e., distributed geo-distribution). For instance, the system can handle up to a set image size in pixels, so when an image with a bigger size is captured, the system might decide to send it to the cloud to take care of it; otherwise, the image can be partitioned by the system into sub-graphs, in which case the system sends them separately into many local collaborative and connected fogs for processing.

- Criticality criterion is about ensuring data availability according to fog/cloud demands. If fog/cloud demands are highly important, then it is highly recommended that data should be sent to fog/cloud regardless of the hop number as per Table 4.1, row 13&14(i.e., geo-distribution) to ensure data availability. Otherwise fog nodes could sort tasks based on their priorities, keeping higher priority actions within a node, sending data that can wait a few minutes for a larger aggregation to cloud node.

4.4 System Evaluation

To validate the fog-cloud collaboration model, a developed test-bed was deployed upon which a set of experiments were carried out. The experiments refer to a healthcare-driven IoT case study in which medical data are collected and then transmitted to different recipients.

4.4.1 Case Study - Healthcare

The recent advances in ICT have facilitated the emergence of a new generation of sensors and IoT-based applications that can be used in different contexts like smart city and smart healthcare in such a way that it becomes ordinary need. Cisco and Business Insider predict that the IoT will make use of 50 billion individual devices that can produce 507.5 zettabytes of data by the end of the current decade [156]. The large distance between the cloud and IoT users, and the number of fog nodes in the network have lowered the overall performance even more, and notoriously cannot guarantee the response time for applications demanding real-time assurance processing and very low latency (e.g., healthcare). An example to consider is the around-the-clock urgent/emergency care services department (or Intensive Care Unit) in a hospital, which deals with genuine life threatening cases (e.g., breathing difficulties, severe allergic reactions, and consequent high blood pressure), where patients may have only moments before a dip in vital signs which might end in a catastrophic crash. In such cases, readings from patient's wearable sensors need to make it to the doctors within a split-second time frame, otherwise life could easily be lost. Such a highly critical department requires use of devices and technologies with real-time analytic and low latency constraint along with mobility features.

4.4.2 Test-bed and experiment configurations

The test-bed was developed based on the case study described in 4.4.1. The configurations were set so that a full test for the developed test-bed with proposed interactions was evaluated. Figure. 4.2 depicts the test-bed's architecture consisting of three layers: thing, fog, and cloud. Each layer includes hardware and/or software components specific to the healthcare case-study. Communications between the thing layer and other layers is taken care by a gateway. The three layers are connected to each other through 4-two-way network topologies that implement the 4 interaction forms discussed in Section 4.3: the T → C, T → F, T → F → C, and T → C → F. Mosquitto[2] was used for exchanging messages, via MQTT protocol, among the 3 layers (i.e., thing, fog and cloud layers). The hardware components and their specs of each layer are as descried below.

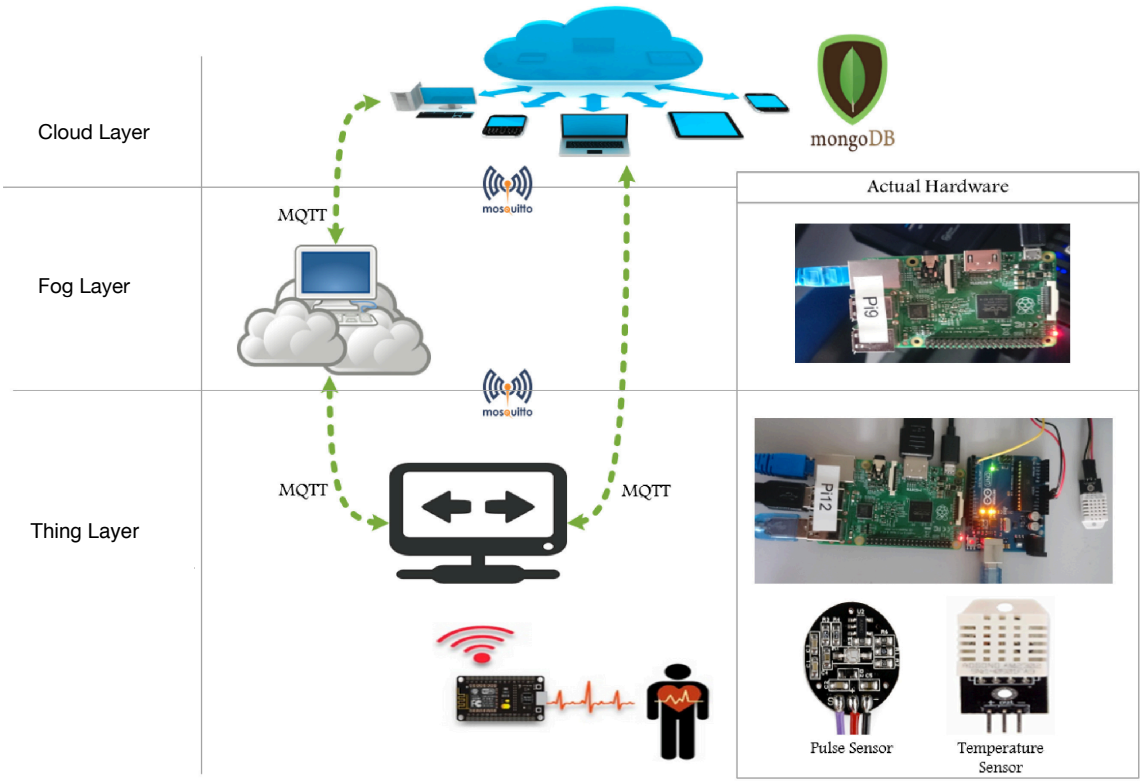


Figure 4.2: Testbed's architecture for the healthcare-driven IoT case study

[2]mosquitto.org, Open Source MQTT Broker and part of Eclipse IoT project v3.1.1.

```
{"Patients":
  {"Patient":
    [
      "Id": "100",
      "Type": "symptoms",
      "Heartrate": "88",
      "Diastolic": "90",
      "Systolic": "128",
      "Temperature": "79",
      "Oxygen": "90",
      "Sugar": "112",
      "Time": "2018-01-20 15:48:20",
    ]
  },
  {"Patient":
    [
      "Id": "101",
      "Type": "symptoms",
      "Heartrate": "80",
      "Diastolic": "95",
      "Systolic": "130",
      "Temperature": "81",
      "Oxygen": "88",
      "Sugar": "116",
      "Time": "2018-01-21 12:48:20",
    ]
  }
}
```


Figure 4.3: Example of messages in JSON format

- The thing layer includes 3 components: (*i*) a gateway featuring a Raspberry Pi (rPi2) model B (1 GB RAM and Broadcom BCM2836 ARMv7 Quad Core 32 bit processor running at 900 MHz), (*ii*) a digital temperature and humidity sensor (AM2302), and (*iii*) a microcontroller Arduino UNO board (Clock Speed 16 MHz and 2 KB SRAM) connected to both the gateway and the sensor. Arduino UNO pushes data to the rPi2 through a serial connection while the gateway is connected to the Internet (with uploading speed at 32.6 Mbps and downloading speed at 98.5 Mbps) through an Ethernet cable CAT5 with 100 Mbps to populate/deploy the data to either cloud, fog, or both.

- The fog layer includes 1 component: a Raspberry Pi (rPi2) with a similar specification to the one in the thing layer. It connects to the Internet through an Ethernet cable, processes data received from the gateway and cloud and then timestamps the received JSON data.

- The cloud layer is a 4 core Virtual Private Server (VPS) located in a data centre in Germany, operates under Linux CentOS7, and is technically specified as follows: 300 GB 100% SSD storage space, 12 GB RAM, and 100 Mbit/s data transmission port for unlimited traffic. Note that the VPS is totally dedicated for this experiment and thus, is not involved in any other processing that may share the server resources and cause delay. Cloud processes data received from the gateway and fog and then timestamps the received JSON data.

Regarding the experiment configuration, we use an in house Python program to let the sensor stream data continuously (about 5-10 readings per second) for 24 hours over each of the 4 network topologies. Upon reception at the end point, JSON messages, are timestamped by an in house Python program prior to storing them into a Mongo database. Fig. 4.3 shows a message formatted in JSON during the experiments. Recall that messages are transferred using MQTT broker. To support message transfers, different MQTT brokers are used to ensure the lowest latency-time. In $\mathsf{T} \rightarrow \mathsf{F}$ and $\mathsf{T} \rightarrow \mathsf{F} \rightarrow \mathsf{C}$ configurations, the fog acts as a broker. In $\mathsf{T} \rightarrow \mathsf{C}$ and $\mathsf{T} \rightarrow \mathsf{C} \rightarrow \mathsf{F}$ configurations, the cloud acts as a broker. In $\mathsf{T} \rightarrow \mathsf{F} \rightarrow \mathsf{C}$, the fog acts as a broker. Finally, in $\mathsf{T} \rightarrow \mathsf{C} \rightarrow \mathsf{F}$, the cloud acts as a broker.

4.4.3 Performance Evaluation

The performance evaluation and results presented in this section are based on the frequency and time criteria and recommendations from Section 4.3. The frequency and time criteria has been selected in this evaluation for two reasons; i) frequency and time criteria are feasible combination to reflect the performance of IoT one-hop {T→C, T→F} interactions and the two-hops {T→F→C, T→C→F} interactions, ii) worth investigating the time (reflect the latency) and the frequency (reflect the traffic) criteria as they can impact IoT performance, thus impacting both QoS and QoE. Also. the selection of both criteria fit with the scope of the performance evaluations of *f*og-2-*f*og coordination model presented in next chapter.

The evaluation taken from running 4 experiments, one for each two-way network topology, the physical topology configurations are based on our one-hop and two-hops interactions, hence the configs are; Config_1: T → C → F, Config_2:T → F → C, Config_3:T → C, and finally Config_4:T → F. These configurations are to compare recommendations indicated in the proposed coordination model (Section 4.3) with the total (end-to-end) latency obtained per topology. Specifically, we experiment on the frequency criterion with "continuous stream" and time criterion with real-time processing, these criteria described in Section 4.4.2. All the experiments (Figures 4.4 to 4.7) were conducted for the same duration (i.e., 24 hours for each configuration) to ensure consistency, in fact the number of transferred packets are also fixed to 25k per each configuration, this mainly to avoid evaluating uneven number of packets in each topology due packet's losses to either connection issue or sensors glitch.

Each transferred packet in each configuration contains similar structure with a raw data like id, value, and timestamps of sending/receiving/processing the packet. For each experiment, the total end-to-end latency was calculate the for transferring data packets produced by things to either fog or cloud nodes for processing. To generate a "continuous data stream", 25k of packets were sent by the thing node, for each topology. For each sent packet the recipient node fetch the data and log the timestamp of which it has been received and then transferred to either the next recipient (in case of two-hops interactions) or sent pack to the thing node (in case of one-hop interactions). In term of evaluation, the packets are aggregated at the thing's node and compute the round-trip to extract the end-to-end latency per packet.

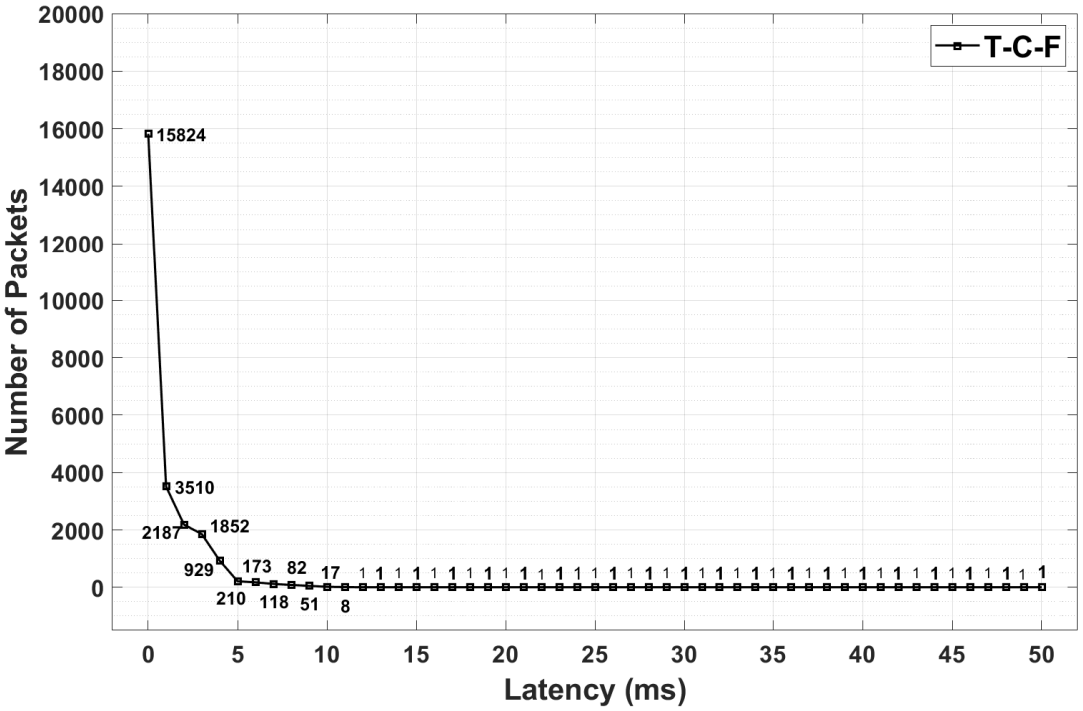


Figure 4.4: Number of packets per latency in $\mathsf{T} \rightarrow \mathsf{C} \rightarrow \mathsf{F}$ configuration$_1$

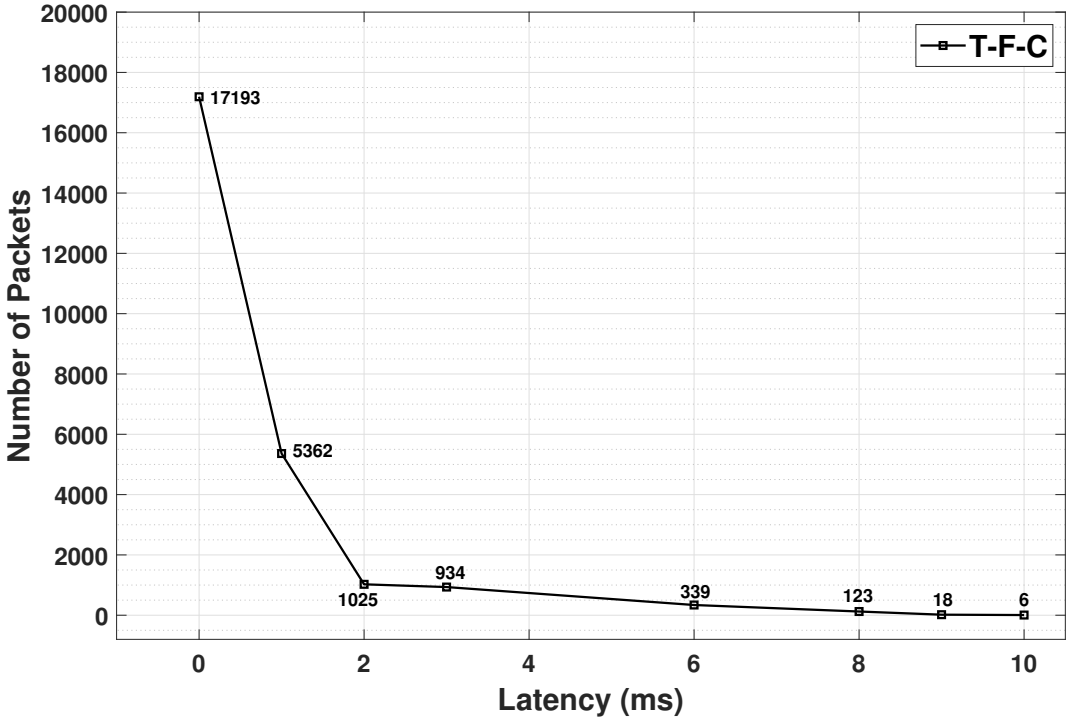


Figure 4.5: Number of packets per latency in $\mathsf{T} \rightarrow \mathsf{F} \rightarrow \mathsf{C}$ configuration$_2$

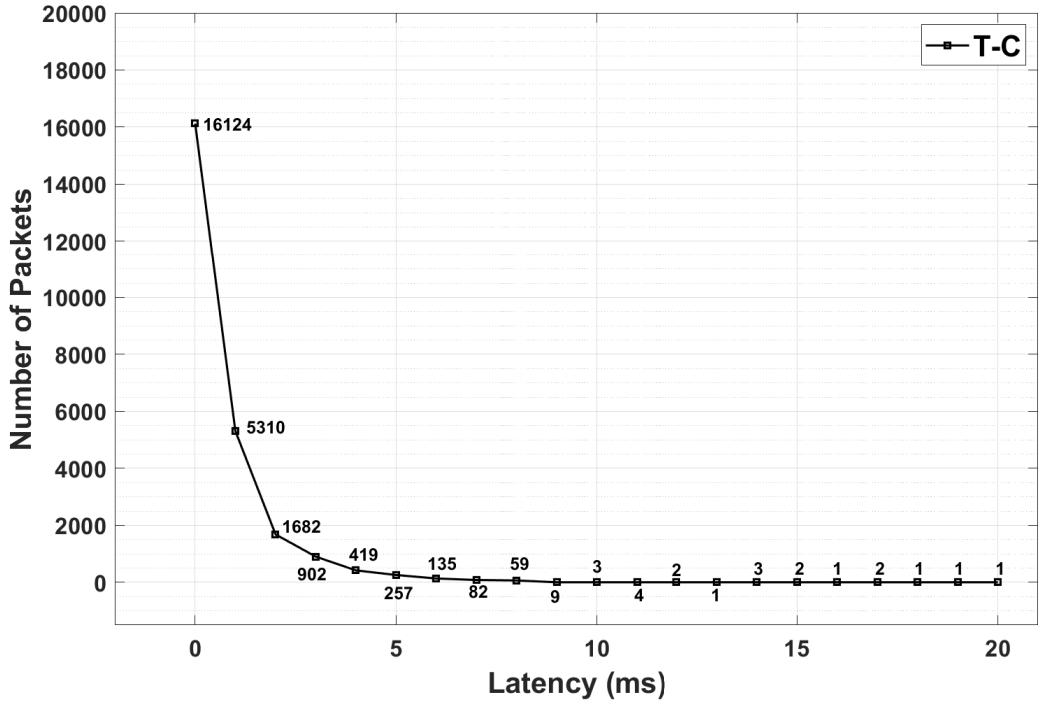


Figure 4.6: Number of packets per latency in $\mathsf{T} \rightarrow \mathsf{C}$ configuration$_3$

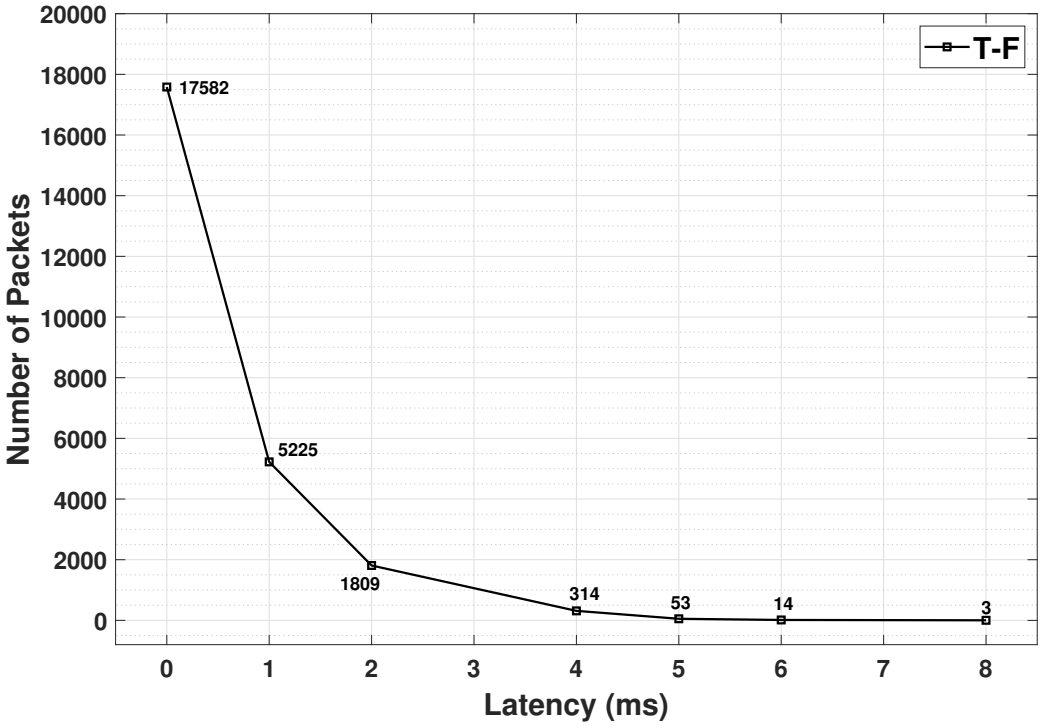


Figure 4.7: Number of packets per latency in $\mathsf{T} \rightarrow \mathsf{F}$ configuration$_4$

After calculating the latency for each packet, they have been grouped based on the end-to-end latency as presented in Figures 4.4 to 4.7. Reminder, the evaluation results are based on 25k of packets that have been fixed for all configurations to ensure consistency. Figure 4.4 shows packets latency of Config$_1$:T→C→F, Figure 4.5 shows packets latency of Config$_2$:T→F→C, and similarly Figure 4.6 and 4.7 shows packets latency of their correspondence topology. To explain more, figures are simply grouping the packets based on latency in each configuration, fog example, in Figure 4.4 there were 210 packets needing round-trip delay of 5 millisecond in Config$_1$:T→C→F topology. The delay in receiving some packets can be down to either packets transfer delay due to channel congestion that occurs with high traffic (i.e., high frequency), or the impact propagation delay to far cloud node (the hired cloud were based in Germany). Moreover, Figures 4.5, 4.6 and 4.7 also groups the 25k packets based on the end-to-end latency in each topology configuration. It is clear that adopting fog as first hop, first recipient to thing's data, will help in providing the lowest delay. In fact, this result proved in Figure 4.8, where the delay mean and the standard deviation (STD) were computed for each of the four configurations, clearly Config$_2$:T→F→C and Config$_4$:T→F have the lowest mean and lowest STD, thus lowest latency.

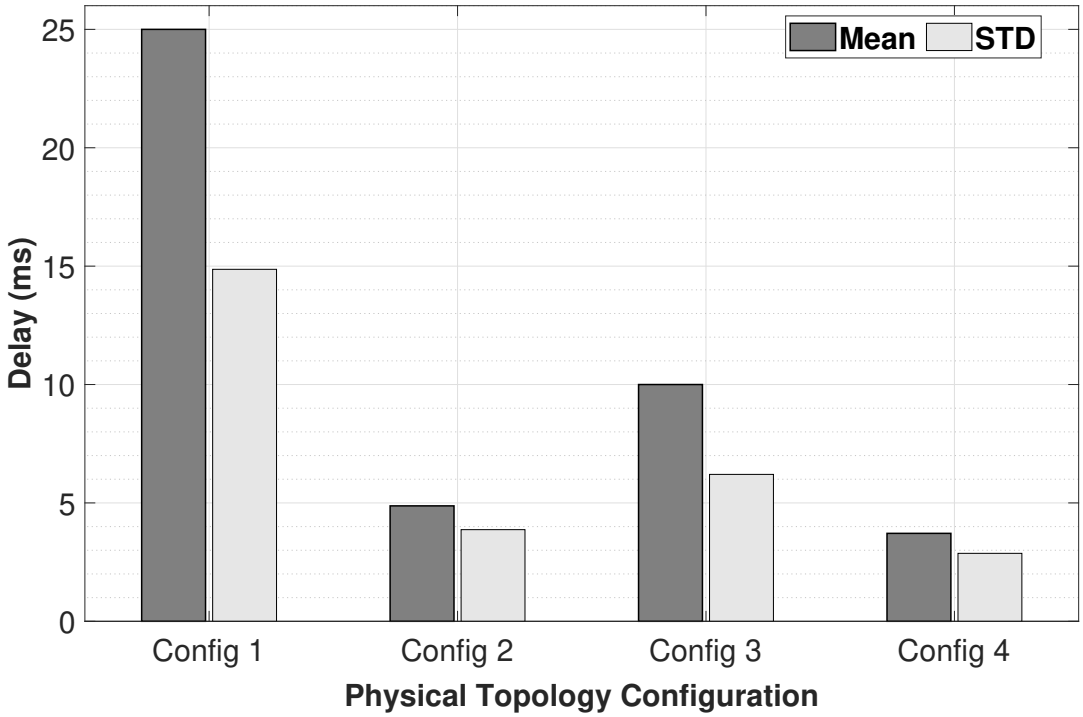


Figure 4.8: Delays means and STDs (for 25k of packets) for each configuration

Figure 4.9 demonstrates the total end-to-end latency in each coordination configuration for streaming data continuously up to 25k of packets. It is clear that Config$_4$:T $\rightarrow$ F topology consumes less time (i.e., lowest delay) than any other of three configurations to send the same amount of sensor-emitted data (i.e., 25k packets). These results reflect the recommendation of HR for Config$_4$:T $\rightarrow$ F in the case of frequency criterion with continuous stream, and NR in Config$_1$: T $\rightarrow$ C $\rightarrow$ F and Config$_3$: T $\rightarrow$ C configurations as they take more than 22k milliseconds and 16k milliseconds, respectively, for total round-trip of the 25k of packets. In term of delays average and STD, Config$_4$:T $\rightarrow$ F topology still outperform other topology configurations as per Figure 4.8.

There is a clear run-time improvement in Config$_2$: T $\rightarrow$ F $\rightarrow$ C, Config$_3$: T $\rightarrow$ C, and Config$_4$: T $\rightarrow$ F topologies in Figure 4.5 to 4.7 respectively, compared to the worst case of run-time of Config$_1$: T $\rightarrow$ C $\rightarrow$ F in Figure 4.4, this results are depicted in Figure 4.10. For further clarification on Figure 4.10, Config$_4$ T $\rightarrow$ F topology in Figure 4.7 spends around 53% less time to serve the 25k packets compared to Config$_1$: T $\rightarrow$ C $\rightarrow$ F in Figure 4.4; whereas Config$_2$: T $\rightarrow$ F $\rightarrow$ C in Figure 4.5 consumes 40% less time compared to the same benchmark, and Config$_3$: T $\rightarrow$ C in Figure 4.6 is only 26%. It worth also comparing Config$_2$: T $\rightarrow$ F $\rightarrow$ C, and Config$_4$: T $\rightarrow$ F topologies with Config$_3$: T $\rightarrow$ C since it is the most common topology for today IoT systems/applications. Moreover, the results shows that Config$_2$: T $\rightarrow$ F $\rightarrow$ C, and Config$_4$: T $\rightarrow$ F topologies are still outperform Config$_3$: T $\rightarrow$ C in term of run-time for the 25k packets, as they have the lowest round-trip time, more precisely the run-time improvement of Config$_2$: T $\rightarrow$ F $\rightarrow$ C, and Config$_4$: T $\rightarrow$ F are 36% and 18%, respectively, compared with the run-time for Config$_3$: T $\rightarrow$ C.

The results presented in Figures 4.4 to 4.10 proven that the proposed recommendations in section 4.3 are valid. To explain more, the results in Figures 4.8, 4.9 and 4.10 for Config$_4$: T $\rightarrow$ F and Config$_2$: T $\rightarrow$ F $\rightarrow$ C are in line with our recommendations for both the time criterion (reflect the delay) and the frequency criterion (reflect the traffic of 25k packets) of Config$_2$: T $\rightarrow$ F $\rightarrow$ C and Config$_4$: T $\rightarrow$ F topologies being recommended, while Config$_1$: T $\rightarrow$ C $\rightarrow$ F Config$_3$: T $\rightarrow$ C topologies being not-recommended.

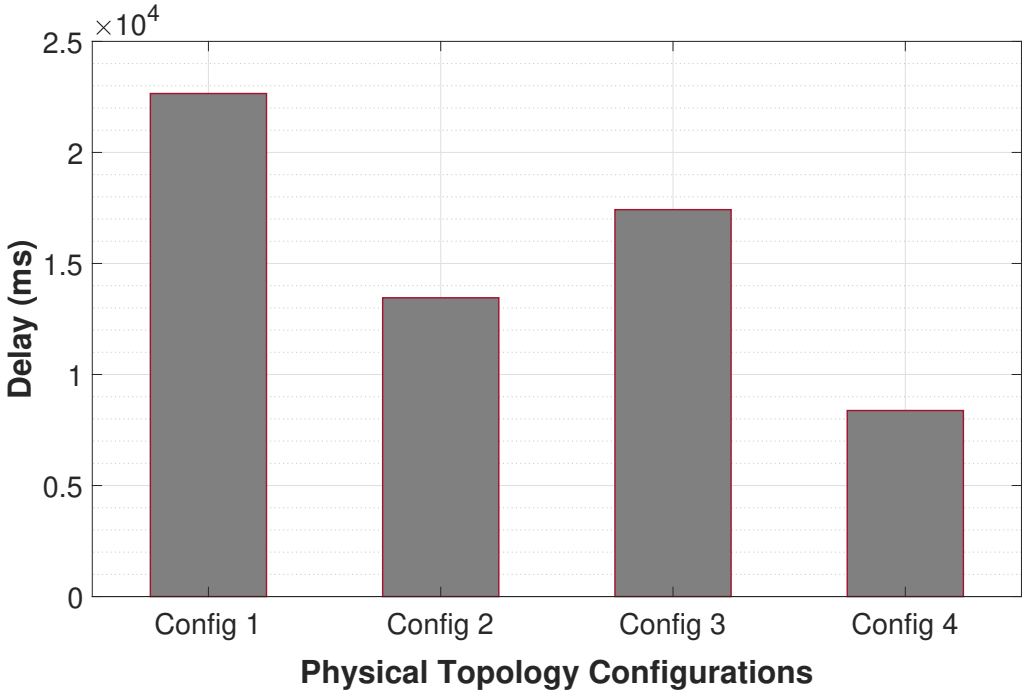


Figure 4.9: Total latency (for 25k of packets) for each configuration

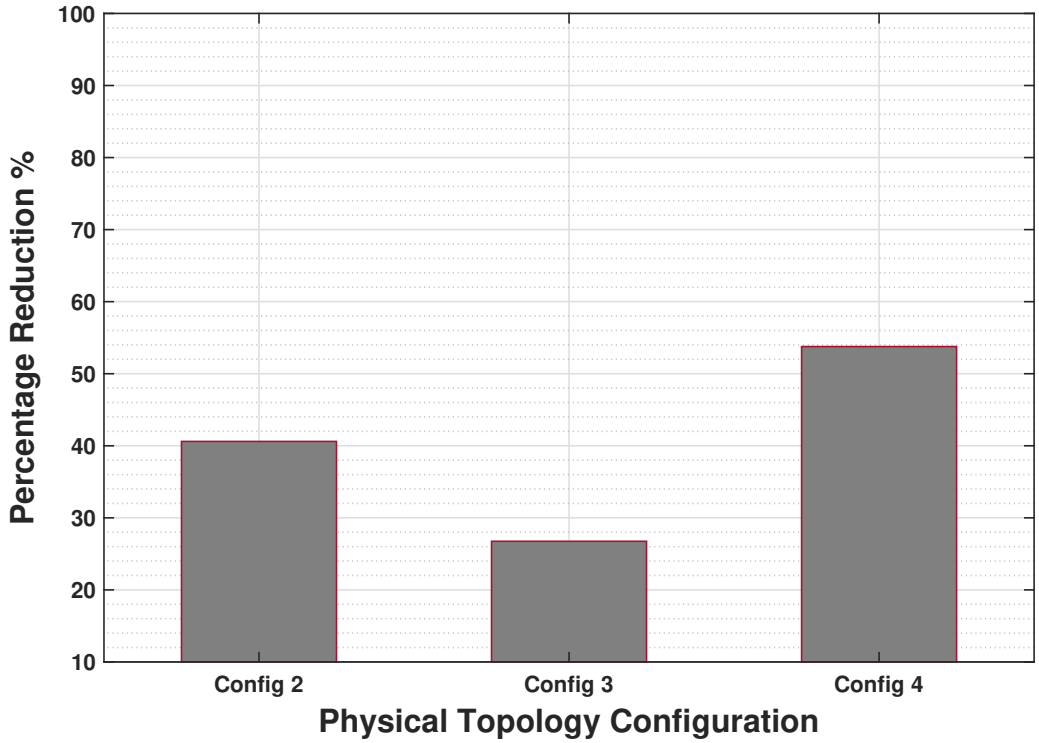


Figure 4.10: Percentage performance improvement of $\mathsf{T} \rightarrow \mathsf{F} \rightarrow \mathsf{C}$, $\mathsf{T} \rightarrow \mathsf{C}$ and $\mathsf{T} \rightarrow \mathsf{F}$

4.5 Chapter Summary

Fog-cloud collaboration has become doable due of the recent advances in storage, networking, and processing capabilities of fog nodes. This chapter presented a fog-cloud collaboration model that assists organizations wishing to ride the IoT wave, in determining where data should be sent (cloud, fog, or cloud & fog concurrently) and in what order (cloud, fog, or cloud & fog concurrently). To this end, a set of data-recipient selection criteria - frequency, sensitivity, freshness, time, volume, and criticality - have been proposed ensuring a smooth collaboration. Hence this chapter have addressed RO2 in term of to proposes data recipient criteria in fog/cloud environment, thus fulfill RQ2.

This fog-cloud collaboration was illustrated with different levels of recommendations about the appropriate data recipients. For instance an IoT application that is keen to handle continuous data-streaming would not consider sending data from things to clouds but from things to fogs. Contrarily, an IoT application that is keen to handle high amounts of data-exchange would consider sending data from things to clouds but not from things to fogs. Different concerns and different priorities mean different data recipients. For validation purposes, a healthcare-driven IoT application along with a test-bed, that features real sensors (temperature and humidity AM2302) and fog node (RPi2 model B) and cloud data-centre (4 core virtual private server) platforms, was permitted to perform different experiments that demonstrated the technical feasibility of the coordination model as well as the appropriateness of recommending one coordination model over another. The experiments targeted frequency and time criterion along with the continuous stream feature. The evaluation results proven that the proposed recommendations and set of criteria that defines where data of things should be sent (cloud, fog, or both) are valid as the results for Config_4: T $\rightarrow$ F and Config_2: T $\rightarrow$ F $\rightarrow$ C are in line with our recommendations for both the time criterion (reflect the delay) and the frequency criterion (reflect the traffic) of Config_2: T $\rightarrow$ F $\rightarrow$ C and Config_4: T $\rightarrow$ F topologies being recommended, while Config_1: T $\rightarrow$ C $\rightarrow$ F and Config_3: T $\rightarrow$ C topologies being not-recommended. Since this chapter have discussed the fog-cloud collaboration model, the next chapter discusses the {*og*-2-{*og* coordination model and Fog Resource manAgeMEnt Scheme (FRAMES) for optimal resource managements and workload distributions for fog computing.

CHAPTER 5

Coordination Model of *F*og-*to*-*F*og

Stability leads to instability. The more stable things become and the longer things are stable, the more unstable they will be when the crisis hits.

Hyman Minsky

5.1 Introduction

The main advantage of fog computing is the proximity to end-users devices, thus fog's hardware and software resources are placed "closer" to things allowing services that rely on IoT-things' inputs to be carried with minimal delay [14, 37], hence benefiting real-time applications. However, fog nodes can quickly become congested when the number of arrived service requests exceed the fog's capability [14, 3]. Consequently, service latency occurs [14]. In addition, the potential of fog congestion occurrence is high due to the limitations of fog capabilities in comparison to cloud [1, 26]. Therefore, fog resource management is the most important issue/aspect of congested fog nodes [1, 15, 108], as poor resource management can lead to fog congestion which causes latency and inefficiency for services within the fog layer [6, 26]. OpenFog [38] reports that, although fog computing provides extensive peer-to-peer interconnection for communication purposes with the clouds, its nodes run in silos, where no collaboration capability, for job processing, is available. Therefore, fog resource management is needed to unlock the silos and free them from the historical stovepipes working pattern. In fact, poor resource management can cause latency and inefficiency for services within the fog [6, 26]. Therefore, in this chapter will propose a fog nodes that are able to outsource their hardware resources and participate in coordination with other fogs to achieve a single task.

The contributions of this chapter are threefold; i) the {*og*-2-{*og* coordination model that achieves an optimal workload among the collaborated fog nodes. This coordination model allow fogs to outsource their resource. ii) a Fog Resource manAgeMEnt Scheme (FRAMES) that promotes load balancing to address the latency concern of service request's received from things. We adopt the notion of fog-as-a-service [157] where each fog node hosts local computation, networking and storage capabilities. and iii) a formal mathematical model that backs the decision of load balancing among fog nodes via offloading. The offloading model considers not only the queue length of the service packets, but also variant node capabilities as well as different data packets or request types, such as, *heavy-weight* data packets from a CCTV and *low-weight* data packets from sensors. The proposed models and their algorithms outperformed the output of two benchmark algorithms; Random Walk Algorithm (RWA) and Neighbouring Fogs Algorithm (NFA).

5.2 Fog Resource manAgeMEnt Scheme

Although fog nodes are placed "closer" to IoT things so, that, latency is "taken care" of, these nodes can quickly become congested when the number of requests soliciting their services exceed their capabilities [14] [3]. The fog layer in the IoT architecture consists of heterogeneous devices clustered together and forming what is called "fog domains". Each fog device/node has it is own coverage range where the desired fog services are provided. In fact, due to node heterogeneity, service types and sizes (e.g., processing speed and storage capacity) vary from one fog node to another, thus its unclear how fog services are managed and provided. Many questions arise: *"How can fog's services be provided?"* and *"Who does manage and monitor fog resources consumption and provisioning?"* in order to evaluate the QoS and performance of the fog devices. To fill this gap, Fog Resource manAgeMEnt Scheme (FRAMES) is proposed. This section discusses FRAMES, which involves managing fog resources status and provides network analysis and statistics for fog resource provisioning and consumption. Figure 5.1 shows the conceptual diagram of FRAMES. The main functionality is to periodically monitoring fogs' statuses and network loads.

5.2.1 Fog management scheme

Fog nodes/devices can be any device with storage, computing, and network capabilities. Fog can be directly installed by an individual user or network administrator who wants to benefit from desired fog services. Therefore, FRAMES is based on the fog node mesh distribution architecture [20, 4], which is similar to the distribution of WiFi access point topology [4] (i.e., installing routers in a distributed manner with respect to coverage range). Thus, network administrators install multiple interconnected networks of fog nodes in public places (e.g., cities) and private places (e.g., homes) to distribute fog services. This way of fog services distribution is achieved through collaboration between cloud providers, IoT operators, and network infrastructure providers. FRAMES can manage the distribution of fog nodes as well as the monitoring of performance and resources managements in the fog layer. FRAMES includes three main parties which take over the process of managing fog services and coherence as per Figure 5.2.

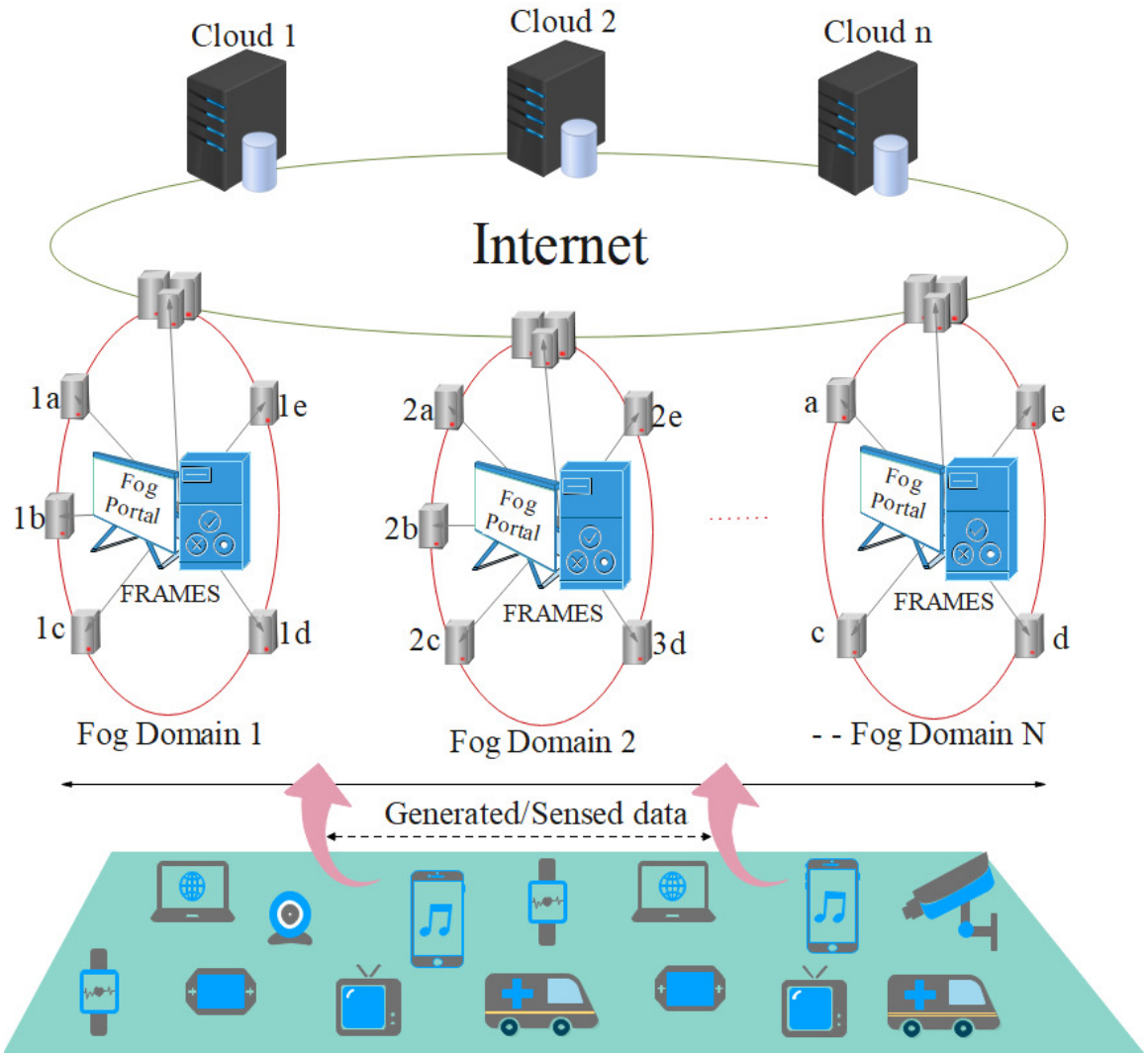


Figure 5.1: Overview of the Fog Resource manAgeMEnt Scheme

- *Fog Portal:* is a distributed software, which is located within each fog domain and forms an intermediate connector between the fog nodes and services' users. This portal features a knowledge-base on a connected fog domain and cloud-based data repository to provide data about all the available fog domains and the services provided by each fog node, thus share data/statistics between fog nodes. The procedure of declaring new/existing fog services via the fog portal starts when the fog owner connects the actual fog node to the IoT network. Thus, as soon as the node is up and running, it will be detected by the local network and assign a unique static IP address to the device, and at this point the node will ping the portal to register de-

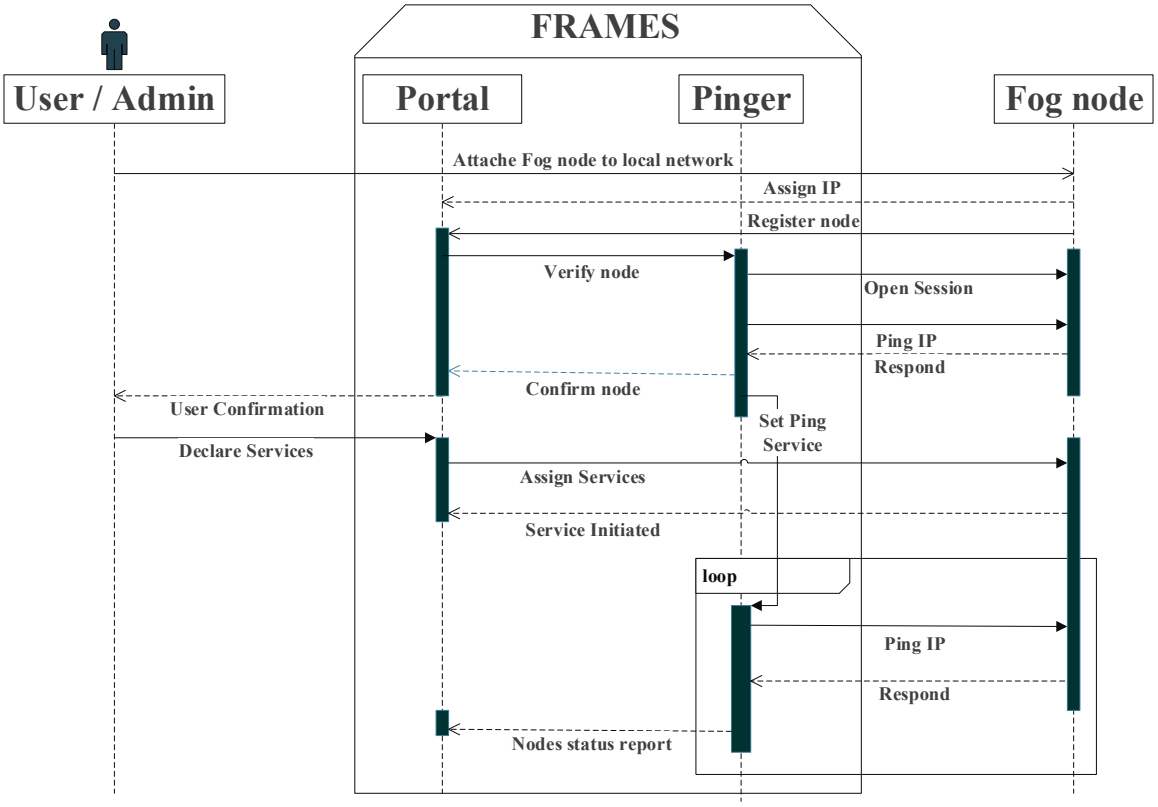


Figure 5.2: Sequence diagram showing FRAMES interactions

vice details in the fog portal. During the registration process, all device information and capabilities of the device are required, such as, device CPU clock, storage size, network capacity, MAC addresses identifier alongside with the IP address assigned by the network which will be used to identify the node. At this time, or thereafter, the fog owner can visit the portal and assign/declare the desired fog services.

- *Fog Pinger:* is an automated ping utility, which is run by FRAMES on a periodic schedule (set according network/admin needs) to check the status of registered fog nodes in each individual domain. The outcome will be reported to the main management portal upon which action is taken in case a fog node is down. The ping utility operate by sending Internet Control Message Protocol (ICMP) echo request packets which is very tiny in term of size, more precisely, it is 84 Bytes including the ICMP and IP headers [158], hence it dose not cause an overhead load in the network. Moreover, the pinger is network's admin feature that uses the services of the Internet Control Message Protocol (ICMP) which encapsulate in an IP header. Thus, pinger

operates on the Network layer of the Open Systems Interconnection (OSI) model.

- *Network Monitor:* part of the FRAMES duties is to monitor and control the computing resources of the fogs within the network. FRAMES tracks fog's resource consumption, maintains resource availability of each fog, and periodically reports to the administrator with an analytical report. Providing analytic and processed statistics to the services provider helps to efficiently maintain nodes resources and conditions to deliver services with high performance.

5.2.2 Fog Workload Balancing

Considering a scenario where a fog node accepts a data processing request from a thing; it will process the request and respond back. However, when the fog node is busy processing other requests, it may only be able to process part of the payload and offload the remaining parts to other fog nodes. Hence, there are two approaches to model interactions among fog nodes to distribute the load. First, the centralised model, which relies on a central node that controls the offload interaction among the fog nodes. Second, the decentralised model, which relies on a universal protocol that allows direct interactions among nodes. In the decentralised/distributed model, there is no need for a centralised node to share the state of fog nodes, instead, FRAMES can help each fog node run a protocol to distribute their updated state information to the neighbouring nodes. Then, each fog node holds a dynamically updated list of best nodes that can serve the offloaded tasks. The distributed model is more suitable for scenarios where things or fog nodes are mobile (e.g., Internet of moving things [159]) to support the mobility and flexibility of data acquisition. Therefore, we adopt this model of interactions in the $\mathcal{F}2\mathcal{F}$ coordination model. The procedure of sharing the overload among fog nodes is as follows:

- *When to offload a service request?* the decision of a fog node to support the processing of a received service request, part of the request or offloading the entire request to another fog is based on computing the response time of that fog. The response time of each fog will be computed periodically based on the fog's current load (i.e., queue size) and service request travel time (minimal latency always preferable). The procedure of offloading a received request by a fog node is as follows: once a service

request(s) is received by the fog node, it checks the request payload based on packet's size (i.e., heavy or light) and calculates the potential response time based on the current requests that are waiting, and also under-processing, in its queue. Meantime, the fog sends requests for coordination to all neighbouring nodes within its domain. It is worth noting that request-and-response times are considered part of the service latency. However, it is very low and even negligible in the overall service latency as the link rate among fogs is usually around 100 Mbps [14], which is very high. Packet's payload size is adopted as *heavy-weight* data packets (e.g. CCTV data) and *low-weight* data packets (e.g., sensors data) as it can be more accurate than naming a data type/format from an application, due to the fact that similar application may give different data payload sizes, also this approach is in line with [14, 160, 161]. The coordination request among fog nodes includes information about the type of service request received and/or awaiting processing; whereas the response from other fog nodes to the sending fog will be with time estimation for processing that request. Thereafter, if the estimated time by the fog is less than the expected response time by the thing (i.e., service deadline), the service will be accepted for processing and enter the queue of the fog. Otherwise, the fog will offload the service to another fog, which provides the lowest latency estimation, or redirects the service request to the cloud in case no fogs are available to handle the service. Simply put, offloading happens when a fog node has a heavy load. In the other extreme case when all fog nodes have heavy loads, offloading becomes useless. Thereof, it is more effective when there is a high load variance among participant nodes.

- *Where to offload a service request?* each fog has a list of best-suitable nodes with whom it can collaborate (i.e., reachability features table includes the estimated computing and response-time), when needed. This list is generated based on node's locations and their neighbouring nodes, i.e, the list will include all nodes that are directly reachable from the current fog node sorted by node distances from low to high. When a node is about to get or become congested[1], it can share the load with nodes from the list based on the payload size received. Thus, the list of best neighbour-

[1]The term "congested node" applies to any node that has a high traffic, which may cause a latency issue for the incoming service requests.

ing nodes is maintained periodically by each fog node. The process of selecting and sorting the best neighbour nodes is based on the possibilities of coordination between each other and being able to provide service processing with low latency and able to meet the deadline for the service request. Moreover, the procedure of selecting the best node takes into account the different request types as well as a node's capabilities and availability, thus, the list will be sorted by best node to the top, and best node is the one that can provide the lowest service latency and is available for coordination. The best node selection and offloading algorithms are explained in Section 5.3.10. It is worth noting that the list of best neighbouring should be updated not only periodically but also upon scenarios where a significant change occurs, such as, adding or removing node(s) to or from the fog domain. This helps keep the list accurate and avoid issues of inconsistency when there are changes within the fog domain. Therefore, the list should be updated on the following offloading occasions: (i) when the fog sends request of status updates to other fog nodes; (ii) adding a new fog node to the fog domain; (iii) removing a fog node from the fog domain; and (iv) when a fog node goes off-line. These interactions and management are handled by the FRAMES. To explain more, fog nodes can join and leave a fog domain by setting this through FRAMES by updating the fog portal to add/remove a fog node and the fog pinger utility to monitor the status periodically. Updating FRAMES may cause changes to the fog network topology, thus fog nodes within the affected domain will be notified by FRAMES to allow fog nodes re-sorting their list of best neighbouring fog node for coordination.

5.3 $\mathcal{F}$og-2-$\mathcal{F}$og Coordination Model

This section discusses the network model that supports $\mathcal{F}2\mathcal{F}$ coordination. It also discusses potential sources of delays that could impact this coordination. Mostly used notations in this chapter are given in Table 5.1.

5.3.1 Network Model

The communication among fog nodes in the context of $\mathcal{F}2\mathcal{F}$ coordination is modelled as an undirected graph, so that all fog nodes are reachable for each other. Having $G = \langle N, L, W \rangle$, where N is a set of thing, fog, and cloud nodes, thus, $G = N^I \cup N^F \cup N^C$ respectively. The notation L denotes the set of communication links between all nodes across the things, fog and cloud layers. While the notation W is the set of edge weights between nodes, according to the distance between them, hence the longer the distance, the higher the weight is. Thus, propagation delay D_p depends on the edge weight between two nodes.

5.3.2 Service Delay

A service request can be defined as a set of tasks that are processed completely to meet the desired service's requirements. Processing a service request can happen over any of the three layers (i.e., thing, fog, and cloud). Hereafter, FRAMES calculate the total delay taken to process a service. Service delay (S_d) for t_n request is expressed in Equation 5.1:

$$\begin{aligned} S_d = \rho_i^F * [D_t^F + D_p^F + D_c^F] \\ + \rho_i^C * [D_t^C + D_p^C + D_c^C] \end{aligned} \tag{5.1}$$

Where ρ_i^I is the probability that t_n processes the data locally at the things layer, ρ_i^F is the probability of processing the service at the fog layer, and ρ_i^C is the probability that the service is processed at the cloud layer; $\rho_i^I + \rho_i^F + \rho_i^C = 1$. S_p^I is the average processing delay of the t_n when it processes data. D_p^F is propagation delay, and D_t^F is the sum of all transmission delays. Similarly, D_p^C is propagation delay for cloud server, and D_t^C is the sum of all transmission delays to the cloud.

Table 5.1: Notations used in the thesis

Symbol	Description
t, n, T	thing, index of t, set of things
f, i, F	fog, index of f, set of fogs
λ	service arrival rate to fog layer
μ	fog node service rate
ρ_i^F	probability of sending the request to the fog
ρ_i^C	probability of sending the request directly to the cloud
ρ_i^I	probability that t processes the data locally
D_t	transmission delay
D_p	propagation delay
p_s	propagation speed
D_c	computational delay
D_{que}	queuing delay
D_{proc}	processing delay
l_p	packet size in bits
$b \uparrow$	upload bandwidth
$d_{f_i^c}^{ts}$	total delay by f_i to process task ts, and c refer to f_i capacity
S, s	Set of services, one service
s_w	service workload
s_d	service deadline
τ_s	total time required to process a service
τ_{que}	is the queuing time
τ_{proc}	service processing time
ρ	system usage
ϱ_{size}	queue size
τ_{que}^{si}	queuing time for s at the resources of fog f_i
f_w	fog workload
f_i^c	processing capacity of the fog node F_i
$\tau_{s_w}^{f_i}$	time to process s_w on f_i
nS_l	number of light services
nS_h	number of heavy services

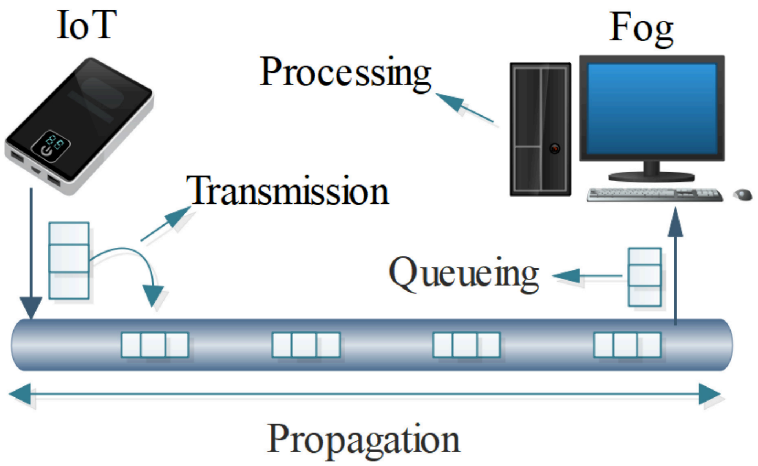


Figure 5.3: Four sources could delay service processing

5.3.3 Delay Sources

Figure 5.3 shows four delay sources; transmission delay (D_t), propagation delay (D_p), queuing delay (D_{que}) and processing delay (D_{proc}). These delay sources can seriously impact service performance and meeting deadline, hence causing latency. To correctly calculate the delay, it's important to be clear about where the service will be processed and what parameters are involved in the processing. Therefore, the focus of FRAMES is on minimising service processing latency over the fog layer, via $\mathcal{F}2\mathcal{F}$ coordination, hence achieving minimal service transmission delay (D_t), propagation delay (D_p), and computational delay (D_c) which includes both queuing delay (D_{que}) and processing delay (D_{proc}).

5.3.4 Transmission Delay

Transmission Delay (D_t) is the time taken by a sender (i.e., thing) to transmit the data packets over the network. To calculate the transmission time that is required by a particular thing, we should know the packet size or packet length l_p in bits and data rate (i.e., upload bandwidth) $b \uparrow$. Thus, the sum of transmission delay D_t^n for thing t node index n is calculated using Equation 5.2.

$$D_t = \frac{l_p}{b \uparrow}$$

$$D_t^n = \sum \frac{l_p^n}{b\uparrow} \tag{5.2}$$

$b\uparrow$ is the upload bandwidth which refers to the maximum data rate in *bps* (bits per second) at which the sender can send packets on the network link. The transmission delays between other layers, such as fog to cloud, are calculated using the same approach and based on l_p and $b\uparrow$.

5.3.5 Propagation Delay

Propagation Delay (D_p) is the time required to transmit all data packets over a physical link from source (e.g., thing) to destination (e.g., fog). The delay will be computed using the length of the physical link to destination l_d and propagation speed p_s. The l_d can be calculated using the latitude and longitude of the thing and fog to find out the length. Thus, the propagation delay D_p^n for a t_n can be calculated using Equation 5.3. The propagation delays between other layers, such as fog to cloud, are calculated using the same approach in Equation 5.3 and based on l_d, and p_s.

$$D_p^n = \frac{l_d^n}{p_s} \tag{5.3}$$

5.3.6 Computational Delay

Computational Delay (D_c) is the total time taken by f_i to compute a service requested by t_n. This time includes both queuing delay (D_{que}) and processing delay (D_{proc}). The D_{que} is the period of time spent by a data packet inside the queue/buffer of a fog node until it gets served. While, the D_{proc} is the time consumed by the fog node to process the received data/packet(s). The D_c will give the actual time required for the service request to be processed according to the fog node's capability and its current load.

Moreover, as mentioned before, IoT requests can be defined as a set of sub- /tasks, thus, these tasks can be processed in a sequential manner, parallel manner, or a mix. Figure 5.4 demonstrates the different possible approaches for processing a service. For a service with sequential tasks the process delay is the sum of all task delays, while the process delay for a parallel processing will be the maximum latencies among all tasks. Therefore, the

processing delay for a service that can be processed immediately without waiting in the queue will be calculated using Equation 5.4.

$$D^{s}_{proc} = max_{q \to Q_s}(\sum_{t \in Q^{q}_{s}} d^{ts}_{f^{c}_{i}}), \forall q \in Q, \forall c \in C \tag{5.4}$$

Where $d^{ts}_{f^{c}_{i}}$ is the total time delay consumed by f_i to process task ts, which belongs to the service s with processing sequence q, and c denotes the total capability (i.e., CPU) of f_i. As mentioned before, Equation 5.4 is used to calculate the total time-delay when a service is immediately processed by a fog node.

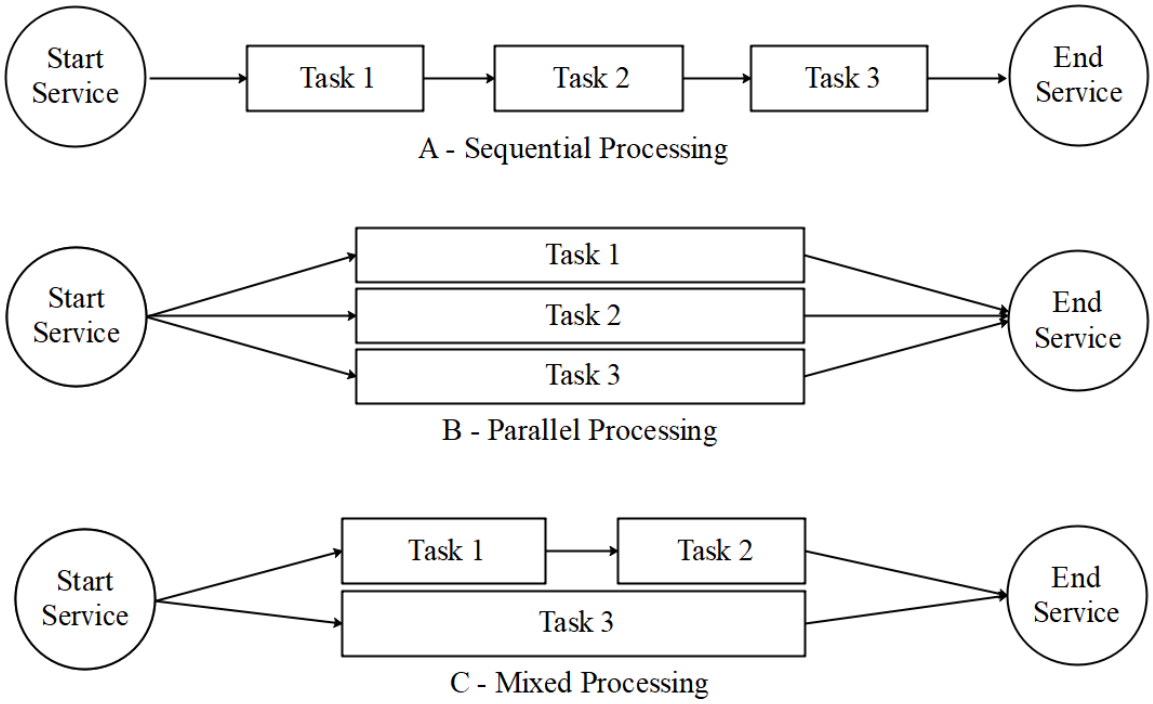


Figure 5.4: Three types of service processing

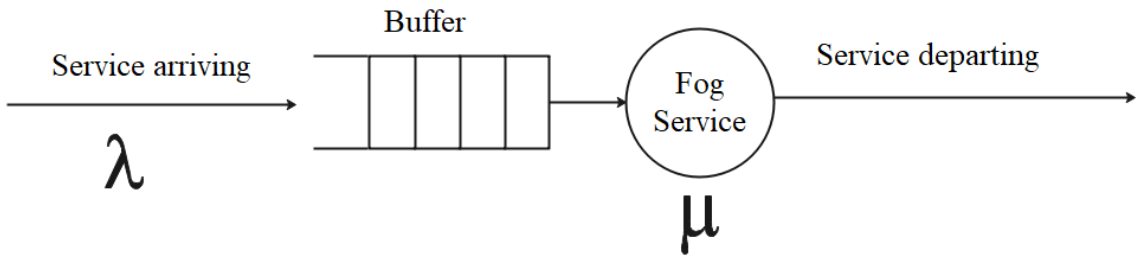


Figure 5.5: Queuing system

Next, we will discuss the scenario when a service request arrives at the fog and has to wait in a queue due to the fog's current load. When the fog is congested (i.e., busy) the arriving services are queued in the fog buffer until the fog becomes available to process the received requests according to priorities. In this case the key factor for service latency will be the average waiting time of a service at the buffer, which is based on the length of the buffer/queue, in addition, the processing time for the services are as per Figure 5.5, where λ is the average service arrival rate and μ is the average serving/processing rate for a fog.

In any queuing system, the network can be modelled using three parameters $A/P/n$, which according to Erlang-C [1] these are; A is service arrival rate, P is the service time probability density, and n the number of fog nodes. Therefore, we model the fog system network in a similar approach since it has queue/buffer within its network topology. Hence the fog network is modelled as as $M/M/n$, where the first M is the services arrival rate according to the Poisson process with average rate λ_i for f_i. The second M is the indication of service rate exponentially distributed over n number of fog nodes and having the mean service of $1/\mu$. In the fog system, n is the set of heterogeneous fog nodes with different capabilities. Thus, when $n > 1$ the first service in the queue will be served by the fog that is currently available (i.e., $queue = \phi$) and will process the service, or offload it to the first node that becomes available through a periodic checking of the reachability table within the fog domain. The total time for a service, is the time for queuing τ_{que} and processing τ_{proc} as follow:

$$\tau_s = \tau^s_{que} + \tau_{proc}$$

Hence, the total time for τ_s can be computed by Equation 5.5.

$$\tau_s = \left[\sum_{x=0}^{n-1} \frac{(\frac{n}{x})!(1-\rho^2)}{(n\rho)^{n-x}} + \frac{1-\rho}{\rho} \right]^{-1} \tag{5.5}$$

where ρ is the system utilisation, obtained using Equation 5.6.

$$\rho = \frac{arrivalRate}{serviceRate} = \sum_{x=1}^{n} \frac{\lambda}{\mu_x} \tag{5.6}$$

The μ can be obtained by $\mu = \frac{l_p}{L_c}$ having l_p average packet size in bits, and L_c is the link transmission capacity (unit is bits/second). It is worth noting that the inverse of service rate is the average service time $\frac{L_c}{l_p}$. To find a queue size and compute the average number of service packets in the queue we use Equation 5.7:

$$\varrho_{size} = \frac{\rho(\frac{[P_s^w(n,\rho)](n\rho)^n}{n!(1-\rho)})}{1-\rho} \tag{5.7}$$

Where P_s is the probability of number of service packets in the fog system and calculated using Equation 5.8:

$$P_s^w(n,\rho) = \left[\sum_{x=0}^{n-1} \frac{(n\rho)^x}{x!} + \frac{(n\rho)^n}{n!(1-\rho)}\right]^{-1} \tag{5.8}$$

Equation 5.8 provides the probability of the newly arrived packets that are not processed immediately in the fog layer and, thus, have to wait. Hence, to obtain the probability of packets that are directly processed we use Equation 5.9.

$$P_s^d = 1 - (P_s(n,\rho)) \tag{5.9}$$

Next, we calculate the average delay for a service packet in a fog's queue. This will help evaluate the performance of fog by the FRAMES and point out the congested node based on ϱ_{size} and queuing time τ_{que} for a process. Thus, the queuing time for a service request is calculated using Equation 5.10.

$$\tau_{que}^{si} = \frac{\rho(\frac{P_s^w(n\rho)^n}{n!(1-\rho)})}{\lambda - \lambda\rho} \tag{5.10}$$

Where τ_{que}^{si} is the queuing time τ_{que} for service s at the resources of fog node i, λ is the service rate and ρ is the system utilisation. To compute the total time for a service's request

in the fog system, its generally by adding the processing delay to τ^{s}_{que} as per Equation 5.11.

$$\tau^{si}_{c} = \tau^{si}_{que} + \frac{1}{\mu} \tag{5.11}$$

5.3.7 Fog Workload

Fog workload f_w refers to the overall usage of a fog node's CPU as *cycles* per *second*, which is consumed during the processing of a particular service request. Thus, there is a limits and constraint for node capability, which leads to a limitation of the abilities for processing different type of services. (i.e., heavy or light). Therefore, the workload assigned to a fog node f_w should not exceed the total capacity of the fog node f^{c}_{i} at anytime.

$$f_w \leq f^{c}_{i}, \forall f \in F \tag{5.12}$$

A service that operates/runs or is provided by a fog node can serve several end-users in the network. Thus, the total ratio of CPU usage by a service task (or tasks in case of parallel processing) should not exceed the total resources allocated for that specific service. This is because these allocated resources are considered to be the total f_w that can be provided by this specific fog node for this particular service. Equation 5.13 computes the total resources (rs) allocated to process all tasks ts for a service s.

$$f^{s}_{rs} = s_w = \sum_{t=1}^{n} C^{f_i}_{ts}, \lceil s \rceil \leq f^{i}_{c}, \forall s \in S, \forall t \in T_s \tag{5.13}$$

The total fog's workload capacity (f_c) depends on the actual hardware specification of the allocated device. The assignment variable s_w (i.e., total service workload) is set so that total service processing workload does not exceed f_c, as per Equation 5.13, where $C^{f_i}_{ts}$ denotes the total resource (CPU in consumption in hertz, having *hertz=cycles/second*) consumed by a service's tasks on fog node f_i.

For more realistic scenarios, the services workload has been separated depending on the service request type, having a heavy-weight and low-weight service request according

to service packet's size. For instance, when a service only processes a small data packets from sensors, this will consume low computational power, thus, the workload on fog is low. While, in services that perform heavy real-time video processing, the workload will be high on this fog node. Therefore, services workload (s_w) on fogs can vary for each service depending on service type. The f_w for all services is the sum of each service workload multipled by λ as per Equation 5.14. Thus, f_w should be less than the f_c assignment variable (i.e., $f_w <= f_c$).

$$f_w = \sum_{x=1}^{n} s_x^w . \lambda_s, \forall s \in S \tag{5.14}$$

5.3.8 Average Delay in a Fog Node

Fog node is a device located within the local network and equipped with communication protocol and computation power. We assume that nodes at the fog layer receive service packets from IoT nodes for processing and it has enough buffer size to accommodate the incoming packets. Thus, the services arrival λ traffic to fog nodes will be according to Poisson and fog processing rate is exponentially distributed over fog nodes according to *light-services* processing (μ) and *heavy-services* processing rate (μ'). To compute the waiting for a service packets on a specific fog node, it will be through calculating the total time for processing the current heavy and light services in the fog buffer/queue. For example, to obtain/calculate the average waiting time for a service s that arrived at f_i at a specific timestamp, it will be through the total time consumed by f_i to process all current service's packets according to their types. Equation 5.15 computes the average waiting time for a newly arrived service on f_i, having nS_h refer to the number of heavy-services and nS_l refer to the number of light-services.

$$\begin{aligned} nS_h &= \sum s_h^{f_i}, \forall s_h \in S \\ nS_l &= \sum s_l^{f_i}, \forall s_l \in S \\ \tau_{s_w}^{f_i} &= \frac{nS_h}{\mu'} + \frac{nS_l}{\mu} \end{aligned} \tag{5.15}$$

It worth mentioning that, if f_i queue is not empty, i.e., we have

$$(nS_h + nS_l) \neq \phi$$

Then the

$$queue = (nS_h + nS_l) - 1$$

This means that there are mixed types of service packets currently in the buffer/queue and only one packet is currently in processing.

5.3.9 Problem Formulation and Constraints

It is crucial to guarantee minimal service delay to end-users during service processing at the fog layer. The four sources of delay mentioned in Figure 5.3 are included in the latency minimising schema. The total latency for a service sent from t_n to f_i is computed by adding the time of uploading a service's packets ($\tau_{\uparrow}$) to the waiting time for the service in the fog queue (τ_{que}) until it gets processed. The delay for processing the service (τ_{proc}) and the time to respond back ($\tau_{\downarrow}$) to t_n is also added to the total latency for the service as per Equation 5.16. For simplification, we assume that($\tau_{\uparrow}$=$\tau_{\downarrow}$), having ([$\tau_{\uparrow}$=$\tau_{\downarrow}$]=2$\tau_{\downarrow\uparrow}$) because logically the returned packets are normally a similar or smaller size than the sent packet.

$$\tau_s = \tau_{\uparrow} + \tau^s_{que} + \tau_{proc} + \tau_{\downarrow}, \forall s \in S$$

$$\tau_s = \tau^s_{que} + \tau_{proc} + 2\tau_{\downarrow\uparrow}, \forall s \in S \tag{5.16}$$

We address the problem of having an optimal workload on fog nodes alongside achieving minimal delay for IoT services. Thus, achieving a reasonable load includes executing/processing the desired services within the threshold limit of fog capability. In addition, low latency for IoT services includes delivering the service results within the required period, i.e., before service deadline (s_d) with the desired QoS and QoE. Therefore, the research problem in 5.17 indicates that the maximum time required to process a service τ_s should not exceed the service deadline s_d.

$$P: \quad max[\tau_s] \leqslant s_d, \forall s \in S \tag{5.17}$$

$$\text{s.t.} \quad f_c^{min} \leqslant f_w \leqslant f_c^{max} \tag{5.18}$$

$$\sum \lambda_s \leqslant \sum \mu_f \tag{5.19}$$

$$P_s^d(n,p) \geqslant serviceLevel \tag{5.20}$$

$$\lambda_s \xrightarrow{min[D_p]} f_i \tag{5.21}$$

$$\tau_s \leqslant s_d, \forall s \in S \tag{5.22}$$

The constraints are on reducing service latency. Therefore, the constraints are written with the focus on achieving minimal service delay. Constraint (5.18), indicates that (f_w) is strictly bound by an upper limit (f_c^{max}) and lower limit (f_c^{min}) which is related to fog capabilities based on CPU frequency (unit $hertz$). Constraint (5.19) imposes that the total traffic arrival rate (λ_s) to a fog domain should not exceed the service rate (μ_f) of that specific fog domain. Constraint (5.20) imposes that the probability of directly processed services should be greater than or equal to the desired service level. Constraint (5.21) imposes the first destination for the IoT thing node's packets generated will be to a fog node with minimal cost of propagation delay within the fog domain. Ideally, lowest propagation delay is for the nearest fog node. Finally, constraint (5.22) is strictly bound to the service time τ_s within the limit of service deadline s_d.

5.3.10 Offloading Model

The offloading model proposes to balance the load within the fog domain by distributing service traffic from the congested fog nodes to other fogs within the domain. To balance services traffic in fogs domain, we assume that fogs at any given location are reachable to each other within the same fog domain as per our network model in Section 5.3.1, which models the fog network as a mesh network; this assumption is in line with the work in [3] and [162]. In this research, we consider a real-world scenario of service flows where services arrival rates can significantly vary from one fog node to another [3] depending on fog location, This consideration from constraint 5.21 ($\lambda_s \xrightarrow{min[D_p]} f_i$) that is services are

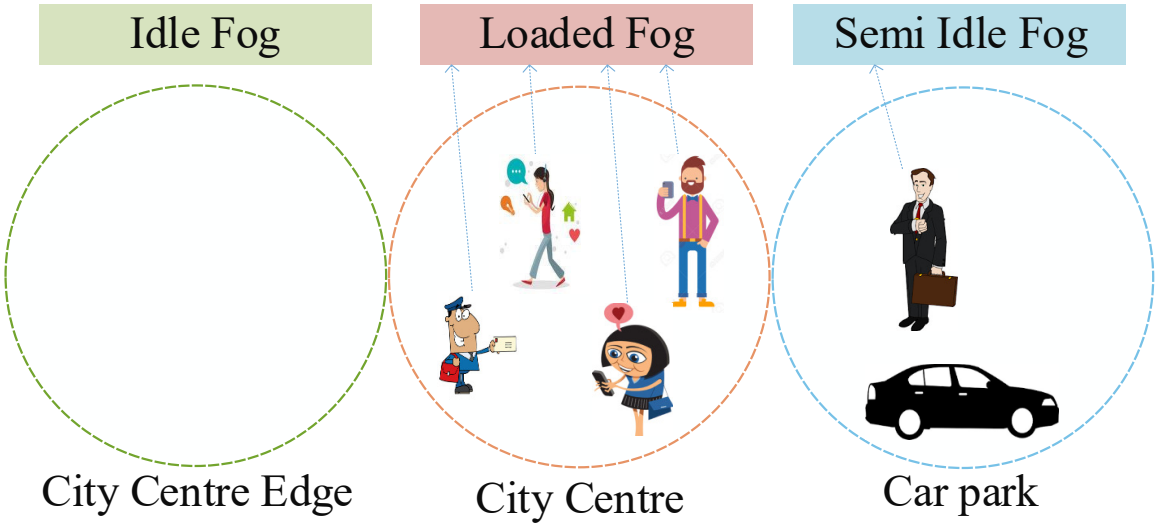


Figure 5.6: Loaded, idle and, semi-idle fog nodes based on $\lambda_s \xrightarrow{min[D_p]} f_i$

directed to the nearest fog from the thing for processing. Hence, Figure 5.6 demonstrates the scenario where fogs can vary in their traffic load due to their geo-location. In a similar scenario, offloading the traffic from loaded fog node to idle fog node can be crucial to mitigate the load and keep the service latency at the minimal.

For example, given that only mobile vehicles are considered in traditional VANET, the authors in [163] discuss how mobile vehicles (which are loaded nodes) and parked vehicles (which are idle or semi-idle nodes) should work together "as fog nodes" to transmit information and process requests to minimise the network load on mobile vehicles, this to increase efficiency and reduce latency. It should be noted that the latency (time variable) and money variable have a linear relationship with each other - they impact directly on each other. For example, in intelligent transportation systems discussed in [164], the vehicular communications prove reducing the traffic congestion and, hence, the Round-trip Delay Time (RDT) thereby cutting down the fuel consumption (money variable).

The decision factors where a node is congested and offloading is significantly required to aid fog workload (f_w) are associated with the service traffic arrival rate (λ_s) and total processing rate (i.e., service rate μ) which is down to fog CPU frequency (i.e., node capacity). In addition, the service processing time τ_s ideally should not exceed service deadline (s_d). Therefore, to make the decision of offloading by a fog node is when having $\tau_s > s_d$, as

per Probability 5.23, having (O_s) refers to the offloading service decision:

$$O_s = \begin{cases} 1, & \text{if } \tau_s > s_d \\ 0, & \text{otherwise} \end{cases} \tag{5.23}$$

Thus:

$$\tau_s > s_d, \forall s \in S$$

$$\tau^s_{que} + \tau_{proc} + \tau_{\downarrow} > s_d$$

In Probability 5.23, O_s value is set to either 0 or 1, where 0 refers to no offload is required, while 1 refers to offloading is significantly required as the newly arrived service will suffer from latency and will not be able to meet the service deadline s_d. Hence, service offloading is required to aid minimising fog workload and meantime avoid service delay to end-users.

Algorithm 1: MAINTAIN FOG LOAD

Input: Fog (F_i); FogCapacity (F_c); QueueSize (Q_s)
Parameters : Offload (O_s); OverLoad (O_l); Services (S); ServiceType (S_t)
Initialisation $F_i = \phi$; $F_c = \phi$; $Q_s = \phi$; $S = \phi$
.
.
Result: Determine Fog overload, if any.

Procedure 1. *Overload Threshold* **by**
  $F_c = F_i^c$ ▷ F_c `initiate fog`
  $Q_s = \longleftarrow getQueueSize(F_i)$
  $S = list\{Q_s\}$ ▷ `get list services`
  $S = sort(S, \text{by } S_t)$
  for *each* $s \in S$ **do**
    $\tau_c^{si} = \tau_{que}^{si} + \frac{1}{\mu}$
    if $(\tau_c^{si} \geq S_d) \mid\mid \lambda \geq \mu)$ **then**
      $setFlag(O_s) = 1$
      $break$;
    else
      $setFlag(O_s) = 0$
    end
  end
  $F_{que} = timeCostFun(s, \tau_c^s)$
  $F_i \longleftarrow F_{que}$
  return (F_i, O_s)
End
Procedure 2. *Determine the Overload* **by**
  get (F_i, O_s)
  $F_c = getCapaxity(F_i)$
  $\mu = \frac{F_c}{F_{que}^i}$
  if $(O_s == 1 \mid\mid \lambda \geq \mu)$ **then**
    for *each* $s \in F_{que}$ **do**
      $S = getServices(out : s \leftarrow \tau_c^s \geq S_d)$
    end
    $F_{que} = F_{que} - S$
    $O_l = S$
  else
    $get(F_i, O_s)$
    continue
  end
  return O_l
End

Algorithm 1 has been developed to detect the fog nodes that suffer from the congestion issue, and determining the overload packets that needs offloading. The goal of this algorithm is to answer the question of *When to offload?* and *What to offload?*. The first part of the algorithm (Procedure 1) determines if the fog node is congested or not. This starts by getting fog queue size and queued services sorted by their types (i.e., heavy-services and light-services) as per lines 1-5. Later, lines 6-8 examine if one or more services in the queue will miss their deadline S_d, or if the service arrival rate λ is bigger than the outcome of the fog node μ (i.e., fog service rate). If any of the conditions is satisfied, a flag indicates that the fog node is congested as per line 9. The second part of the algorithm (Procedure 2) determines the overload by computing the number of service requests that are causing the congestion as per lines 24-26. The overload O_l will be held in a list that contains reference to all service requests that require offloading to other fog nodes as per lines 27-28. It worth noting that there is no intermediate processes to be executed between procedure 1 and 2, hence procedure 2 run immediately after procedure 1. The outcome of this algorithm will feed into Algorithm 2.

To balance the services on fog nodes and to achieve optimal workload and minimal service delay, the offloading to the best available fog node is adopted, so that, the best available fog node can deliver the desired services within the scheduled time (i.e., $\tau < d_s$). Therefore, to obtain the best node, which will handle the overload, we compute the service time τ_s for the services requiring offloading among all available nodes using Equation 5.24, thus, having some constraints on the node that participates in the process to handle the overload such as load limit.

$$min[\tau_s] = min \sum_{i=1}^{n} [\tau_{que}^{f_i} + \tau_{proc}^{f_i} + \tau_{\downarrow}] \tag{5.24}$$

$$\text{s.t.} \quad f_c^{min} \leqslant f_w \leqslant f_c^{max}$$

$$\sum \lambda_s \leqslant \sum \mu_f$$

$$\tau_s \leqslant s_d, \forall s \in S$$

The best available nodes are those that provide a service with minimal delay. To find these fog nodes, Algorithm 2 is developed. Algorithm 2 will find the best fog node to handle the overload on the congested fog node, and then offload the overload from the congested fog node. In addition, the goal of the algorithm is to answer the question of *Where to offload?*.

Algorithm 2: SERVICE OFFLOADING

Input: FogNode (F_n); FogLoad (F_l); OverLoad (O_l).
Parameters: FogCapacity (F_c); Propagation (D_p).
Initialisation $F_n = \phi$; $F_c = \phi$; $F_l = \phi$; $O_l = \phi$.
⋮
Result: Share the Overload with best available node

Procedure 1. *Determine best available node* **by**
  $F_L = list\{\phi\}$ ▷ F_L `initiate fog list`
  $F_L = list[F_n] \longleftarrow getFogNodes(out : (F_n, F_c))$
  $F_L = sort(F_L,$ by F_c DESC$)$
  for *each* $F_n \in F_L$ **do**
    if $Fn \longleftarrow (F_l \geq Fc_{max})$ **then**
      $F_L = pop(Fn)$ ▷ `remove busy node`
    else
      $\tau_s = \sum_{i=1}^{n}[\tau_{que}^{i} + \tau_{pro}^{i} + \tau_{\downarrow}]$
      if $(\tau_s < s_d)$ **then**
        $list.add(F_n, \tau_s)$
        continue
      else
        $F_L = pop(Fn)$
      end
    end
  end
  return F_L
End
Procedure 2. *Handover the Overload* **by**
  if $F_L \neq \phi$ **then**
    $F_n = min[F_L(\tau_s, D_p)])$
    $F_l^n = F_l + O_l$
  else
    `goto:` **Procedure 1;**
  end
End

The first part of the Algorithm 2, Procedure 1, shows the process of finding the best available node(s) for handling the overload pointed to in Algorithm 1. Lines 2-3 of the algorithm initiate the list of active fog nodes in the domain alongside the node's capacity and current load (i.e., queue size). The list of available fog nodes will be refined by removing the nodes that are already busy with other services (i.e., $\lambda_i = \mu_i$) as per lines 6-8. The remaining part of Procedure 1, lines 9-18 compute the time required for a service request to be run on each of the available fog nodes. If the time is within the limit allowed for the service (i.e, before S_d), the algorithm will keep the fog node in the list and log the expected service time against the fog node ID as per lines 9-12. If the τ_s on F_n is greater than S_d, then F_n will be removed from the list as per lines 13-15. The second part of Algorithm 2, Procedure 2, receives the list of best available nodes. If the list is not empty, that means there is at least one fog node that is able to take the overload for processing. However, if there is more than one node in the list, the system will direct the overload to a fog node that can provide minimal τ_s and has the lowest propagation delay D_p as per lines 21-23. It worth noting that there is no intermediate processes to be executed between procedure 1 and 2, hence procedure 2 run immediately after procedure 1.

5.4 System Evaluation

In this section, the $\mathcal{F}og$-2-$\mathcal{F}og$ coordination model is evaluated through a MATLAB based simulation. The simulation setting and functions are built according to FRAMES which is about providing optimal fog workload with minimal latency for IoT services. A scientific and comprehensive network latency has been calculated, including time delays to compute heavy-packets, light-packets, mixed types of packets and latency per fog node according to their capacities. This is to demonstrate the superior performance of the proposed $\mathcal{F}og$-2-$\mathcal{F}of$ coordination model. The results have been validated against two benchmark algorithms; Random Walk Algorithm (RWA) [132, 133], and Neighbouring Fogs Algorithm (NFA) [165]. Simulation settings are presented in the following subsection, followed by a discussion of the achieved simulations results.

5.4.1 Experiment Configurations

This section describes the adopted MATLAB simulation settings along with the setup parameters. The configurations settings are according to the model proposed in Section 5.3, hence it specifies the network topology, propagation and transmission delay, link bandwidth and fog nodes capabilities, as follows:

- Network topology: this has been modelled as an indirect graph the represents fog mesh network at the fog layer. Fifteen fog nodes ($f_n = 15$) were used in the simulation and remain the same topology with 15 fog nodes throughout all experiments and during the evaluation of all algorithms. These nodes are connected together through internal communication link based on links transmission speed. Moreover, the links between nodes are weighted based on the propagation time between nodes, for instance, if D_p between fog_1 and fog_2 is two second, then the link weight between both nodes is ($fog_1 \underset{\longleftrightarrow}{2} fog_2$). Also, the services arriving at the fog layer are assigned to fog based on the smallest D_p between the node and source, which has the smallest distance. It worth noting that there is no explicit effect/changes of using random topology (i.e., fog nodes can join and leave during run-time) as the FRAMES, using the portal and pinger utilities, will notify other fog nodes when an updates is available. Thus, when a fog node get congested and needs to offload a request, it will have access to only fog nodes reported by FRAMES and no matter whether they are 10, 15 or 20.

- Network bandwidth: link bandwidth depends on the type of service, thus, *heavy-packets* provided by heavy services will require more bandwidth compare to *light-packets* generated by light services. Therefore, for light-packets (e.g., data packets from sensors) the communication bandwidth used has a transmission rate of 250 $Kbps$ [161], which is equivalent to 2.0×10^6 $hertz$). Such communication protocol is the IEEE 802.15.4, and ZigBee. While for heavy-packets (e.g., data packets from camera) the communication bandwidth used with a transmission rate of 54 $Mbps$ [160], which is equivalence to 4.3×10^8 $hertz$) [160]. Such communication protocol is the IEEE 802.11a/g. The transmission rate between the fog nodes is expected to be higher, around $\simeq 100$ $Mbps$ [14].

- Transmission and propagation delays: the transmission delay D_t for a packet depends on the packet size l_p alongside the associated upload bandwidth b$\uparrow$. Hence, impose an average packet size that will vary according to the type of packet (i.e., heavy and light packets). The average packet size for light-packets is 0.1 KB, while the average packet size for heavy-packets is 80 KB [14]. With regard to the propagation delay D_p, the packet round trip time (i.e., $\tau_{\downarrow\uparrow}$) adopted and inline with [14] by having:

$$\tau_{\downarrow\uparrow} = 0.03 \times l_d + 5$$

  Where l_d is the distance with unit km, and $\tau_{\downarrow\uparrow}$ time unit is ms.

- Fog node capabilities consider the service rate μ that varies from one fog node to another. The capability of a fog node will highly affect the processing capacity (i.e., performance) of the fog node. Thus, a fog node's capability is determined by CPU frequency. hence a fog node's CPU variant and the range between 0.2 GHz to 1.5 GHz [166].

5.4.2 Benchmark Algorithms

In order to validate the results achieved by the proposed $\mathcal{F}og$-2-$\mathcal{F}og$ coordination model and the offloading algorithms, two benchmarks algorithms have been considered:

1. Random Walk Algorithm (RWA) [132, 133], which imposes that arriving service requests are assigned to the nearest fog node to the data source. If the fog is congested it will offload the service randomly to another fog node. In this scenario. This makes the assumption that each fog node within the domain has the same probability of being selected.
2. Neighbouring Fogs Algorithm (NFA) [165], which imposes that the congested fog node will offload the overload to the nearest fog node with bigger capacity.

Moreover, our comparison also includes the typical service distribution based on assigning service's packets to the nearest node to the IoT thing with No Offloading Algorithm (NOA). We refer to the proposed offloading algorithm as Optimal Fog Algorithm (OFA).

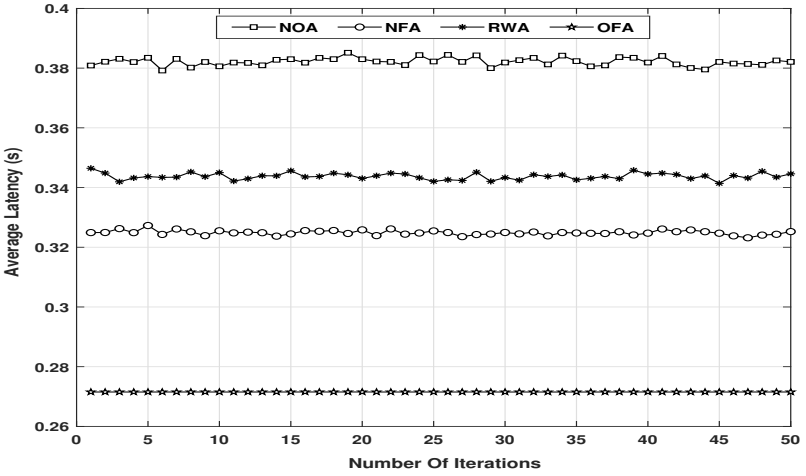
5.4.3 Performance Evaluation and Discussion

The performance metric we used is the average service time that reflects the efficiency of service completion time (*aka* amount of delay/latency). The lower the average service time ($min[\tau_s]$), the better the efficiency of service and the QoS and QoE.

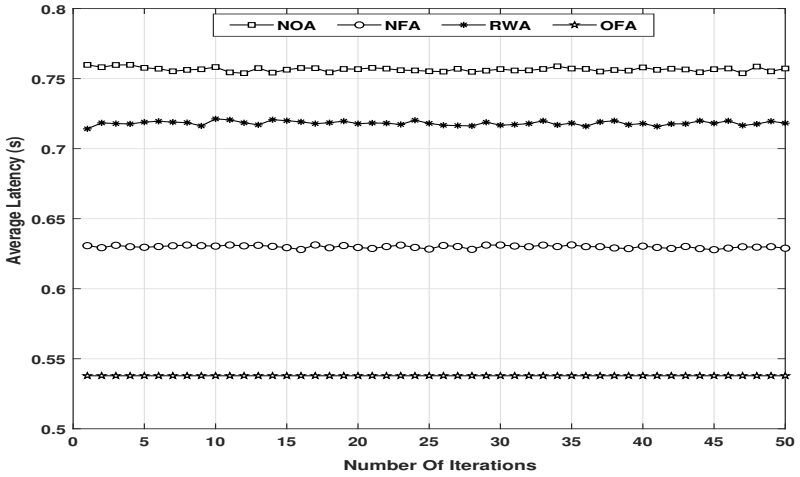
Figure 5.7 illustrates the performance of our OFA based on the average response time for all received service requests according to a service's packet types. Also, it provides a comparison between the results of OFA and the results obtained from other algorithms mentioned in Section 5.4.2. The simulation settings for this experiment is as follows:

- Fog nodes with different capabilities, hence, nodes vary in their service rate μ.
- Fog nodes capability based on CPU frequency with a minimum of 200×10^6 hertz, incremented by 100 hertz until it gets to maximum CPU capability of 15×10^8.
- Service arrival rate $\lambda = 3 \times 10^2$ packet per second as in [3], and λ is fixed during the experiment to ensure all algorithms have the same traffic arrival rate.

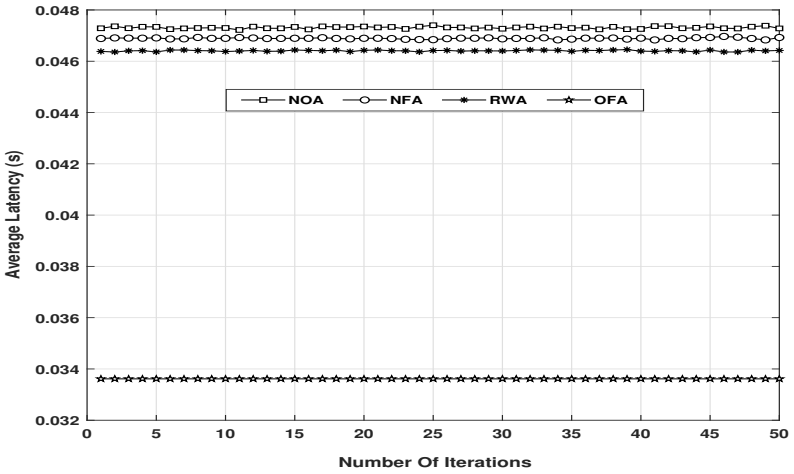
Figures 5.7a, 5.7b, and 5.7c are grouped by packet types, having heavy-packets versus light-packets versus mixed-packets. In Figure 5.7a, the packets type is mixed (MTP), having a random number of heavy and light packets. However, the random number is fixed through out the experiment to ensure consistency across all algorithms. In Figures 5.7b and 5.7c, the packets are set to either all heavy-packets (AHP) or all light-packets (ALP). This is to examine the performance based on different scenarios. In Figure 5.7 the vertical line represents the average latency per algorithm to serve all arriving services, and the horizontal line is the number of iterations carried out to ensure that the obtained results are consistent and not random. It is clear that OFA has the lowest service latency among other algorithms through all iterations and with all types of packets. It is obvious that NOA has the largest service time because it does not consider offloading when a fog node becomes congested. Hence, we end-up having a small node capacity with large queue size (i.e., $\mu_i < \lambda_i$), and a large node capacity with low queue size. The performance of RWA and NFA are better than NOA but still higher than our OFA. However, RWA has the worst performance with MTP and AHP as it randomly offloads the overload, which is a relatively blind algorithm as it does not consider the current fog workload (f_w) and the propagation delay (D_p) between



(a) Mixed types of packets (MTP)



(b) All heavy packets (AHP)



(c) All light packets (ALP)

Figure 5.7: Average latency according to offloading model

sender and receiver. It worth noting that the OFA results in Figure 5.7 are mostly steady because the evaluation has been done over 50 iterations and in each iteration the mean value of processing all packet is taken, hence the mean mostly steady, as in most other algorithms in Figure 5.7b and 5.7c.

The next simulations were conducted based on service latency per fog node. Similar to previous experiments, we use fog nodes with different capabilities based on CPU frequency with a minimum of 200×10^6 hertz, incremented by 100 hertz until it gets to maximum CPU capability of 15×10^8, having $F_n = 14$. In this simulation, we increment the service arrival rate so, that, the total packet received is one million service requests. The packet type in this experiment is mixed, having a random number of heavy-packets and light-packets. Figure 5.8 shows the average latency per fog node. It is clear that OFA achieves a consistent average latency. In contrast between OFA, on the one hand, and NFA and RWA, on the other hand, OFA has the lowest average latency between fog nodes 1 to 7, but greater average latency from node 8, and thereafter. However, the average latency difference is much higher for NFA and RWA in comparison to OFA for fog nodes from 1 to 7 compared to the average latency differences from node 9 to 14. This difference accrues as OFA workload distribution strategy, OFA tries to achieve balanced service distribution based on node capacity. Therefore, the work assigned to fog nodes considers the overall capacity and current load before it offloads a request, while NFA and RWA are relatively blind in this manner. Hence, OFA achieves almost consistent latency on each individual node, while the average latency for NFA and RWA vary and are inconsistent.

To prove the optimal distribution of packet with OFA we run a new experiment and recall the settings from the previous experiment. However, in this experiment, the vertical line represents service usage (i.e., number of packets) as per Figure 5.9. The fog nodes are sorted from smallest capacity (i.e., lowest CPU) to largest (i.e., largest CPU), having the first node with 200×10^6 hertz and node 14 with 800×10^6 hertz. It is clear that the packets distribution with OFA is completely different from NFA and RWA as it distributes packets according to the fog node capacity. Hence, the first node receives fewer packets and the last node receives more packets. In comparison with NFA and RWA, the packet distribution on average is steady among all fog nodes regardless of the node capacity, which causes

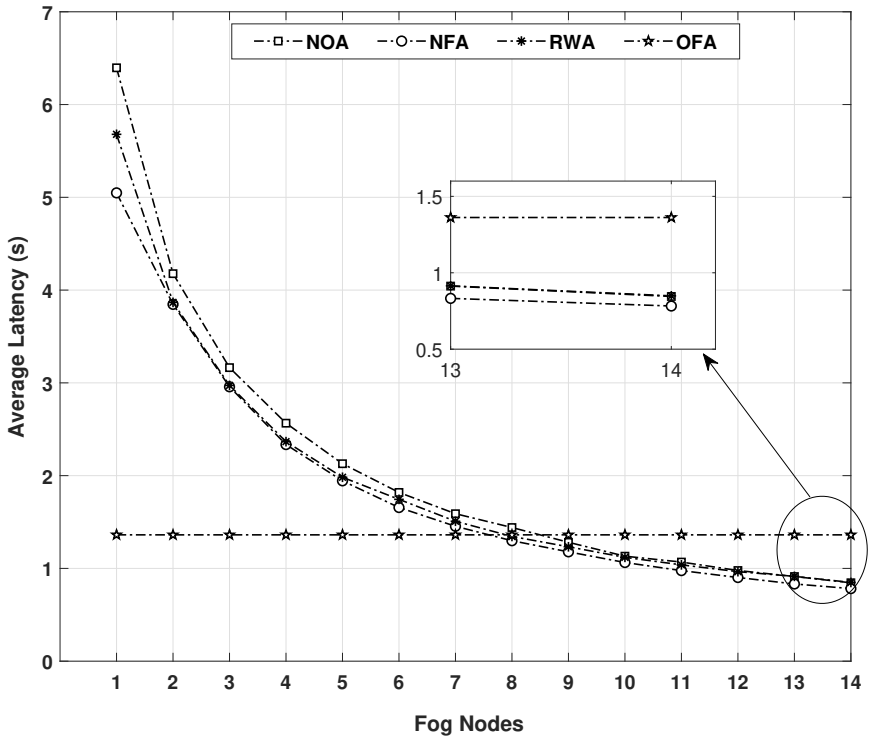


Figure 5.8: Average latency per node

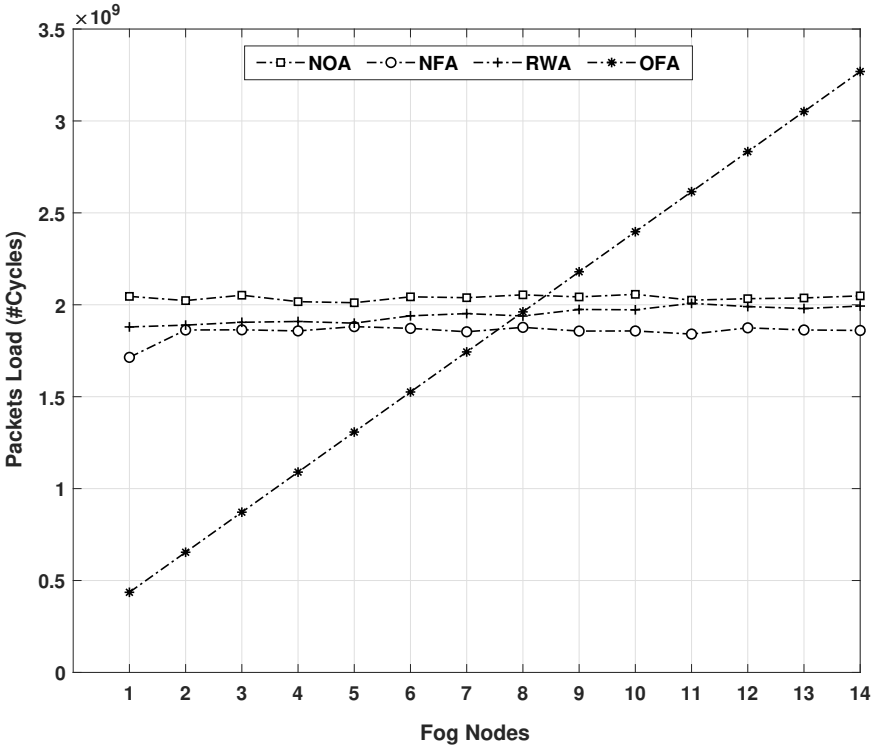
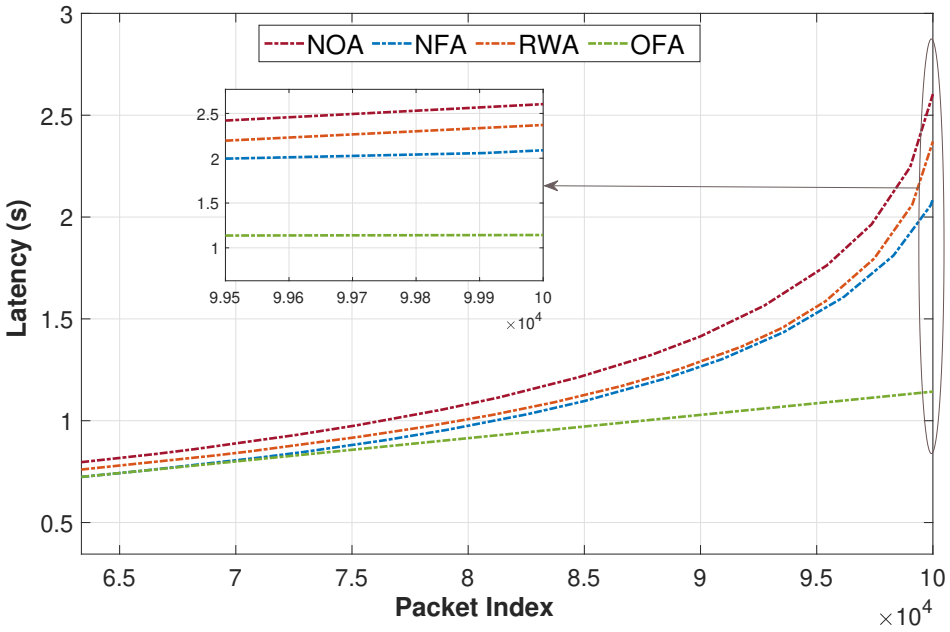
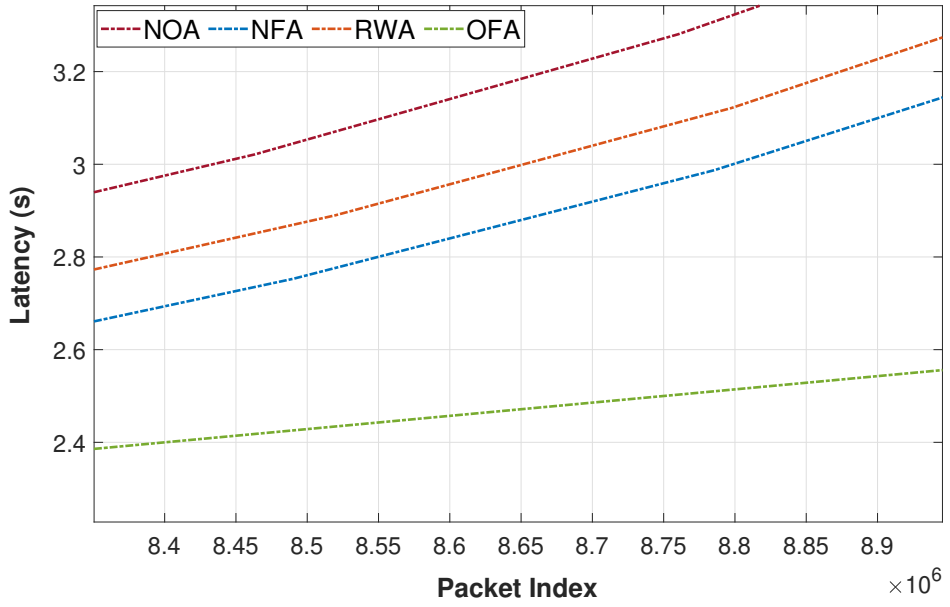


Figure 5.9: Average load on nodes



(a) Packet index upto 10^4



(b) Packet index upto 10^6 to show the variations

Figure 5.10: Latency per packet

the issue of latency as, on the one hand, fog nodes with small CPU frequency consume significant time to process all received packets, while, on the other hand, fog nodes with large CPU frequency have already finished processing the received packets as per the results in Figure 5.8.

Figure 5.10 shows the impact of increasing the number of packets on latency. During simulation, service's packets are varied from one packet to 10×10^4 packets in Figure 5.10a; thus, the packet type is fixed to heavy-packet for consistency. The service utilisation rate is an incremental parameter from 1% to 100%, thus, this rate is fixed at any given timestamps, for example, if the service utilisation rate is 50%, all algorithms; OFA, NAF, RWA, and NOA will receive the same rate. It is obvious that increasing the number of arrived packets (i.e., increase the service arrival rate λ) will increase the overall latency. The total latency and performance of the algorithms vary; OFA has the lowest service latency as per Figure 5.10a. The service latency is stable with small delay of approximately 0.6 second for the received packets upto 6.5×10^4, thereafter, the latency start to increase significantly for NOA, RWA, and NAF. While, OFA remains stable with less than 1.2 second latency for all received packets and upto 10×10^4 packet. Moreover, in Figure 5.10b, we have increased the packets utilisation to 10×10^6 to show the continuous latency variations for the different algorithms compared to OFA. It is clear that OFA has a sustainable packets processing with the increase in service packets (i.e., high traffic), in terms of latency, as it has the lowest packet latencies.

Moreover, in the new experiment, we increase the packet arrival rate λ to 15×10^4 to monitor how the offloading performance and service latency will be effected. Latency will be increased for all offloading algorithms. However, the incremental rate will matter as this will reflect the sustainability of the offloading algorithm. Figure 5.11 shows the maximum and average latencies for the 15×10^4 packets (with type heavy) based on the offloading algorithms. In comparison between the maximum latencies for all offloading algorithms in Figure 5.10 and 5.11, it is clear that the increment of maximum latency for NFA, NOA, and RWA is significantly more than the maximum latency of OFA, as in Figure 5.10 the maximum latency for a packet with OFA is around 1.2 second, and in Figure 5.11 the maximum latency is 0.8 second. Whereas, within NFA and RWA the maximum latency is 2.1 and 2.8 seconds, respectively, in Figure 5.10, while the maximum latency is 2.7 and 3.2 seconds, respectively, in Figure 5.11. It is clear that OFA outperform NFA and RWA in either cases in terms of achieving faster response time.

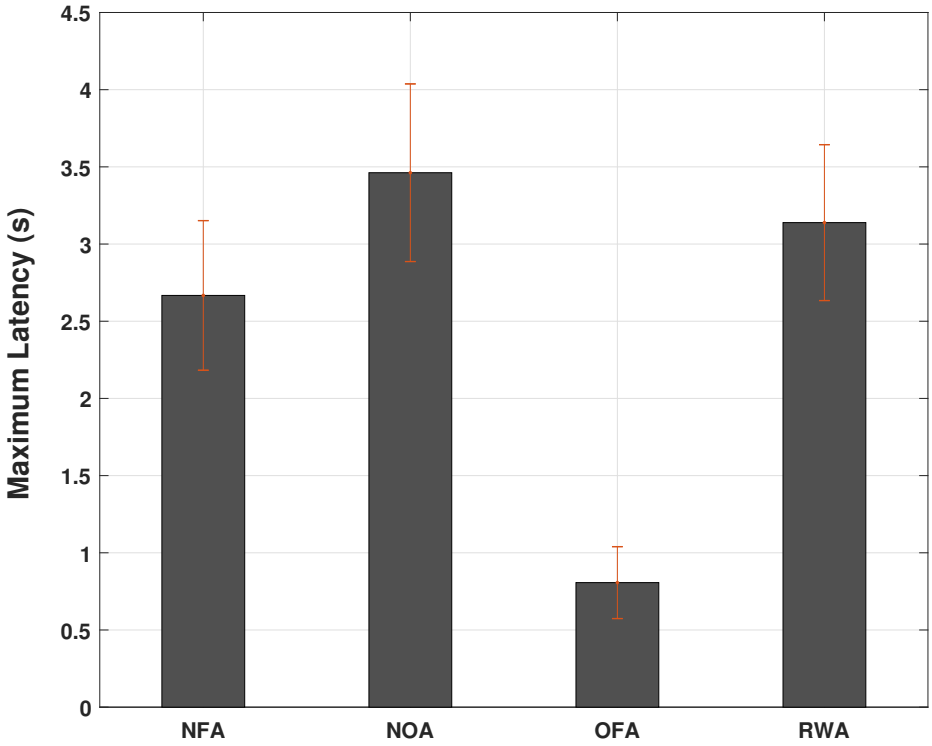


Figure 5.11: Maximum latency upon heavy-packets

5.5 Chapter Summary

This chapter focuses on the practicality and management of fog computing. Although fog computing is recognized as a computing model that suits IoT systems/applications, it is still not widely used due to the spatial and temporal dynamics of IoT thing's distribution that makes the management and distribution of fog nodes difficult. Also, this could make the computations loads on fogs vary significantly. Therefore, some fog nodes could be lightly loaded, while others are not, causing fog congestion hence latency.

In this chapter, a novel Fog Resource manAgeMEnt Scheme (FRAMES) has been proposed to crystallise fog distribution and management with an appropriate service load distribution and allocation. Also, service load reallocation via service request(s) offloading among participated fog nodes within one domain to: i) achieve minimal latency for IoT services, and ii) allocate minimal load on fog nodes. FRAMES proposes/allow $\mathcal{F}og$-2-$\mathcal{F}og$ coordination, which turned out to be a feasible solution that enables fog traffic management

via service request offloading in fog based network architecture that serves the purpose of minimizing the average response time for real-time IoT services. Through the extensive experiments, it is clear that FRAMES and its proposed offloading algorithms significantly impact the overall latency of the IoT services. Thus, the proper resources managements with the accurate offloading decisions, the services response time is significantly improved. Also, the number of fog nodes and their capacities will also impact services delays. This chapter have addressed RO3 as it has investigated the barriers that might impede fog in term of resource managements and the approach to provide fog services, also addressed RO4 in term of the design and develop of a comprehensive solution that manages the fog network resources. Hence this chapter fulfill RQ3 and RQ4.

From the experiment's results, it is clear that the proposed OFA has the lowest service response time in comparison with RWA and NFA. Moreover, OFA has not only outperformed on RWA and NFA in latency, but also in the service packets distribution over fog nodes upon their capabilities. In general, if all fog nodes have low load, offloading is unnecessary, and if all fog nodes have heavy loads, offloading will not help to reduce the delay. Offloading only helps when there is a high degree of variance among the fog nodes. OFA has the potential to achieve a sustainable network paradigm to highlight the significance and benefits of adopting the fog computing paradigm. Having the fog-cloud collaboration model discussed in previous chapter and the $f$og-2-$f$og model discussed in this chapter, the next chapter will look on having a secure IoT environment for node's interactions and resources sharing among fog nodes through the fog COMputIng Trust manageMENT (COMITMENT) model.

CHAPTER 6

Fog Computing Trust Management

In questions of science, the authority of a thousand is not worth the humble reasoning of a single individual

Galileo Galilei

6.1 Introduction

Fog computing is generally considered to be more secure than cloud computing for the following reasons: Firstly, the collected data is transiently maintained and analyzed on local fog nodes closest to data sources, which decreases the dependency on the Internet connections. Secondly, local data storage, exchange and analysis potentially make it more difficult for hackers to gain access to user's data, since there can be separate and different security barriers at different fog nodes. This limits the amount of user data that could be accessed in any given data breach compared to a more centralized cloud computing environment. However, fog computing cannot be deemed to be secure, since it still inherits various security risks from cloud computing. In general, the fog nodes and clouds are honest but curious. They are deployed by fog vendors to offer specific services honestly to users for their own benefits. On one hand, for monetary reasons, they may not deviate from the protocols agreed upon among the ones involved, on the other hand, they may snoop on the content of maintained data and the personal information about data owners. Therefore, the fog nodes could be honest-but-curious, even malicious. Further, malicious fog nodes might acquire personal information about users, resulting in the privacy leakage for users. Thus, there exist several challenges for preserving security and privacy in fog computing [103, 23].

In fog computing, fog-based services are generally owned by different parties for various reasons: (*i*) the deployment choice that may include the selection of Internet service providers or wireless carriers, (*ii*) businesses extending their existing cloud-based services to the edge for performance improvement, (*iii*) offering spare resources on the local private cloud as fog services to local businesses on lease [23]. This flexibility of different providers offering different fog-based services complicates the trust situation between fog nodes. Moreover, the devices used by the fog users are often considered resourceful in terms of their capabilities, but they are still incapable of executing certain complex tasks such as those required in applications like image processing, virtual reality, augmented reality and smart transportation [167]. Thus, such tasks are offloaded and user's control over data is handed over to the fog layer where fog nodes may work independently or in the coordination ($\mathcal{F}og$-2-$\mathcal{F}og$ coordinations) on the tasks to achieve the overall objective. Since, the outsourced data can be transferred to a rogue fog node, an adversary can tamper or steal user confidential

data and can easily launch more attacks. A rogue node would be a malicious fog device that appears to be legitimate and coaxes end users to use them, but, in reality, these nodes are malicious in nature. Various cryptographic-based approaches exist that can effectively prevent external attack, but are not useful in case of internal attacks where rogue fog nodes are already part of the application using legitimate identities. We, therefore, resort to trust to "single out" malicious fog nodes and mitigate security risk, respectively. Fog nodes are expected to be collaboratively monitored by their neighboring nodes, based on $\mathcal{F}og$-2-$\mathcal{F}og$ model, for any sign of deviation from acceptable behaviors and predict their reliability for handling future jobs based on past reputation.

Therefore, in this chapter a fog computing trust management approach is proposed. The focus of the proposed approach is to ensure that the fog computing layer can be secure and efficient. Hence, ensure i) Quality of Service (QoS) for fog node to achieve maximum bandwidth and deal with the service requests with minimal latency and low error rate, ii) Quality of Protection (QoP) for fog nodes to protect the received data during processing as well as transferring or sharing the data with other fogs (e.g., service integrity and confidentiality). The major contributions are threefold:

1. Fog COMITMENT: *COMputIng Trust manageMENT* approach to impart useful prognostic information on fogs trustworthiness. Thus, providing a secure and trusted fog computing environment to share node's resources and exchange data securely and efficiently in $\mathcal{F}og$-2-$\mathcal{F}og$ collaboration model.

2. A load balancing algorithm to monitor fog's resources (i.e., CPU consumption), active fog processes (e.g., stakeholder services processes), and the incoming services requests volume onto fog. Thereof, it is able to monitor fog's performance and to promote load balancing via offloading ($\mathcal{F}og$-2-$\mathcal{F}og$) to address the latency concern on fog nodes, thus, triggering the offloading function upon fog congestion. The offloading function and algorithm will be aided with a trustworthiness function to avoid coordination with malicious fog nodes.

3. Trust and Recommendation model with an algorithm that helps fog nodes make the right decision for selecting the appropriate fog node(s) for collaboration during the

offloading process. Hence, this process includes assessing the trustworthiness level of the nominated fog nodes to ensure that the QoP and QoS provided by hosted fogs are meet.

6.2 Fog Computing Trust Management Model

Before we dive into the Fog *COMputIng Trust manageMENT* (COMITMENT) details, it is worth mentioning the network environment adopted for the fog computing. The network topology adopted a distributed-based fog topology where nodes are physically distributed over different locations and connected to each other via a communication protocol forming a mesh networks. Thus every node has a unique identity address (e.g., IP), so the fog nodes are reachable to each other without a central controller to help resource sharing and service requests offloading. In addition, there is no centralised trust authority among fogs to point out the trusted nodes within the network, hence each fog node compute the trust evaluation periodically to its neighbouring fog nodes and stores the generated list of trusted fog nodes locally through COMITMENT. Here are some importance preliminaries that are required for COMITMENT.

- Fog Quality of Service (QoS): we refer to fog QoS as the ability of fog to achieve maximum bandwidth (associated with the time to upload and download a packet $\tau_{\downarrow\uparrow}$) and deal with the service requests with minimal latency and low error rate. The problem preliminaries associated with QoS are the fog's workload (f_x), service workload on fog ($s_w^{f_i}$) and the total time required to process a service (τ_s).

- Fog Quality of Protection (QoP): we refer to fog QoP as the degree to which the fog protects the received data during processing as well as transferring or sharing the data with other fogs. The QoP properties (e.g., service integrity and confidentiality) are defined according to the type of processes and services provided by the fog. QoP problem preliminaries are associated with the proposed trustworthiness model and based on direct trust ($\tau_{a,b}^d$) and recommendation/indirect trust ($\tau_{a,b}^r$).

- Fog Secure Service Level Agreement (SSLA): this refers to the commitment between two fogs in delivering a service according to a certain level of quality, availability and

protection. Thus, SSLA includes the problem preliminaries associated with both QoS and QoP. Thus, service requirements of both QoS and QoP should be provided by the collaborating fog nodes.

- Fog Requirements of Protection (RoP): is a set of security requirements which includes the security factors required for delivering the desired services, thus RoP defines and measures the QoP for a fog node.

- Level of Trust (LoT): is a score that refers to the trustworthiness among fogs. LoT is computed based on the previous coordinations experiences, and is periodically updated after each coordination. The problem preliminaries associated with LoT are the experience satisfaction score $ES_{a,b}$, the α and β which logs the satisfied and unsatisfied experience, respectively. LoT indicates the level of trust or distrust between the fogs, therefore, the LoT score used is based on a fuzzy logic where the score *one* is an indicator of absolute trust and the score *zero* is an indicator of absolute distrust.

COMITMENT is a software-based approach that is installed on each fog node within the fog layer. COMITMENT is responsible for providing a secure and trusted environment for fog nodes to share their resources and exchange data packets, COMITMENT architecture is shown in Figure 6.1. Thus, COMITMENT provides a concise decision for the fog node to point out the best fog node that it can cooperate with, more precisely the fog node that provide lowest latency (for best QoS and QoE) and secure data sharing and processing (for best QoP) during the fog-2-fog coordination. The decision not only includes the best fog node that can handle the jobs efficiently but also the most trusted fog nodes that could offer/provide best QoS (e.g., low latency) and best QoP (e.g., meeting the SSLA). The offloading model is to balance the workload and service traffic within the fog layer by distributing service requests from the congested fog node to another fog node in a secure manner. In order to enable COMITMENT to select the trusted node, it essential to assess both the QoP and QoS provided by the fog nodes that are possibly hosting the service delivery. This can be achieved through checking the trust level of each fog node. The trust level is evaluated based on; i) a *direct experiences* which is based on direct interaction among the collaborating fog nodes in the past and/or present interactions along with different interactions experiences, ii) an *indirect experiences* which is based on *recommendations*

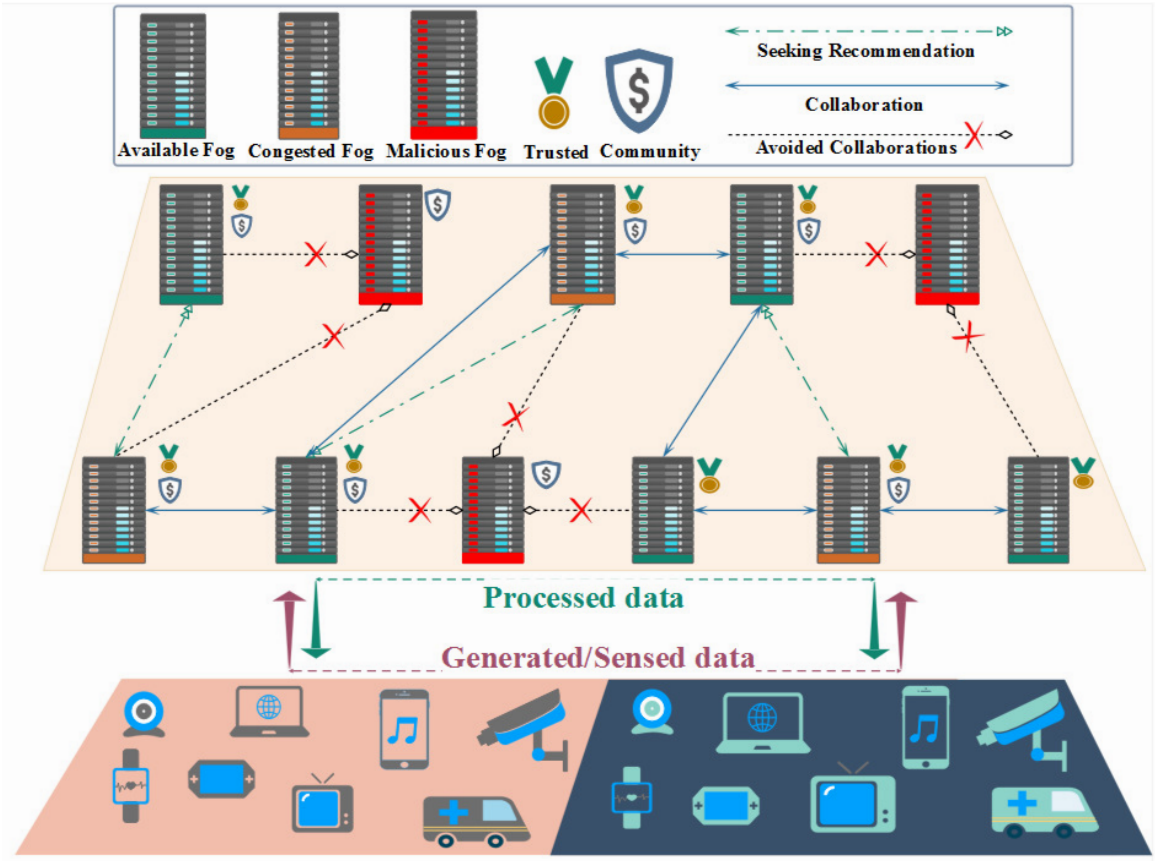


Figure 6.1: Architecture of the proposed COMITMENT approach, including the different types of fog's statuses and interactions

from neighbouring fog nodes in case of no previous experiences between the two fog nodes that are intended to collaborate with each other.

Obviously, the trust level will be computed based on the previous coordinations satisfactions, hence the self experiences obtained from direct interactions will always have a higher weight than recommendations from neighbouring fogs because the trustworthiness among fogs is subjective and asymmetric as per fog security requirements in Section 3.3. The most used notations in this chapter are given in Table 6.1. The main procedures and processes run by COMITMENT are categorised as follows:

1. Fog performance: COMITMENT periodically monitors fog's resources (e.g., CPU consumption), active processes (e.g., stakeholder's services processes), and the incoming services requests traffic on the fog node. This process is to monitor fog per-

formance to identify resource exhausted services that potentially can be an attack. COMITMENT will trigger the load balancing function via offloading upon fog's overload detection, thus a trustworthy fog node can be called upon to handle the overload. This process is discussed further in Section 6.3.

2. Fog interactions: upon overload detection, COMITMENT has the responsibility to handle the process of finding the best neighbouring nodes that can handle the overload securely and efficiently. This process includes assessing the trust level of the nominated fog nodes for sharing the overload. This process ensures that the QoP and QoS provided by the hosted fog node meets the SSLA standard of the service and user expectations about the desired service, for example, service run with no delay and assured data protection. This process is discussed further in Section 6.4. It worth noting that fog node's failures during or before a cooperation is establishing is handled by FRAMES, as discussed in previous chapter (Section 5.2), the fog pinger utility will notify FRAMES upon a fog node being unresponsive (i.e., failure) and unable to handle processes, hence other fog nodes in the domain will be notified to act accordingly with both data feeds from FRAMES and COMITMENT.

Table 6.1: Notations used in the paper

Symbol	Description
t, n, T	thing, index of t, set of things
f, i, F	fog, index of f, set of fogs
λ	service arrival rate to fog layer
μ	fog node service rate
S, s	set of services, one service
s_w	service workload
$s_w^{f_i}$	service workload for fog node (f_i)
s_d	service deadline
τ_s	total time required to process a service
t_s	service's tasks
rs	fog node resources
τ_{que}	is the queuing time
τ_{pro}	service processing time
ρ	system usage
τ_{que}^{si}	queuing time for s at the resources of fog f_i
f_c	fog capacity
f_w	fog workload
f_i^c	processing capacity of the fog node f_i
f_{rs}^s	total fog resources (rs) allocated to processes service (s)
D_p	propagation delay
$\tau_{\downarrow\uparrow}$	time to upload and download a packet
α_{f_a,f_b}	logs the satisfied experience from fog_a to fog_b
β_{f_a,f_b}	logs the unsatisfied experience from fog_a to fog_b
$ES_{a,b}$	experience satisfaction from fog_a to fog_b
n_{int}	number of direct interactions between the two fogs
r_{f_a,f_b}	recommendation of fog_a toward fog_b
$LoT(f_a, f_b)$	level of trust score of fog_a toward fog_b
$C_{ts}^{f_i}$	total CPU (in hertz), consumed by a t_s on fog node f_i
$\tau_{a,b}^d$	direct trust of f_a toward f_b
$\tau_{a,b}^r$	indirect trust of f_a toward f_b (recommendation)

6.3 Fog Performance: Safe Load Balancing

One of COMITMENT's roles is to keep tracking the performance of a fog node so it achieves the best efficiency. The important factor that COMITMENT monitors is fog's workload (f_w), which refers to the overall usage of a fog's CPU that is consumed during the processing of a particular service's request. COMITMENT tries to identify resource exhausted services that can potentially affect the performance of the fog node or if it been attacked, therefore an action must be taken, such as terminating the running services and/or offloading them safely to other fog nodes.

As mentioned before, the total CPU used by the running services should not exceed the allocated fog node resources for a service. The total resources allocated to process a service are based on the type of service packets (heavy-packets and low-packets) and the current load of the fog. Equation 6.1 (recall from Section 5.3) computes the total resources (rs) allocated to process all tasks (ts) for a service (s).

$$f_{rs}^{s} = s_w = \sum_{t=1}^{n} C_{ts}^{f_i}, \lceil s \rceil \leq f_c, \forall s \in S, \forall t \in T_s \tag{6.1}$$

The total fog's workload capacity (f_c) depends on the actual hardware specification of the allocated device. The assignment variable s_w (i.e., total service workload) is set so that total service processing workload does not exceed f_c, as per Equation 6.1, where $C_{ts}^{f_i}$ denotes the total resource (CPU consumption in hertz, having *hertz=cycles/second*) consumed by a service's tasks on fog node f_i. COMITMENT's main fog performance constraints and safe offloading algorithm are discussed in the following subsections.

6.3.1 Problem Formulation and Constraints

COMITMENT's main focus is to deliver the IoT services with best QoS and QoP. Hence, COMITMENT aims at delivering IoT services securely with minimal delay. The total latency for a service's request sent from t_n to f_i is computed by adding the time of uploading a service's packets ($\tau_{\uparrow}$) to the waiting time in the fog node queue (τ_{que}) until it gets processed. The delay for processing the service (τ_{pro}) and the time to respond back ($\tau_{\downarrow}$) to t_n is also

added with the total latency for the service as per Equation 6.2 (recall from Section 5.3).

$$\tau_s = \tau_{\uparrow} + \tau^s_{que} + \tau_{pro} + \tau_{\downarrow}, \forall s \in S$$

$$\tau_s = \tau^s_{que} + \tau_{pro} + 2\tau_{\downarrow\uparrow}, \forall s \in S \tag{6.2}$$

COMITMENT brings a new constraint to the problem formulation of fog computing in Equation 6.3. The new constraint is associated with the Secure Service Level Agreement (SSLA). Moreover, since COMITMENT is keen to ensure the QoP, it use SSLA as one of the main constraint to ensure the quality of delivery of user desired services. Therefore, the problem can be formulated as in Equation 6.3, where low latency includes delivering the services before the deadline (s_d) with the desired QoS, also the SSLA should be met according to the service's requirements of protection (QoP).

$$P: \quad max[\tau_s] \leqslant s_d, \forall s \in S \tag{6.3}$$

$$\text{s.t.} \quad f_c^{min} \leqslant f_w \leqslant f_c^{max} \tag{6.4}$$

$$\sum \lambda_s \leqslant \sum \mu_f \tag{6.5}$$

$$f_{QoP} \geqslant S_{RoP} \tag{6.6}$$

$$\lambda_s \xrightarrow{min[D_p]} f_i \tag{6.7}$$

$$\tau_s \leqslant s_d, \forall s \in S \tag{6.8}$$

The constraints main focuses are on the QoS and QoP, Therefore, they can be written as follows; Constraint (6.4), indicates that (f_w) is strictly bound by an upper limit (f_c^{max}) and lower limit (f_c^{min}) which is related to fog capabilities based on CPU frequency (unit $hertz$). Constraint (6.5) imposes that the total traffic arrival rate (λ_s) to a fog domain should not exceed the service rate (μ_f) of that specific fog domain. Constraint (6.6) indicates that the QoP provided by fog should be either equal to or greater (i.e., better) than the RoP of the desired service. Constraint (6.7) imposes that the first destination for the IoT thing node's packets generated will be to a fog node with minimal cost of propagation delay within the

fog domain. Ideally, lowest propagation delay is for the nearest fog node. Finally, constraint (6.8) strictly binds the service time τ_s within the limit of service deadline s_d.

6.3.2 Safe Offloading Model

The decision factors where a node is congested and offloading is required rely significantly on fog workload (f_w) and SSLA. The f_w is mainly associated with the service traffic arrival rate (λ_s), total service rate (μ) and the fog node's capability(i.e., CPU frequency), while the SSLA is mainly associated with the service protection and quality. Therefore, the offloading decision made by a fog node is dependent on Probability 6.9, where i) the service delivery time by fog (f_{τ_s}) is greater than the service deadline ($\tau_s > s_d$), ii) the fog node can not provide the required SSLA for a service's request, received from an end-user, in other words, fog QoP (f_{QoP}) is less than the service requirement of protection (S_{RoP}).

$$O_s = \begin{cases} 1, & \text{if } f_{\tau_s} > S_d \\ 1, & \text{if } f_{QoP} < S_{RoP} \\ 0, & \text{otherwise} \end{cases} \tag{6.9}$$

Probability 6.9 is the decision maker for the COMITMENT model to either allow the fog to process the service request or offload the service requests to another fog node. In Probability 6.9, O_s value is set to either 0 or 1, where 0 refers to no offload is required and 1 refers to offloading is required. When the fog node makes the decision for offloading, it has to find the alternative fog node to deliver the offloaded service request. Thus, having a workload limit and both QoS and QoP constraints on the fog node that participates in handling the overload and/or delivering the service. Hence, for this process, the fog node has to do the following:

1. compute the service time τ_s for the service requires offloading among all available fog nodes using Equation 6.10.

$$min[\tau_s] = \sum_{i=1}^{n} [\tau^i_{que} + \tau^i_{pro} + \tau_{\downarrow}] \tag{6.10}$$

$$\text{s.t.} \quad \tau_s \leqslant s_d, \forall s \in S$$

$$f_{QoP} \geqslant S_{RoP}$$

2. check the trustworthiness of the fog nodes in handling the service processing according to the desired SSLA, more details on finding fog's trustworthiness in Section 6.4.

Algorithm 3 is developed to find the best available fogs to aid the congested fog node and provide the best service SSLA, then offload the service request to the identified fog node. The first part of the algorithm, Procedure 1, shows the process of finding the best available fog node(s) for coordination. Lines 2-3 of the algorithm initiate the list of active fog nodes in the domain alongside the fog's capacity and current load (i.e., queue size). The list of available nodes will be refined by removing the fog nodes that are already busy processing other services (i.e., $\lambda_i = \mu_i$) as per lines 6-8, or their QoP is not enough to meet the service's RoP as per lines 9-11. The remaining part of Procedure 1, lines 12-22 will compute the time required for the service to run on each of the available fog nodes. If the time is within the limit allowed for the service (i.e, before S_d), the system will keep the node in the list and log the expected service time against the fog node as per lines 13-15. If the τ_s on F_n is greater than S_d, then F_n will be removed from list as per lines 16-18. The second part of the algorithm, Procedure 2, receives the list of best available fog nodes that are able to meet the service's QoS and QoP. Hence, the fog node can deliver service with no latency providing adequate SSLA that meets the service's RoP. If the list is not empty, this means there is at least one fog node able to process the service. However, if there is more than one fog node in the list, the system will direct the overload to a node that can provide; minimal τ_s, best QoP and has the lowest propagation delay D_p as per lines 21-23.

Algorithm 3: SERVICE OFFLOADING

Input: FogNode (F_n); FogLoad (F_l); OverLoad (O_l); RoP (S_{RoP}).
Parameters : FogCapacity (F_c); Propagation (D_p); FogQoP (f_{QoP})
Initialisation $F_n = \phi$; $F_c = \phi$; $F_l = \phi$; $O_l = \phi$.
⋮
Result: Share the Overload with best available node

Procedure 1. *Determine best available node* **by**
  $F_L = list\{\phi\}$ ▷ F_L `initiate fog list`
  $F_L = list[F_n] \longleftarrow getFogNodes(out : (F_n, F_c))$
  $F_L = sort(F_L,$ by F_c DESC$)$
  for *each* $F_n \in F_L$ **do**
    if $Fn \longleftarrow (F_l \geq Fc_{max})$ **then**
      $F_L = pop(Fn)$ ▷ `remove busy node`
    else
      if $Fn \longleftarrow (S_{RoP} \geq f_{QoP})$ **then**
        $F_L = pop(Fn)$ ▷ `remove busy node`
      else
        $\tau_s = \sum_{i=1}^{n}[\tau_{que}^{i} + \tau_{pro}^{i} + \tau_{|}]$
        if $(\tau_s < s_d)$ **then**
          $list.add(F_n, \tau_s)$
          continue
        else
          $F_L = pop(Fn)$
        end
      end
    end
  end
  return F_L
End
Procedure 2. *Handover the Overload* **by**
  if $F_L \neq \phi$ **then**
    $F_n = min[F_L(\tau_s, D_p)])$ && $best[F_L(QoP)]$
    $F_l^n = F_l + O_l$
  else
    goto:1
  end
End

6.4 Fog interactions: Trust and Recommendation

This section will propose a model that helps fog nodes to make a right decision for selecting the appropriate fog node to collaborate in delivering the desired service to the end-user. Generally, in any network architecture there will be two types of fog nodes, *Trusted* fog nodes and *Malicious* fog nodes. *Malicious* fog nodes are defined as fogs that seek to breach user privacy or any security principles, hence these fog nodes are under attack that affects the fog's performance and efficiency. Such malicious fog nodes exhibit behaviour such as i) packets drop with bandwidth consumption so that no other legitimate fog node can use them, ii) stale packets are injected into the network to congest the network and cause confusion other fog nodes, and iii) purposely delay services and dispose user's data and breach their privacy [168]. While the *Trusted* fog node is defined as fogs which are working with full capacity to satisfy users and services requirements, thus providing highest QoS and QoP possible. These features make the trusted fog nodes vulnerable, hence they are exposed to attacks by malicious fog nodes. In the following subsections we will propose a trust and recommendation model to help *trusted* fog nodes to identify *malicious* fog nodes and avoid dealing or collaborating with them. The trust model assists fogs to find other fog nodes trustworthiness based on previous direct experience, while the recommendation model assists fogs to find other fog nodes trustworthiness based on collecting trustworthiness recommendations from neighbouring fog nodes, when current fog nodes have no previous interactions to rely on, hence they seek recommendations.

6.4.1 Trust - Direct Experiences

In the fog-2-fog coordination model, the direct communication between the fog nodes is evaluated based on the quality-of-service (QoS) and quality-of-protection (QoP) for the services provided by both collaborating fog nodes, thus, each fog node scores the coordination experiences against the partner fog node in terms of both QoS and QoP. The coordination experiences score is logged locally by each fog node after every interaction, this to be used in the future to predict/assist the coordination success and trustworthiness between each other in the future interactions. This can be seen as a direct experience as both fog nodes can evaluate each other based on their own experiences and not based on recommendation

from other fog nodes, thus, this evaluation helps a fog to determine the LoT against its partner fog node.

Moreover, the history of past interactions between fog nodes is essential to assess node's trustworthiness. Obviously, from the past direct interactions, the fog nodes that have a positive history should have a positive/good impact on the LoT score. While the nodes that have a negative history should have a negative/bad impact on the LoT score. Therefore, in the proposed model, it is essential for each fog node in the fog domain to log the score of its Experience Satisfaction (ES) during the direct interactions with other fog nodes. The ES score is a binary value, hence can be either *zero* or *one*, where *one* is indication of trust/satisfied and *zero* is indication of distrust/unsatisfied. Thus, the ES score will be given upon meeting the QoS (e.g., low latency) and QoP (e.g., data protection). Bayesian network is adopted to evaluate the direct satisfaction experiences based on direct interactions between fog nodes. Bayesian has been adopted because it has proven results with peer-2-peer network modelling in terms of trust/reputation and in line with [35, 169]. The satisfaction experience parameter of f_a toward f_b is represented by ES score $ES_{a,b}$. The value of ES is a binary value, either it is set to 1 for satisfied experience or to 0 for unsatisfied experience during the interactions.

The ES is distributed between satisfied and unsatisfied experiences (i.e., distributing of 1s and 0s) according to Bernoulli trial distribution, thus, referring to the probability of satisfied experience by a positive experience parameter $p_{a,b}$ according to Beta distribution, hence, the posterior $Pr(p_{a,b}|S_{a,b})$. The direct trust $\tau^d_{a,b}$ of f_a toward f_b is computed as per Equation 6.11.

$$\tau^d_{a,b} = \frac{\alpha_{f_a,f_b}}{\alpha_{f_a,f_b} + \beta_{f_a,f_b}} \in [0-1] \tag{6.11}$$

Where the α_{f_a,f_b} and β_{f_a,f_b} refer to the parameters of Beta distribution, thus, α_{f_a,f_b} logs the satisfied experience, while β_{f_a,f_b} logs the unsatisfied experience. Both α_{f_a,f_b} and β_{f_a,f_b} are computed and updated after every direct interaction between f_a and f_b with a consideration for the trust decay as per Equations 6.12 and 6.13.

$$\alpha'_{f_a,f_b} = e^{d\Delta t}.\alpha_{f_a,f_b} + ES_{a,b} \tag{6.12}$$

$$\beta'_{f_a,f_b} = e^{d\Delta t}.\beta_{f_a,f_b} + 1 - ES_{a,b} \tag{6.13}$$

Where α'_{f_a,f_b} and β'_{f_a,f_b} refers to the new scores, while α_{f_a,f_b} and β_{f_a,f_b} refers to the old scores. The $e^{d\Delta t}$ refers to the exponential decay, thus, d is the decay factor and the Δt is the trust update interval. It worth noting that d is a small value to represent the trust decay over the execution/run time.

In order to make the trusted network reliable and scalable, the fog node should not burden its resources with redundant trust scores and only logs the most recent ES score against the fog node (e.g., $f_a \Longleftrightarrow ES$) along with the number of interactions between the two fog nodes. Therefore, the ES score is an accumulative score and it is periodically updated and logged in an ES_{score} as a mapping function as per Equation 6.14. Where $f_a \longrightarrow f_b$ maps the interaction from fog_a to fog_b and n_{int} refers to the number of direct interactions between the two fog nodes.

$$ES_{score}(a,b) =< f_a \longrightarrow f_b, n(int), \alpha_{f_a,f_b}, \beta_{f_a,f_b}, LoT > \tag{6.14}$$

It is worth noting that in previous research the initial value of α and β is set to *null* or 1 as there is no previous knowledge and no prior interactions between the two fog nodes. However, we adopt a recommendation based approach to obtain the initial value of α and β through seeking a recommendation from a neighbouring node(s) that has the same QoP, this is discussed in Section 6.4.2. If no initial value can be obtained from either the direct experience or the recommendations, then the initial value of α and β is set to 1 since no prior knowledge is available and in line with [35].

6.4.2 Recommendations - Indirect Experiences

In this research, we refer to the recommendations as an indirect trust experience as a fog node can not evaluate its partner's trustworthiness directly based on its own experiences as there is no prior knowledge (i.e., no direct interactions in the past), instead it makes the trustworthiness evaluation based on recommendations from neighbouring fog nodes. In the recommendations model we adopt the design concept of distributed Collaborating Filtering (CF) [170, 35] to obtain a trustworthiness score from neighbouring fog nodes that share similar interests [35]. Therefore, CF classifies the received recommendations based on recommender party into two types:-

- Recommendations from *trusted fog nodes*: this includes recommendations provided from a trusted fog node based on our trust model in Section 6.4.1. The recommenders of this type of recommendations are evaluated based on their LoT from past interactions with the desired fog node. Thus, they should have a satisfactory experience score obtained from positive/secure past interactions prior to making a recommendation. With this type of recommender its sufficient to check the LoT without checking the QoP and SSLA (service specific) as it should be already met, prior to previous interactions. The fog nodes that are seeking recommendation from this type of recommenders are likely to have a general (i.e., non subjective) trust score toward the desired recommender fog nodes.

- Recommendations from *community fog nodes*: these recommendations are provided from fog nodes that have the same service interests as the desired fog node. It is not necessarily for the recommender of this type of recommendations to have a LoT or previous interactions. However, the recommender should share the same service's interests with regard to the QoP and SSLA toward the desired IoT services and the one provided by the desired fog node. Hence, in such case, fog nodes that have the same sentiment towards the desired fog node. The fog nodes that are seeking recommendation from this type of recommenders are likely to have a similar subjective trust score toward the desired recommender fog nodes.

It is worth noting that in order to consider the recommendations provided from the two type of recommenders, *trusted fog nodes* and *community fog nodes*, we first evaluate the relationship between the trustor fog and the recommender fog node to avoid intruder/malicious neighbouring fog nodes. Evaluating the relationship will be based on the type of the recommender, if a trusted fog node has a satisfactory LoT score, then we can consider the recommendation, otherwise, ignoring the recommendation. Whereas, if the recommendation is from a community fog node, we first check if the recommender fog meets the QoP, thus meeting all the SSLA's requirements of protection (RoP shared by trusty fog node) before we can consider its recommendation. Hence, recommendation will only be considered if the recommenders have similar SSLA's RoP standards (i.e., providing the same QoS and QoP experiences). Moreover, the trustor fog nodes will weigh the recommendations provided by the recommenders fog nodes toward the trustee fog node to get the overall trustworthiness as per Equations 6.15.

$$r(a,b) = \sum_{rp \in R} [w_{rp} \times r_{f_a,f_b}], R \in [m,c] \tag{6.15}$$

Where w_m and w_c is the weight of recommendations obtained from trusted fog nodes and community fog nodes, respectively. Thus, total recommendation value/score obtained from the recommenders is 1, thus, $w_m + w_c = 1$, having the value of w_m and w_c is a number between 0 and 1 (i.e., $0 \leq w_m, w_c \leq 1$). The r_{f_a,f_b} denotes the recommendation of fog_a toward fog_b. Each fog node can send a recommendation request to its neighbouring fog nodes and upon receiving the response (recommendation score), the fog weight the recommendations from all recommenders and calculates the over all indirect trust using Equations 6.16.

$$\tau^r_{a,b} = \frac{r_{f_a,f_b}}{\sum_{i=0}^{nr} r_{f_a,f_b}(a,b)} \tag{6.16}$$

Since the outcome of the trust score $\tau^r_{a,b}$ that has been obtained from the recommendations is a value between 0 to 1, therefore, we apply the fuzzy logic function to determine the level of trust as per the fuzzy logic determination in Figure 6.2, where 1 is indicator of absolute trust and 0 is indicator of utter distrust.

Algorithm 4: Proposed Recommendation Model

Input: $FogNode_a$ (f_a); $FogNode_b$ (f_b); $SSLA$
Result: LoT from neighbouring fogs ($\tau^r_{a,b}$) for f_a towards f_b
Initial : $\tau^r_{a,b} = \phi$; $F_L = list\{\phi\}$
Params : $trustScore$ ($\tau^r_{a,b}$); $FogList$ (F_L); $recommend$ (r)

Procedure 1: *get trusted fog for recommendation* **by**
  $F_L = list[F_n] \longleftarrow getNeighbourFogs(out : (F_n, LoT))$;
  $F_L = sort(F_L,$ by LoT desc) ;
  for *each* $F_n \in F_L$ **do**
    if $F_n \longrightarrow$ *untrusted by* F_a **then**
      $F_L = pop(F_n)$; ▷ `remove untrusted node`
    else
      $F_n = m_r\{f_b, SSLA\}$;
      $F_L = update(F_n, r, out : F_L)$; ▷ `update list adding r`
    end
  end
  return F_L ;
End

Procedure 2: *Compute trustworthiness* **by**
  $F_L = list[F_n, r]$; ▷ `the new fog list with r`
  $F_L = sort(F_L,$ by LoT desc) ;
  for *each* $F_n \in F_L$ **do**
    $r(f_n, f_b) = \sum_{rp \in R}[w_{rp} \times r_{f_n, f_b}]$, $R \in [m, c]$
  end
  $\tau^r_{a,b} = \frac{r(a,b)}{\sum_{i=0}^{nr} r(a,b)}$; ▷ `compute the overall trustworthiness`
  return $\tau^r_{a,b}$;
End

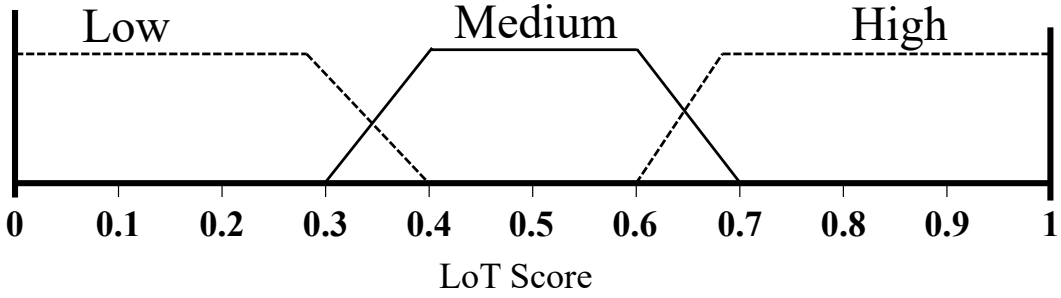


Figure 6.2: Level of Trust (LOT) according to fuzzy logic

Algorithm 4 elaborates the process of seeking a recommendations from a neighbouring fog node. Considering a scenario where fog f_a wishes to interact with fog f_b and it has no previous interactions history, f_a go through the Procedures 1 and 2.

Procedures 1: f_a will try to seek recommendations from neighbouring fog nodes to get the trustworthiness of f_b, so for this, f_a asks fog nodes f_c, f_d, f_e, for example, for recommendations on the trustworthiness of f_b. The recommendation requests are sent only to trusted fog nodes, i.e., trusted by f_a as per lines 3-6. The recommendation messages request will be sent to the trusted fog nodes in the format of $m_r = \{f_b, SSLA\}$ as per lines 7-10, where the first part, in this case f_b, is the desired fog node for checking its trustworthiness, while the other part, the SSLA, is the Secure Service Level Agreement, which is a set of requirements to be used in the evaluation of the trust score of f_b by the recommender. It is worth noting that the SSLA parameters are set according to f_a QoP that is based on the RoP parameters for a specific service. The recommenders, i.e., $f_c, f_d, f_e..f_n$ fog nodes, will evaluate the trustworthiness toward the desired fog (i.e. f_b) based on the SSLA requirements from past interactions experiences, using the proposed trust model in Section 6.4.1 to compute LoT. Then, the trust score is returned to the trustee fog node as per line 12.

In Procedures 2:, the f_a estimates the trustworthiness of f_b according to the gained recommendations, thus, the fog f_a will decide whether f_b is trusty and can deliver the

desired service or not. Hence, the trustworthiness estimation will be computed using Equations 6.16 after filtering the recommendation by the weight of the recommender according to Equations 6.15, as per lines 14-22.

6.4.3 Reputation Assessment

The reputation assessment process will provide the output of the final LoT score which will be used to identify the trustworthiness of a particular fog. In this process, both trust (i.e., direct experiences) and recommendations (i.e., indirect experiences) will be involved to get the LoT score. However, the trust score and recommendations score will be considered in different weights, scores from direct experiences will always have a higher weight, this is due to the level of satisfactory/unsatisfactory experience gained from direct interactions in previous coordinations. Hence, the score of recommendations will be only considered with a higher weight when there are not enough direct interactions between the two fog nodes. The LoT function $LoT(f_a, f_b)$ in Equation 6.17 computes the LoT score which will be used by the desired fog node to make the final decision whether to collaborate or not with the candidate fog node.

$$LoT(f_a, f_b) = \begin{bmatrix} \gamma \\ \delta \end{bmatrix} [\tau^r_{a,b}, \tau^d_{a,b}] = \gamma.\tau^r_{a,b} + \delta.\tau^d_{a,b} \tag{6.17}$$

where δ and γ represent the corresponding weights of the direct (τ^d) and indirect (τ^r) trust score respectively. The score of LoT will be an indication of the level of trust (or distrust) between two fog nodes. For example, the LoT score provided by the function $LoT(f_a, f_b) \in [0-1]$ refers to the LoT score of f_a trust (or distrust) toward f_b according to the previous direct/indirect experiences with f_b. The LoT score will be used based on a fuzzy logic as in Figure 6.2. The fuzzy logic function classifies the LoT score into three main parts, Low, Medium and High to represent the trustworthiness between the two fog nodes. Hence the score of 1 is the indicator of absolute *trust*, while the score of 0 is the indicator of utter distrust. It worth noting that the LoT score, computed using Equation 6.17 is asymmetric and not transitive, hence, each fog node has it's own LoT score that defines fog's QoP. To explain more, LoT score is asymmetric means if f_a finds f_b is trustworthy

based on f_a LoT score towards fog_b, it is not necessarily that f_b finds f_a is trustworthy. Similarly, the LoT score is not transitive means if fog_a trust fog_b and fog_b trust fog_c, it is not necessarily true that fog_a trusts fog_c.

6.5 System Evaluation

In this section, we evaluate the proposed COMITMENT model for a secure $\mathcal{F}og$-2-$\mathcal{F}og$ coordination, which aims at providing secure offloading for fog node service requests. The proposed COMITMENT model has been simulated using MATLAB (2018b) on a Lenovo ideaPad with Intel Core $i5$ processor and 8GB of RAM. Simulation settings are presented in the following subsection (Section 6.5.1), followed by the results and discussion (Section 6.5.2) on the COMITMENT simulation results.

6.5.1 Experiment Configurations

The system characteristics adopted during the simulations are presented in Table 6.2. We specify the simulation settings in terms of network topology, propagation and transmission delay, link bandwidth and fogs capabilities.

Table 6.2: Simulation Settings

Parameter	Value
Operating system	Win 8.1
Simulation environment	Matlab 2018b
Number of fog nodes	15
Fog CPU	$[0.2 - 1.5]$GHz
Network topology	mesh
Number of service's requests	10^5
Package Size	$[0.1 - 80]$KB
Bandwidth	up-to 54Mbps

- Network Topology: this has been modelled as an indirect graph, represents fog nodes as a mesh network. The simulation has 15 fog nodes (i.e., $f_n = 15$) connected together through internal communication link. The links between nodes are weighted based on the propagation delay (D_p) among fog nodes, for instance, if D_p between fog_1 and fog_2 is four seconds, then the link will be represented as ($f_1 \underset{\longleftrightarrow}{4} f_2$). It is worth

nothing that the services arriving at the fog layer are assigned to a fog node with the smallest distance (i.e., smallest D_p).

- Network Bandwidth: the link bandwidth depends on the type of service request, hence, *heavy-request* will require more bandwidth then *light-request*. For light-request (e.g., data packets from sensors) the communication bandwidth used has a transmission rate of 250Kbps [171]. While, for the heavy-packets (e.g., data packets from camera) the communication bandwidth used with a transmission rate of 54Mbps [160]. The transmission rate between the fog nodes is expected to be higher $\simeq$ 100Mbps [14].

- Transmission delay: D_t for a packet is obtained from packet size l_p alongside with the associated upload bandwidth b$\uparrow$. Therefore, we impose the average packet size that will vary according to the type of packet (i.e., heavy and light packets). The average packet size for light-packets is 0.1KB, while the average packet size for heavy-packets is 80KB [14].

- Propagation delay: D_p for a packet is based on the round trip time (i.e., $\tau_{\downarrow\uparrow}$), in line with [14, 172], $\tau_{\downarrow\uparrow}$ computed as $\tau_{\downarrow\uparrow} = 0.03 \times l_d + 5$, where l_d is the distance with unit km, and the $\tau_{\downarrow\uparrow}$ time unit is ms.

- Fog nodes capabilities: fog nodes are simulated with different capabilities, hence, the service rate (μ) will vary from one fog node to another. The capabilities of fog nodes will significantly affect the processing ability (i.e., performance) of the fog node. The capability of the fog node is determined by the CPU frequency, therefore, fog nodes vary in CPU frequency having the CPU frequency ranging from 0.2GHz to 1.5GHz [166].

- Fogs interactions: as we adopted a Bayesian network to evaluate the satisfaction experience among collaborating fog nodes, each fog node develops a naive Bayes network model for all other fog nodes that it has interacted with. This is achieved by locally storing the binary values of ES score, which is either $satisfying$ or $unsatisfying$ interaction, denoted by 1 and 0, respectively. Then, computing the LoT score based on all the past interactions/coordinations between nodes and which will be used to identify the trustworthiness of the partner fog node.

6.5.2 Performance Evaluation and Discussion

This section demonstrates the numerical results of the simulation experimentation on the proposed secure COMITMENT model. This to validate the accuracy of the secure offloading in $\mathcal{F}og$-2-$\mathcal{F}og$ coordinations based on the COMITMENT approach to finding trusted fog nodes and avoiding malicious fog nodes.

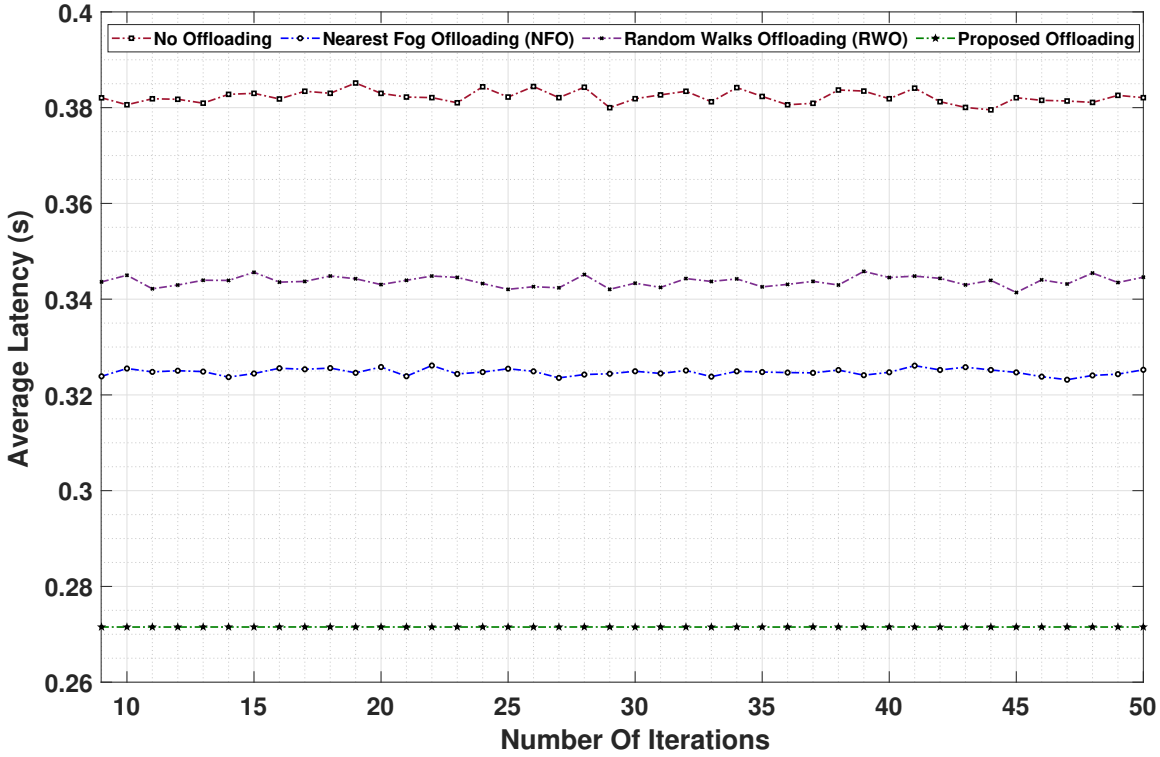
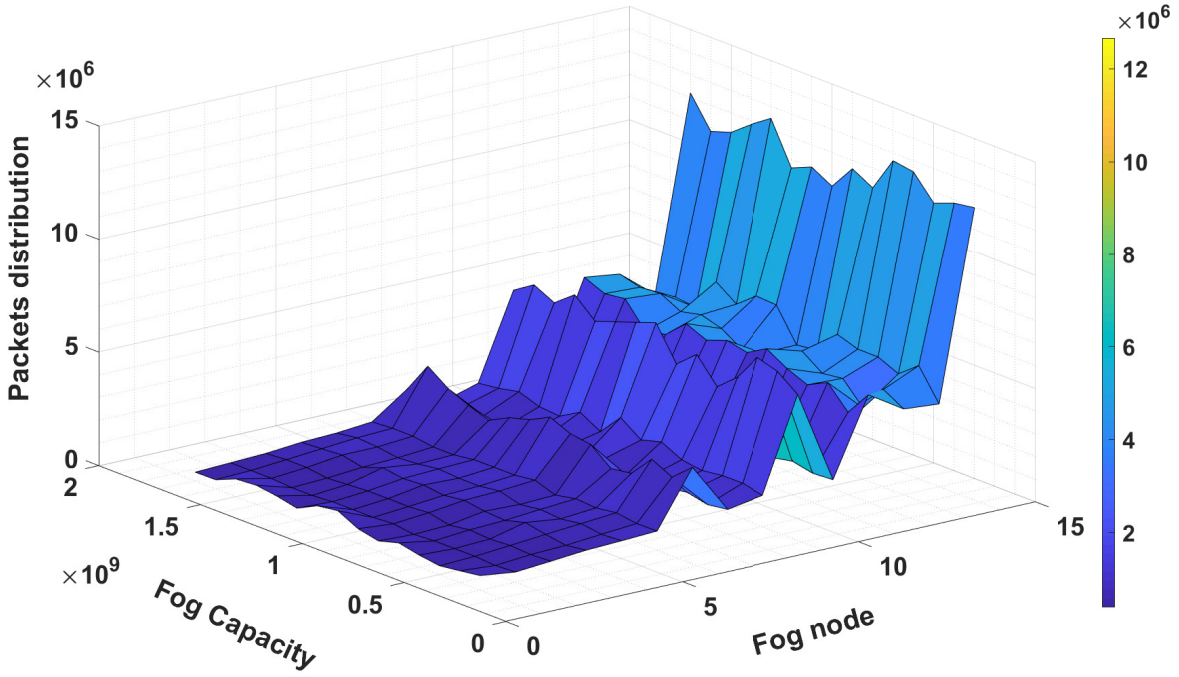


Figure 6.3: Average latency against two benchmark algorithms (RWO and NFO) and based on mixed type of packets

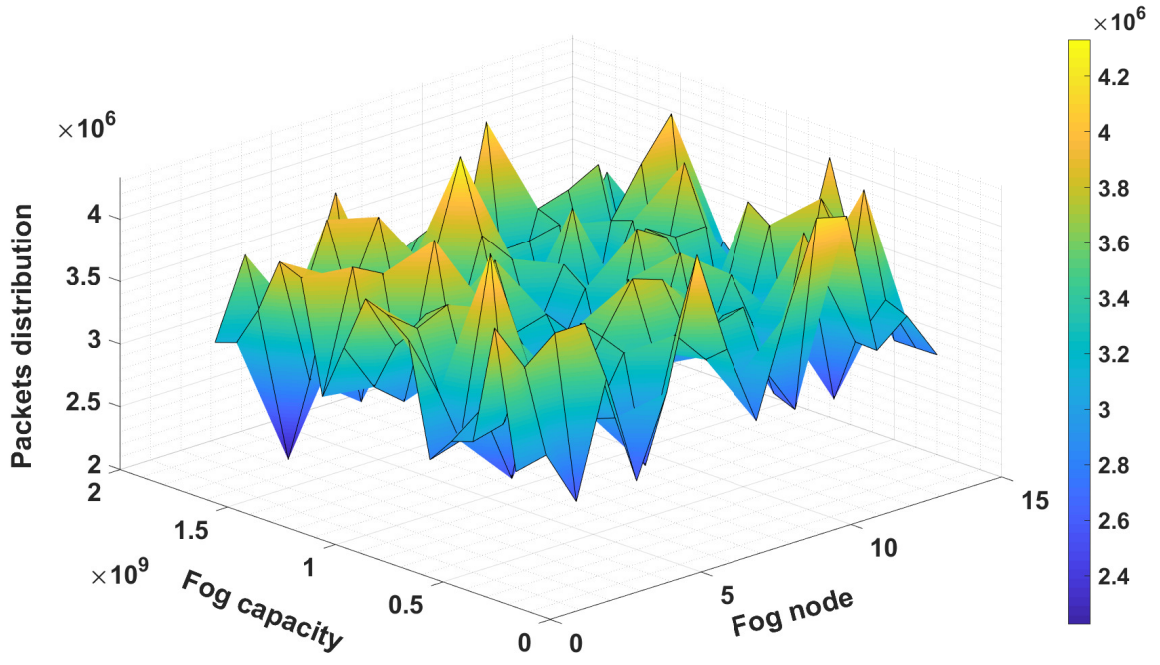
We first evaluate the performance of the Secure Offloading Algorithm (SOA) against two benchmark algorithms: i) Random Walks Offloading (RWO) [132, 133]. ii) Nearest Fog Offloading (NFO) [45, 165]. Figure 6.3 demonstrates the performance based on the average response time to all received service requests considering different packet types (i.e., heavy-packets and light-packets), however, the random number of heavy or light packets is fixed through out the experiment to ensure consistency in terms of load utilization against all the offloading algorithms. During the simulation of this experiment, we set the fog nodes with different capabilities, hence, fog nodes will vary in their service rate μ (as they have different capabilities). Thus, the capability of a fog node set is based on CPU frequency

with a minimum of 300×10^6Hertz, incremented by 100Hertz until it get to maximum CPU capability of 17×10^8Hertz. In addition, service arrival rate $\lambda = 2 \times 10^2$ packets per second as in [3], and λ is fixed during the experiment to ensure all the offloading algorithms have the same traffic arrival rate. Figure 6.3 shows the outcome of this experiment, thus the vertical line represents the average latency per algorithm to process service requests, and the horizontal line is the number of iterations carried out to ensure that the obtained results are consistent and not due to chance. It is clear that SOA has the lowest processing latency among other algorithms through all iterations. The highest processing time goes for No Offloading Consideration (NOC) as it does not consider the offloading when a node becomes congested. Hence, it ends up having small node capacity with large queue size (i.e., $\mu_i < \lambda_i$), and large node capacity with low queue size. The performance of RWO and NFO are better than NOC but still higher than SOA.

The following experiment was conducted based on packets distribution over the three offloading algorithms (i.e., SOA, RWO and NFO) on the fog nodes. The experiment settings are similar to our previous experiment, except having fixed packet type (i.e., all heavy or light packets) to ensure consistency. Figure 6.4 shows packet distribution, Figure6.4a shows packet's distribution according to SOA. While, Figure 6.4b shows packet's distribution according to RWO and NFO. It is clear that SOA has more sustainable packet's distribution compared to RWO and NFO, this due to the workload distribution strategy that each algorithm is uses. The SOA adopting the strategy and algorithm in Section 6.3 which allow the workload distribution on fog nodes based on not only fog's capabilities, but also thier current workload (i.e., queue size and active processes). Thus SOA distributes the packets with respect to fog node's capabilities. While, the other methods were relatively blind as they have not considered the current load (f_w) of fog nodes, rather they just allocate services according to the algorithm's policy.



(a) Average packets distribution according to SOA



(b) Average packets distribution according to RWO and NFO

Figure 6.4: Packets distribution

Figure 6.5: Coordination requests according to their type; *secure*, *malicious* and *anonymous* requests based on the LoT score

Figure 6.5 shows the results of detecting a malicious event (i.e., malicious coordination requests) in $\mathcal{F}og$-2-$\mathcal{F}og$ coordination. The malicious event detection is according to the LoT score. In this figure (Figure 6.5), the number of service requests is set to 1K and we have distributed the 1K requests in two iterations with this experiment. The first iteration is used to make enough coordinations between the fog nodes, so that they have a precise LoT score against each other, this mainly for simulation purposes to allow COMITMENT compute LoT score among nodes, however, even when there is no previous coordination among nodes, then the recommendations approach (Section 6.4.2) is triggered to compute the LoT score. The second iteration is where the COMITMENT operates on the fog nodes to observe the $\mathcal{F}og$-2-$\mathcal{F}og$ interactions and flag for any malicious events. The coordination requests in $\mathcal{F}og$-2-$\mathcal{F}og$ are grouped according to request's type; *secure*, *malicious* and *anonymous* requests as per Figure 6.5. The coordination requests are grouped based on the LoT score produced by the LoT function and according to the fuzzy logic in Figure 6.2.

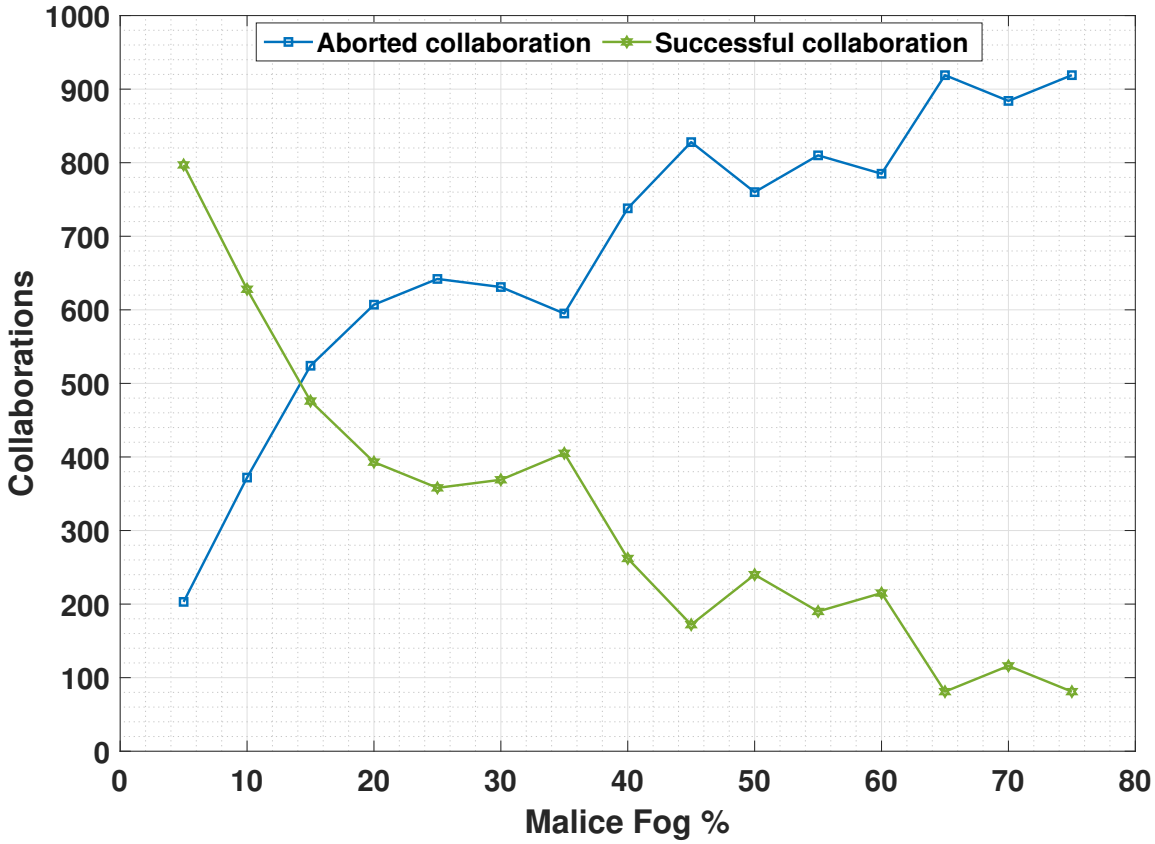


Figure 6.6: Average number of *successful* and *aborted* coordinations according to the percentage of malicious fogs

It is worth noting that the anonymous coordination requests in the figure are down to the fact that either there isn't enough LoT score gained from the past coordinations in $\mathcal{F}og$-2-$\mathcal{F}og$, or the gained LoT score on the borderline of the trustworthiness for a particular fog node.

In the $\mathcal{F}og$-2-$\mathcal{F}og$ coordinations, requests are accepted/declined according to LoT. The different types of coordination requests; *secure*, *malicious* and *anonymous* requests will control the decision of whether a coordination can be accepted or rejected between two fog nodes. Figure 6.6 shows the average number of *successful* and *aborted* coordinations according to the percentage of malicious fog nodes within the network. In this experiment, the initial percentage of malicious fog nodes in the network is 5%, then it increases by 5% up until we have 75% of the fog nodes being malicious. Through out the experiment, we observe the average number of *successful* and *aborted* coordinations requests. Although,

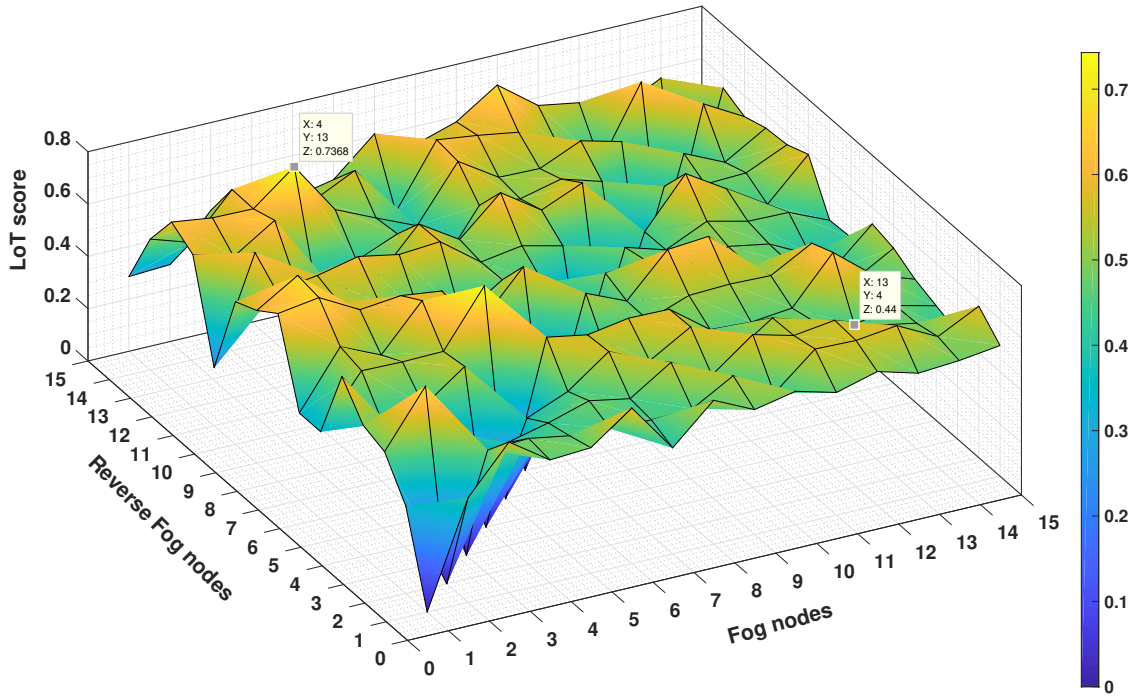


Figure 6.7: LoT score for the 15 participated fogs against each other proven that LoT is asymmetric

the *successful* and *aborted* coordinations are fluctuating while increasing the number of malicious fog nodes due to have higher average of amicable coordinations at a specific timestamp, however it is clear that with the increase in the malicious fog nodes in the network; the number of *successful* coordinations will be reduced and the number of *aborted* coordinations will be increased as per Figure 6.6 due to the increase of malicious fogs/events in the $\mathcal{F}og$-2-$\mathcal{F}og$ coordinations.

The next experiment is about fog node's trustworthiness policy, having the LoT score asymmetric and not transitive. Thus, each fog node has its own LoT score that defines its QoP, hence, if fog_a finds fog_b is trustworthy based on fog_a LoT score that meets its RoP towards fog_b, it is not necessarily that fog_b finds fog_a is trustworthy. Figure 6.7 shows the corresponding three dimensional view of the LoT score of the 15 participating fog nodes against each other. It is clear that the fog nodes have different LoT scores against each other, for example, the LoT score from fog_4 to fog_{13} is 0.7, while the the LoT score from fog_{13} to fog_4 is 0.4 as shown in the highlighted points in Figure 6.7. It is worth noting that the highest LoT score is 1, thus it is classified according to a fuzzy logic function into

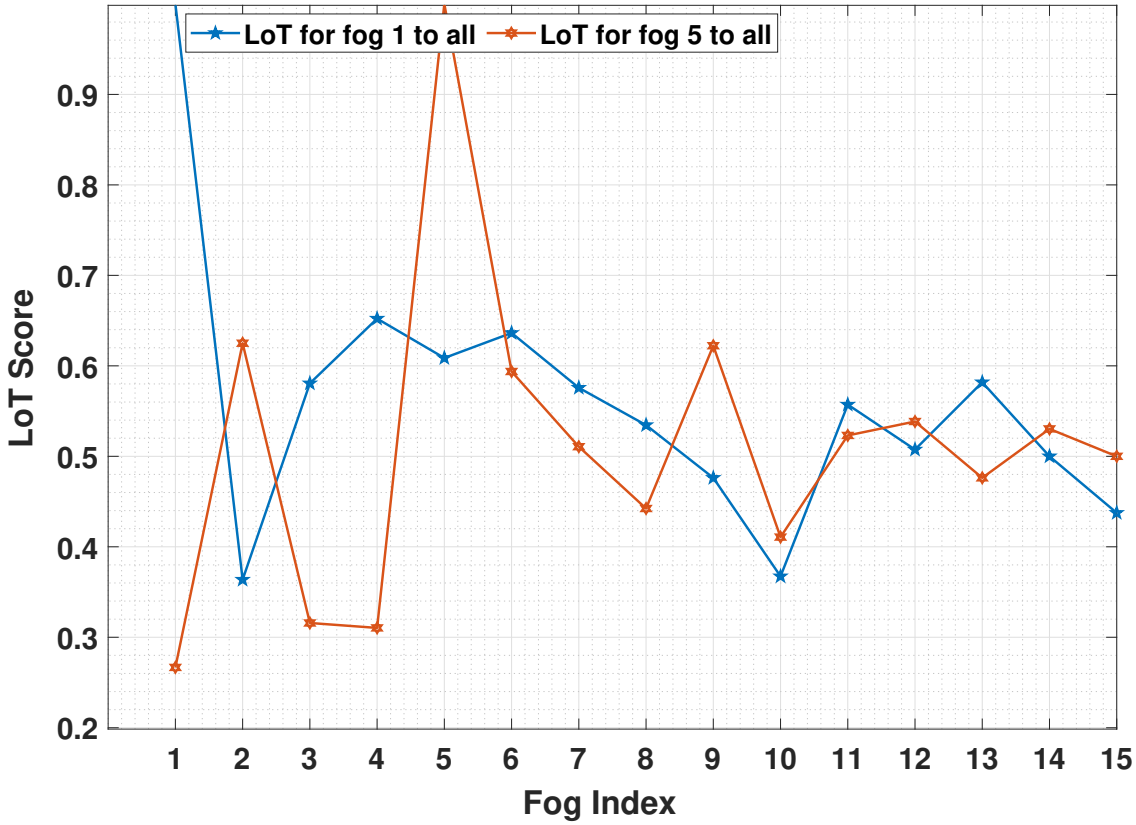


Figure 6.8: Lot score for fog_1 and fog_5 proven that LoT is not transitive

three main parts, Low, Medium and High to represent the trustworthiness between two fog nodes. Hence the LoT score of 1 is the indicator of absolute *trust*, while the score of 0 is the indicator of utter distrust. Similarly, the LoT score is not transitive, for example, if fog_a trusts fog_b and fog_b trusts fog_c, it is not necessarily true that fog_a trusts fog_c. This transitive property of fog nodes is proven in Figure 6.8. In this figure, the fog node fog_1 trust fog node fog_5 and fog node fog_5 trust fog_2, while fog_1 finds fog_2 not trustworthy.

6.6 Chapter Summary

Fog computing puts a substantial amount of cloud computing facilities at the edge of a network as opposed to establishing dedicated channels to a more centralized remote cloud infrastructure. This approach reduces service latency, and improves data security. Although, fog computing is generally considered to be more secure than cloud computing, there exist several challenges for preserving security and privacy in fog computing due to

the presence of rogue and malicious fog nodes that are undertaking attack within the networks. These malicious fog nodes appear to be legitimate and coaxes other fog nodes, but, in reality, these nodes are malicious in nature.

This chapter presented the Fog *COMputIng Trust manageMENT* (COMITMENT) approach. COMITMENT is a software-based approach that is responsible for providing a secure and trusted environment for fog nodes to share their resources and exchange data packets. COMITMENT helps fog nodes in making concise decisions for the fog node to point out the best node that it can cooperate with. The decision not only includes the best fog node that can handle the jobs efficiently but also the most trusted fog nodes that could provide best QoS and QoP. COMITMENT's main procedures and processes look after the safe offloading and finding best fog nodes for coordination based on trust and recommendations models to avoid malicious fog nodes.

COMITMENT and it's Secure Offloading Algorithm (SOA) has outperformed the competitive benchmark algorithms, namely Random Walks Offloading (RWO) and Nearest Fog Offloading (NFO) in the experiments to verify the validity and performance. The experiments were conducted on the performance (in terms of latency) and packets distribution over fog nodes in the $\mathcal{F}og$-2-$\mathcal{F}og$ model. In addition, the evaluation results proven that COMITMENT able to identify fog node's trustworthiness and detecting malicious events and attacks as it's proven by increasing the malicious fog nodes in the network, the number of $successful$ coordination will be reduced and the number of *aborted* coordination will be increased. The decision of offloading and requests accepts/declines was made according to the LoT score. Moreover, the evaluation results proven that the LoT score is asymmetric and not transitive, hence, each fog node has it's own LoT score, to explain more if f_a finds f_b is trustworthy based on f_a LoT score towards fog_b, it is not necessarily that f_b finds f_a is trustworthy. Similarly, it's proven that the LoT score is not transitive means if fog_a trust fog_b and fog_b trust fog_c, it is not necessarily true that fog_a trusts fog_c. Hence, by ensuring the level of security and trust within the network, the RO5 has been addressed, thus RQ5 and RQ6 are fulfilled.

CHAPTER 7

Conclusions and Future Directions

I may not have gone where I intended to go, but
I think I have ended up where I needed to be.

Douglas Adams

7.1 Conclusion

The Internet of Things has become, indeed, a reality. IoT technology is expanding significantly, thus, the IoT has become a trend that promises a substantial economic and scientific value for industry and academia in the upcoming years. However, the expanding of the IoT and its technologies nature has brought challenges (e.g., such as data handling and resource management) to the current networking paradigms (e.g., cloud computing) due to the tremendous amount of data that are generated every second or even millisecond. A novel computing paradigm which is introduced to support and unleash the full extent of the IoT is **fog computing**. Fog computing as a concept was introduced in the last few years, hence, the corresponding studies/research on this computing paradigm are still in their infancy. Although the theoretical aspects of fog computing have already been introduced, there is a lack of concrete fog-based solutions that fulfill the full stack of IoT requirements (e.g., fast processing and resource management). Hence, this thesis comes into the action to fulfill the requirement of developing a comprehensive and concrete solution to tackle the limitations/challenges (refer to Section 2.4.3) of deploying fog computing in the IoT network.

The scope and focus of this thesis is to have a stable performance for fog computing to aid the IoT-services. Aspects related to the performance stability of fog computing involve the development of reliable resource management and trusted network. Hence, this is to ensure best Quality of Service (QoS), Quality of Experience (QoE) and Quality of Protection (QoP) to the end-users. In this thesis, the proposed fog-based solutions and algorithms are designed to address the limitations of; *(i)* network resources management by an efficient resource provisioning algorithm to ensure the QoS, *(ii)* services reliability and availability in high-traffic network, hence to ensure the QoE, and, *(iii)* security and privacy through evolving trusted network environment for fogs to share resources and data, hence avoiding malicious attacks to ensure the QoP. The solutions and algorithms for the above mentioned challenges are integrated in the proposed **Cognitive Fog** (CF).

The fundamental contributions and results of this thesis are concluded and grouped based on the inspired Research Questions (RQ):

- The contribution for **RQ1** is the novel development of CF computing to empower fog nodes with reasoning, learning, and adaptation capabilities so that, it would weave these fog nodes into services provisioning models. CF advocates that fog can interpret the gathered/received data in a way that mimics the process of cognition in the human mind. The operations of CF run over four connected worlds; *data world* that features both raw and filtered data, *processes world* featuring processing models, *fog world* featuring the CF processes and controls, and finally the *things world* that is controlled by the CF to adapt with environment changes. Moreover, one of the important characteristics of CF is node federations, which is about gathering multiple fog nodes to perform/achieve a specific task in a certain situation. There are two types of federations; planned and ad-hoc federations. Planned federations are formed at the design-time, while ad-hoc federations are formed at run-time. The performance of the developed CF test-bed shows that fog can perform better on repeated processes. The core concepts and design of cognitive fog are discussed in Chapter 3.

- The contribution for **RQ2** is the collaboration model of fog and cloud with a set of criteria for selecting data recipients. These criteria define where data of things should be sent (cloud, fog, or both) and in what order (cloud then fog or fog then cloud or both concurrently). This fog-cloud collaboration was illustrated with different levels of recommendations about the appropriate data recipients. For instance an IoT application that is keen to handle continuous data-streaming would not consider sending data from things to clouds but from things to fogs. Contrarily, an IoT application that is keen to handle a high amount of data-exchange would consider sending data from things to clouds but not from things to fogs. Different concerns and different priorities mean different data recipients. This is supported by a healthcare driven IoT case study deployed on a test-bed to demonstrate fog-cloud collaboration. The experiments targeted frequency and time criteria along with the continuous stream feature. The objective was to assist engineers who are in-charge of developing IoT applications to know what is best for their system/network. The fog-cloud collaboration and criteria for selecting data recipients are discussed in Chapter 4.

- The contribution for **RQ3** and partially **RQ4** is a novel Fog Resource manAgeMEnt Scheme (FRAMES) that promotes load balancing to address the latency concern of service request's received from things. This is based on the load distribution algorithm in the $\mathcal{F}$og-2-$\mathcal{F}$og coordination model that achieves an optimal workload among the collaborative nodes. The load distribution model considers not only the queue length of a node, but also the node's capabilities (i.e., CPU frequency) and their performance with different request types, such as, heavy-request (e.g., from sensor) and light-request (i.e., from CCTV). FRAMES is discussed in Chapter 5.

- The contribution for **RQ4** and **RQ5** is a mathematical model that backs the decision of load balancing among fog nodes. This investigate the time delay issue and the requests offloading opportunities in the $\mathcal{F}$og-2-$\mathcal{F}$og coordination model. Hence, a time-cost function is developed to compute the time-cost for a service to be processed in multi-nodes based on the number of participant nodes and network conditions. Simulation results shows that the proposed Optimal Fog Algorithm (OFA) has the lowest service response time in comparison with two benchmark algorithms named; Random Walk Algorithm (RWA) and Neighbouring Fogs Algorithm (NFA). Moreover, OFA not only outperformed RWA and NFA in latency, but also in the service's packets distribution over nodes dependent upon their capabilities. The mathematical model was discussed in Chapter 5.

- The contribution for **RQ6** is met by introducing a novel Fog COMputIng Trust manageMENT (COMITMENT) approach to impart useful prognostic information on networked nodes trustworthiness. Hence, providing a secure and trusted networking environment for nodes to share their resources and exchange data securely and efficiently. This is achieved based on a novel trust/recommendation model and algorithm that helps nodes make the right decision for selecting the appropriate fog nodes to collaborate with in the $\mathcal{F}$og-2-$\mathcal{F}$og coordination environment. This to support during the offloading processes to avoid malicious nodes and attacks. Thereof, this process includes assessing the trustworthiness level of the nominated nodes to ensure that the QoP and QoS are met by the hosting node before a coordination is formed. COMITMENT outperformed the competitive benchmark algorithms in the experiments to

verify the validity and performance. The experiments were conducted on the performance (in term of latency), fog node's trustworthiness, packets distribution over fog nodes and detecting malicious event and attacks in the $\mathcal{F}og$-2-$\mathcal{F}og$ model. The core concepts and design of COMITMENT and the trust and recommendation model are discussed in Chapter 5.

The overall evaluation results of the proposed algorithms that reflection thesis contributions are promising. Defining the Cognitive Fog (CF) functional and non-functional requirements that fit under the umbrella of performance requirements (i.e., functional requirements) and general security requirements (i.e., non-functional requirements) helps in adding fog during both $f$og-$c$loud collaborations and $f$og-2-$f$og coordination within the IoT network. To explain more, with regards to $f$og-$c$loud collaborations, the set of data-recipient selection criteria - frequency, sensitivity, freshness, time, volume, etc - have been proposed to define where data of things should be sent (cloud, fog, or both) are valid based on the evaluation results. The results proven for Config$_4$: $\mathsf{T} \rightarrow \mathsf{F}$ and Config$_2$: $\mathsf{T} \rightarrow \mathsf{F} \rightarrow \mathsf{C}$ are in line with our recommendations for both the time criterion (reflect the delay) and the frequency criterion (reflect the traffic) of $\mathsf{T} \rightarrow \mathsf{F} \rightarrow \mathsf{C}$ and $\mathsf{T} \rightarrow \mathsf{F}$ topologies being recommended, while $\mathsf{T} \rightarrow \mathsf{C} \rightarrow \mathsf{F}$ and $\mathsf{T} \rightarrow \mathsf{C}$ topologies being not-recommended. Moreover, with regards to $f$og-2-$f$og coordination, FRAMES results proven that the proposed OFA has the lowest service response time in comparison with RWA and NFA, also, OFA has not only outperformed on RWA and NFA in latency, but also in the service packets distribution over fog nodes upon their capabilities. In term of network security, the evaluation results proven that COMITMENT able to identify fog node's trustworthiness and detecting malicious events and attacks as it's proven by increasing the malicious fog nodes in the network, the number of $successful$ coordination will be reduced and the number of *aborted* coordination will be increased due to the ability of identifying malicious event. The decision of offloading and requests accepts/declines was made according to the LoT score which accurate as it's has been proven that the LoT score is asymmetric and not transitive, hence, each fog node has it's own LoT score, to explain more if f_a finds f_b is trustworthy based on f_a LoT score towards fog_b, it is not necessarily that f_b finds f_a is trustworthy. Similarly, it's proven that the LoT score is not transitive means if fog_a trust fog_b and fog_b trust fog_c, it is not necessarily true that fog_a trusts fog_c.

7.2 Future Directions

Since fog computing is a recent and emerging research field, there are many open research questions and promising research directions. In this thesis, a CF test-bed, collaboration and coordination models and algorithms were presented to aid the deployment of fog computing in the IoT network. However, this work can be extended in many possible directions as there are a number of approaches, boundaries and special cases which require attention from the research community, such as, the impact of encrypted traffic in CF on the cognitive capabilities and learning activities. The listing below presents some possible future improvements of the CF.

- **Energy-efficient network:** The energy property has not been studied in the proposed work. Therefore this could be an interesting research direction due to the amount of energy that is being consumed in such networking activities. The annual energy consumption of US data-centres was estimated at 91 billion kilowatt-hours in 2013 which is enough to power all households in New York City for two years [173]. It is even expected to have increased to approximately 140 billion kilowatt-hours in 2020 which will cost US 13 billion annually [174]. Hence, improving the energy-efficiency computing paradigm is a research problem of the utmost interest in academia and industry.

- **Supporting big data mining:** the largest percentage of data produced today is coming from media, videos and other similar data streams which became a very important source in training data mining and machine learning algorithms used for research and commercial purposes. The problem of learning from such data streams presents unprecedented challenges, especially in resource-constrained scenarios. Therefore, fog computing can be adopted to aid the analysis of such massive data streams that requires a set of techniques dedicated to help in running experiments and implement algorithms to deal with scalability and performance challenges. Hence, fog computing could have the potential to aid data mining techniques.

- **Container-based services:** container technology has emerged recently and gained popularity among the Research and Development (R&D) community. There are several benefits of adopting container technology, such as resource efficiency and density. Instead of hardware level virtualization, containers use operating system level virtualization. Only the application, and the libraries and file system needed are packed in a container. It enables not only lightweight deployment and easy migration, but also can increase the scalability and the cross-platform compatibility which will be beneficial to fog computing. In order to provide an efficient container powered fog nodes services, algorithms regarding container service orchestration, and scheduling and migration algorithms need to be investigated.

- **Fog landscape nodes rearrangements:** the fog nodes arrangements in the fog landscape is a promising research direction to be considered in the future work. After a specific number of fog nodes in the fog landscape fail, it is likely that the topology constellation/arrangements is not optimal in terms of latency, bandwidth, location mapping, and connection preservation. Therefore, the rearrangement of fog nodes to build a new effective nodes topology is a noteworthy aspect for future work.

7.3 Final Remarks

In conclusion, this thesis focuses on developing a stable performance for fog computing to aid the IoT-services and cloud computing in the ever growing industry of smart things, hence ensuring best Quality of Service (QoS), Quality of Experience (QoE), and the Quality of Protection (QoP) to the end-users. Aspects related to the performance stability of fog computing involve the development of:

- **Cognitive fog nodes**: in order to allow fog nodes to be cognitive, a reasonable training is required to allow the learning processes to enrich nodes knowledge, hence the decision of fog in participating in processes.
- **Reliable resources management**: the resources management is mostly controlled by the scheduling and offloading algorithms. In general, if all fog nodes have low loads, offloading is unnecessary, and if all fog nodes have heavy load, offloading will not help to reduce the delay. Offloading only helps when there is a high degree of variance among the fog nodes.
- **Trusted networks**: identifying malicious fog nodes is not an easy task. Also, increasing the safety measures for security/privacy can increase the overall processing/checking time, hence reducing the QoE. Therefore, a network specific security and privacy arrangements should be considered to endure both QoE and QoP.

Bibliography

[1] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet of Things Journal*, vol. 3, pp. 1171–1181, Dec 2016.

[2] Q. Fan and N. Ansari, "Towards workload balancing in fog computing empowered iot," *IEEE Transactions on Network Science and Engineering*, pp. 1–1, 2018.

[3] X. Wang, Z. Ning, and L. Wang, "Offloading in internet of vehicles: A fog-enabled real-time traffic management system," *IEEE Transactions on Industrial Informatics*, vol. 14, pp. 4568–4578, Oct 2018.

[4] W. Kim and S. Chung, "User-participatory fog computing architecture and its management schemes for improving feasibility," *IEEE Access*, vol. 6, pp. 20262–20278, 2018.

[5] X. Zeng, S. K. Garg, P. Strazdins, P. P. Jayaraman, D. Georgakopoulos, and R. Ranjan, "Iotsim: A simulator for analysing iot applications," *Journal of Systems Architecture*, vol. 72, pp. 93 – 107, 2017. Design Automation for Embedded Ubiquitous Computing Systems.

[6] S. Ningning, G. Chao, A. Xingshuo, and Z. Qiang, "Fog computing dynamic load balancing mechanism based on graph repartitioning," *China Communications*, vol. 13, pp. 156–164, March 2016.

[7] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, pp. 7–18, May 2010.

[8] L. Wang, G. von Laszewski, A. Younge, X. He, M. Kunze, J. Tao, and C. Fu, "Cloud computing: a perspective study," *New Generation Computing*, vol. 28, pp. 137–146, Apr 2010.

[9] J. Sun, G. Zhu, G. Sun, D. Liao, Y. Li, A. K. Sangaiah, M. Ramachandran, and V. Chang, "A reliability-aware approach for resource efficient virtual network function deployment," *IEEE Access*, vol. 6, pp. 18238–18250, 2018.

[10] F. Mattern and C. Floerkemeier, *From the Internet of Computers to the Internet of Things*, pp. 242–259. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010.

[11] M. Al-khafajiy, T. Baker, A. Waraich, D. Al-Jumeily, and A. Hussain, "Iot-fog optimal workload via fog offloading," in *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, pp. 359–364, Dec 2018.

[12] C. Systems, "Fog computing and the internet of things: Extend the cloud to where the things are," 2016.

[13] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," *Cisco Internet Business Solutions Group (IBSG)*, vol. 1, pp. 1–11, 01 2011.

[14] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On reducing iot service delay via fog offloading," *IEEE Internet of Things Journal*, vol. 5, pp. 998–1010, April 2018.

[15] S. Khanagha, H. Volberda, and I. Oshri, "Business model renewal and ambidexterity: Structural alteration and strategy formation process during transition to a cloud business model," *R and D Management*, vol. 44, 06 2014.

[16] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, (New York, NY, USA), pp. 13–16, ACM, 2012.

[17] M. Al-khafajiy, T. Baker, H. Al-Libawy, A. Waraich, C. Chalmers, and O. Alfandi, "Fog computing framework for internet of things applications," in *2018 11th International Conference on Developments in eSystems Engineering (DeSE)*, pp. 71–77, Sep. 2018.

[18] K. Kang, W. Cong, and T. Luo, "Fog computing for vehicular ad-hoc networks: Paradigms, scenarios, and issues," *The Journal of China Universities of Posts and Telecommunications*, vol. 23, pp. 56–96, 04 2016.

[19] S. Soo, C. Chang, and S. N. 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)*, pp. 561–566, Dec 2016.

[20] M. Al-khafajiy, L. Webster, T. Baker, and A. Waraich, "Towards fog driven iot healthcare: challenges and framework of fog computing in healthcare," in *In Proceedings of the 2nd International Conference on Future Networks and Distributed Systems*, p. 9, ACM, Jun 26 2018.

[21] Z. Maamar, T. Baker, N. Faci, E. Ugljanin, M. Al-Khafajiy, and V. Buregio, "Towards a seamless coordination of cloud and fog: Illustration through the internet-of-things," *The 34th ACM/SIGAPP Symposium on Applied Computing*, 2019.

[22] Z. Maamar, T. Baker, N. Faci, E. Ugljanin, Y. Atif, M. Al-Khafajiy, and M. Sellami, "Cognitive computing meets the internet of things," in *Proceedings of the 13th International Conference on Software Technologies :*, pp. 741–746, 2018.

[23] S. Yi, Z. Qin, and Q. Li, "Security and privacy issues of fog computing: A survey," in *International conference on wireless algorithms, systems, and applications*, pp. 685–695, Springer, 2015.

[24] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Communications Surveys Tutorials*, vol. 20, pp. 1826–1857, thirdquarter 2018.

[25] B. McMillin and T. Zhang, "Fog computing for smart living," *Computer*, vol. 50, pp. 5–5, Feb 2017.

[26] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Communications Surveys Tutorials*, vol. 20, pp. 416–464, Firstquarter 2018.

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

[28] P. Habibi, M. Farhoudi, S. Kazemian, S. Khorsandi, and A. Leon-Garcia, "Fog computing: A comprehensive architectural survey," *IEEE Access*, vol. 8, pp. 69105–69133, 2020.

[29] M. Al-khafajiy, T. Baker, H. Al-Libawy, Z. Maamar, M. Aloqaily, and Y. Jararweh, "Improving fog computing performance via fog-2-fog collaboration," *Future Generation Computer Systems*, vol. 100, pp. 266–280, 2019.

[30] D. Puthal, R. Ranjan, A. Nanda, P. Nanda, P. P. Jayaraman, and A. Y. Zomaya, "Secure authentication and load balancing of distributed edge datacenters," *Journal of Parallel and Distributed Computing*, vol. 124, pp. 60–69, 2019.

[31] M. Al-khafajiy, T. Baker, A. Waraich, D. Al-Jumeily, and A. Hussain, "Iot-fog optimal workload via fog offloading," in *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion*, pp. 359–364, IEEE, Dec 17 2018.

[32] X. Wang, L. T. Yang, X. Xie, J. Jin, and M. J. Deen, "A cloud-edge computing framework for cyber-physical-social services," *IEEE Communications Magazine*, vol. 55, pp. 80–85, Nov 2017.

[33] D. Puthal, S. Nepal, R. Ranjan, and J. Chen, "Threats to networking cloud and edge datacenters in the internet of things," *IEEE Cloud Computing*, vol. 3, pp. 64–71, May 2016.

[34] D. Puthal, S. P. Mohanty, S. A. Bhavake, G. Morgan, and R. Ranjan, "Fog computing security challenges and future directions [energy and security]," *IEEE Consumer Electronics Magazine*, vol. 8, no. 3, pp. 92–96, 2019.

[35] R. Chen, J. Guo, and F. Bao, "Trust management for soa-based iot and its application to service composition," *IEEE Transactions on Services Computing*, vol. 30, p. 3, Oct 2014.

[36] A. Dastjerdi, H. Gupta, R. Calheiros, S. Ghosh, and R. Buyya, "Fog Computing: principles, architectures, and applications," in *Internet of Things*, Morgan Kaufmann, 2016.

[37] A. Kapsalis, P. Kasnesis, I. S. Venieris, D. I. Kaklamani, and C. Z. Patrikakis, "A cooperative fog approach for effective workload balancing," *IEEE Cloud Computing*, vol. 4, pp. 36–45, March 2017.

[38] O. C. A. W. Group, *OpenFog reference architecture for fog computing.* Available at `https://www.openfogconsortium.org/wp-content/uploads/OpenFog_Reference_Architecture_2_09_17-FINAL.pdf`, Last Visit: February.10.2019.

[39] M. Al-khafajiy, T. Baker, H. Al-Libawy, Z. Maamar, M. Aloqaily, and Y. Jararweh, "Improving fog computing performance via fog-2-fog collaboration," *Future Generation Computer Systems*, vol. 100, pp. 266–280, 2019.

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

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

[42] J. Zhou, Z. Cao, X. Dong, and A. V. Vasilakos, "Security and privacy for cloud-based iot: Challenges," *IEEE Communications Magazine*, vol. 55, no. 1, pp. 26–33, 2017.

[43] T. Wang, G. Zhang, A. Liu, M. Z. A. Bhuiyan, and Q. Jin, *A secure IoT service architecture with an efficient balance dynamics based on cloud and edge computing.* IEEE Internet of Things Journal, 2018.

[44] C. Alcaraz, R. Roman, P. Najera, and J. Lopez, "Security of industrial sensor network-based remote substations in the context of the internet of things," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1091–1104, 2013.

[45] Y. Xiao and M. Krunz, "Qoe and power efficiency tradeoff for fog computing networks with fog node cooperation," in *Conference on Computer Communications* (I. I. 2017-ieee, ed.), pp. 1–9, IEEE, 2017.

[46] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: architecture, key technologies, applications and open issues," *Journal of network and computer applications*, vol. 98, pp. 27–42, 2017.

[47] A. V. Dastjerdi and R. Buyya, "Fog computing: Helping the internet of things realize its potential," *Computer*, vol. 49, no. 8, pp. 112–116, 2016.

[48] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, "A design science research methodology for information systems research," *Journal of management information systems*, vol. 24, no. 3, pp. 45–77, 2007.

[49] J. W. Creswell and J. D. Creswell, *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications, 2017.

[50] M. Al-khafajiy, L. Webster, T. Baker, and A. Waraich, "Towards fog driven iot healthcare: Challenges and framework of fog computing in healthcare," in *Proceedings of the 2Nd International Conference on Future Networks and Distributed Systems*, ICFNDS '18, (New York, NY, USA), pp. 9:1–9:7, ACM, 2018.

[51] M. Abdmeziem, D. Tandjaoui, and I. Romdhani, "Architecting the Internet of Things: State of the Art," in *Robots and Sensor Clouds* (A. Koubaa and E. Shakshuki, eds.), Springer International Publishing, 2016.

[52] P. Barnaghi and A. Sheth, "On Searching the Internet of Things: Requirements and Challenges," *IEEE Intelligent Systems*, vol. 31, no. 6, 2016.

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

[54] DZone, "The Internet of Things, Application, Protocls, and Best Practices," tech. rep., https://dzone.com/guides/iot-applications-protocols-and-best-practices, 2017 (visited in May 2017).

[55] P. Mell, T. Grance, *et al.*, "The nist definition of cloud computing," 2011.

[56] T. Dillon, C. Wu, and E. Chang, "Cloud computing: issues and challenges," in *2010 24th IEEE international conference on advanced information networking and applications*, pp. 27–33, Ieee, 2010.

[57] M. Boniface, B. Nasser, J. Papay, S. C. Phillips, A. Servin, X. Yang, Z. Zlatev, S. V. Gogouvitis, G. Katsaros, K. Konstanteli, *et al.*, "Platform-as-a-service architecture for real-time quality of service management in clouds," in *2010 Fifth International Conference on Internet and Web Applications and Services*, pp. 155–160, IEEE, 2010.

[58] R. Jain and S. Paul, "Network virtualization and software defined networking for cloud computing: a survey," *IEEE Communications Magazine*, vol. 51, no. 11, pp. 24–31, 2013.

[59] A. Hameed, A. Khoshkbarforoushha, R. Ranjan, P. P. Jayaraman, J. Kolodziej, P. Balaji, S. Zeadally, Q. M. Malluhi, N. Tziritas, A. Vishnu, *et al.*, "A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems," *Computing*, vol. 98, no. 7, pp. 751–774, 2016.

[60] S. M. Parikh, "A survey on cloud computing resource allocation techniques," in *2013 Nirma University International Conference on Engineering (NUiCONE)*, pp. 1–5, IEEE, 2013.

[61] J. Gibson, R. Rondeau, D. Eveleigh, and Q. Tan, "Benefits and challenges of three cloud computing service models," in *2012 Fourth International Conference on Computational Aspects of Social Networks (CASoN)*, pp. 198–205, IEEE, 2012.

[62] S. Ullah and Z. Xuefeng, "Cloud computing research challenges," *arXiv preprint arXiv:1304.3203*, 2013.

[63] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1171–1181, 2016.

[64] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, and R. Ranjan, "Fog computing: Survey of trends, architectures, requirements, and research directions," *IEEE access*, vol. 6, pp. 47980–48009, 2018.

[65] C. Avasalcai, I. Murturi, and S. Dustdar, "Edge and fog: A survey, use cases, and future challenges," *Fog Computing: Theory and Practice*, pp. 43–65, 2020.

[66] A. Taivalsaari and T. Mikkonen, "A Roadmap to the Programmable World: Software Challenges in the IoT Era," *IEEE Software*, vol. 34, no. 1, 2017.

[67] M. Satyanarayanan, P. Bahl, R. Cáceres, and N. Davies, "The Case for VM-based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, vol. 8, no. 4, 2009.

[68] F. Bonomi, R. Milito, P. Natarajan, and J. Zhu, "Fog Computing: A Platform for Internet of Things and Analytics," in *Big Data and Internet of Things: A Roadmap for Smart Environments, Studies in Computational Intelligence*, Cisco, Springer International Publishing, 2014.

[69] B. Varghese, N. Wang, D. Nikolopoulos, and R. Buyya, "Feasibility of Fog Computing," *arXiv preprint arXiv:1701.05451*, 2017.

[70] M. Aazam and E. Huh, "Fog Computing and Smart Gateway Based Communication for Cloud of Things," in *Proceedings of the International Conference on Future Internet of Things and Cloud (FiCloud'2014)*, (Barcelona, Spain), 2014.

[71] G. Lewis, S. Echeverría, S. Simanta, B. Bradshaw, and J. Root, "Tactical Cloudlets: Moving Cloud Computing to the Edge," in *Proceedings of the IEEE Military Communications Conference (MILCOM'2014)*, (Baltimore, USA), 2014.

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

[73] I. Petri, J. Diaz-Montes, O. Rana, Y. Rezgui, M. Parashar, and L. Bittencourt, "Coordinating Data Analysis & Management in Multi-Layered Clouds," in *Proceedings of the EAI International Conference on Cloud, Networking for IoT Systems (CN4IoT'2015)*, (Rome, Italy), 2015.

[74] Z. Wen, R. Yang, P. Garraghan, T. Lin, J. Xu, and M. Rovatsos, "Fog Orchestration for Internet of Things Services," *IEEE Internet Computing*, vol. 21, no. 2, March-April 2017.

[75] D. Chekired, L. Khoukhi, and H. T. Mouftah, "Industrial IoT Data Scheduling Based on Hierarchical Fog Computing: A Key for Enabling Smart Factory," *IEEE Trans. Industrial Informatics*, vol. 14, no. 10, 2018.

[76] S. Chen, H. Xu, D. Liu, B. Hu, and H. Wang, "A vision of iot: Applications, challenges, and opportunities with china perspective," *IEEE Internet of Things journal*, vol. 1, no. 4, pp. 349–359, 2014.

[77] R. Van Kranenburg and A. Bassi, "Iot challenges," *Communications in Mobile Computing*, vol. 1, no. 1, p. 9, 2012.

[78] R. Mahmoud, T. Yousuf, F. Aloul, and I. Zualkernan, "Internet of things (iot) security: Current status, challenges and prospective measures," in *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, pp. 336–341, IEEE, 2015.

[79] I. Lee and K. Lee, "The internet of things (iot): Applications, investments, and challenges for enterprises," *Business Horizons*, vol. 58, no. 4, pp. 431–440, 2015.

[80] J. Rivera and R. van der Meulen, "Gartner says the internet of things will transform the data center," *Retrieved August*, vol. 5, p. 2014, 2014.

[81] S. A. Al-Qaseemi, H. A. Almulhim, M. F. Almulhim, and S. R. Chaudhry, "Iot architecture challenges and issues: Lack of standardization," in *2016 Future Technologies Conference (FTC)*, pp. 731–738, IEEE, 2016.

[82] A. Pal and B. Purushothaman, *IoT technical challenges and solutions*. Artech House, 2016.

[83] A.-R. Sadeghi, C. Wachsmann, and M. Waidner, "Security and privacy challenges in industrial internet of things," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, IEEE, 2015.

[84] E. Levy, "Crossover: online pests plaguing the off line world," *IEEE Security & Privacy*, vol. 1, no. 6, pp. 71–73, 2003.

[85] Z. B. Celik, E. Fernandes, E. Pauley, G. Tan, and P. McDaniel, "Program analysis of commodity iot applications for security and privacy: Challenges and opportunities," *ACM Computing Surveys (CSUR)*, vol. 52, no. 4, p. 74, 2019.

[86] K. Rawlinson, "Hp study reveals 70 percent of internet of things devices vulnerable to attack," *HP*, 2014.

[87] E. Bertino and N. Islam, "Botnets and internet of things security," *Computer*, no. 2, pp. 76–79, 2017.

[88] S. Aguzzi, D. Bradshaw, M. Canning, M. Cansfield, P. Carter, G. Cattaneo, S. Gusmeroli, G. Micheletti, D. Rotondi, and R. Stevens, "Definition of a research and innovation policy leveraging cloud computing and iot combination," *Final Report, European Commission, SMART*, vol. 37, p. 2013, 2013.

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

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

[91] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Communications Surveys Tutorials*, vol. 20, pp. 1826–1857, thirdquarter 2018.

[92] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Communications Surveys Tutorials*, vol. 20, pp. 1826–1857, thirdquarter 2018.

[93] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the Suitability of Fog Computing in the Context of Internet of Things," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 46–59, 2018.

[94] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, and S. Clarke, "Middleware for internet of things: A survey," *IEEE Internet of Things Journal*, vol. 3, pp. 70–95, Feb 2016.

[95] T. Wang, J. Zhou, A. Liu, M. Z. A. Bhuiyan, G. Wang, and W. Jia, "Fog-based computing and storage offloading for data synchronization in iot," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4272–4282, 2018.

[96] M. Chiang and T. Zhang, "Fog and iot: An overview of research opportunities," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 854–864, 2016.

[97] G. Zhang, F. Shen, Y. Yang, H. Qian, and W. Yao, "Fair task offloading among fog nodes in fog computing networks," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2018.

[98] C. Fricker, F. Guillemin, P. Robert, and G. Thompson, "Analysis of an offloading scheme for data centers in the framework of fog computing," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 1, pp. 16:1–16:18, Sept. 2016.

[99] M. A. Hassan, M. Xiao, Q. Wei, and S. Chen, "Help your mobile applications with fog computing," in *2015 12th Annual IEEE International Conference on Sensing, Communication, and Networking - Workshops (SECON Workshops)*, pp. 1–6, June 2015.

[100] S. Ningning, G. Chao, A. Xingshuo, and Z. Qiang, "Fog computing dynamic load balancing mechanism based on graph repartitioning," *China Communications*, vol. 13, pp. 156–164, March 2016.

[101] I. Stojmenovic and S. Wen, "The Fog Computing Paradigm: Scenarios and Security Issues," *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems*, vol. 2, pp. 1–8, 2014.

[102] R. Roman, J. Lopez, and M. Manbo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Gener. Comput. Syst.*, vol. 78, pp. 680–698, January 2018.

[103] J. Ni, K. Zhang, X. Lin, and X. S. Shen, "Securing fog computing for internet of things applications: Challenges and solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 601–628, 2017.

[104] S. A. Soleymani, A. H. Abdullah, M. Zareei, M. H. Anisi, C. Vargas-Rosales, M. K. Khan, and S. Goudarzi, "A secure trust model based on fuzzy logic in vehicular ad hoc networks with fog computing," *IEEE Access*, vol. 5, pp. 15619–15629, 2017.

[105] M. Al-khafajiy, T. Baker, C. Chalmers, M. Asim, H. Kolivand, M. Fahim, and A. Waraich, "Remote health monitoring of elderly through wearable sensors," *Multimedia Tools and Applications*, Jan 2019.

[106] W. Masri, I. Al Ridhawi, N. Mostafa, and P. Pourghomi, "Minimizing delay in iot systems through collaborative fog-to-fog (f2f) communication," in *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, pp. 1005–1010, IEEE, 2017.

[107] B. Ottenwälder, B. Koldehofe, K. Rothermel, and U. Ramachandran, "Migcep: Operator migration for mobility driven distributed complex event processing," in *Proceedings of the 7th ACM International Conference on Distributed Event-based Systems*, DEBS '13, (New York, NY, USA), pp. 183–194, ACM, 2013.

[108] S. Agarwal, S. Yadav, and A. Yadav, "An efficient architecture and algorithm for resource provisioning in fog computing," *International Journal of Information Engineering and Electronic Business*, vol. 8, pp. 48–61, 01 2016.

[109] A. Heil, M. Knoll, and T. Weis, "The internet of things-context-based device federations," in *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*, pp. 58–58, IEEE, 2007.

[110] W. Mathlouthi and N. B. B. Saoud, "Flexible composition of system of systems on cloud federation," in *2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)*, pp. 358–365, IEEE, 2017.

[111] J. Sun, S. Sun, K. Li, D. Liao, A. K. Sangaiah, and V. Chang, "Efficient algorithm for traffic engineering in cloud-of-things and edge computing," *Computers and Electrical Engineering*, vol. 69, pp. 610 – 627, 2018.

[112] L. Ni, J. Zhang, C. Jiang, C. Yan, and K. Yu, "Resource allocation strategy in fog computing based on priced timed petri nets," *IEEE Internet of Things Journal*, vol. 4, pp. 1216–1228, Oct 2017.

[113] F. Al-Turjman, "Cognitive caching for the future sensors in fog networking," *Pervasive and Mobile Computing*, vol. 42, pp. 317–334, 2017.

[114] F. Jalali, O. J. Smith, T. Lynar, and F. Suits, "Cognitive iot gateways: automatic task sharing and switching between cloud and edge/fog computing," in *Proceedings of the SIGCOMM Posters and Demos*, pp. 121–123, ACM, 2017.

[115] Y. Wang, Y. Xiang, J. Zhang, W. Zhou, G. Wei, and L. T. Yang, "Internet traffic classification using constrained clustering," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, pp. 2932–2943, Nov 2014.

[116] J. Ni, K. Zhang, X. Lin, and S. Shen, "Securing fog computing for internet of things applications: Challenges and solutions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 601–628, 2018.

[117] S. F. Abedin, M. G. R. Alam, N. H. Tran, and C. S. Hong, "A fog based system model for cooperative iot node pairing using matching theory," in *2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 309–314, Aug 2015.

[118] A. Alrawais, A. Alhothaily, C. Hu, and X. Cheng, "Fog computing for the internet of things: Security and privacy issues," *IEEE Internet ComputingMar*, vol. 1, no. 21, p. 2, 2017.

[119] T. Wang, G. Zhang, M. D. Z. A. Bhuiyan, A. Liu, W. Jia, and M. Xie, *A novel trust mechanism based on fog computing in sensor–cloud system.* Future Generation Computer Systems, 2018.

[120] A. M. Elmisery, S. Rho, and D. Botvich, "A fog based middleware for automated compliance with oecd privacy principles in internet of healthcare things," *IEEE Access*, vol. 4, pp. 8418–8441, 2016.

[121] S. A. Soleymani, A. H. Abdullah, M. Zareei, M. H. Anisi, C. Vargas-Rosales, M. K. Khan, and S. Goudarzi, "A secure trust model based on fuzzy logic in vehicular ad hoc networks with fog computing," *IEEE Access*, vol. 5, pp. 15619–15629, 2017.

[122] M. Henze, R. Hummen, R. Matzutt, and K. Wehrle, "A trust point-based security architecture for sensor data in the cloud," *In Trusted Cloud Computing*, pp. 77–106, 2014.

[123] L. Galluccio, S. Milardo, G. Morabito, and P. S. S. wise: Design, "Prototyping and experimentation of a stateful sdn solution for wireless sensor networks," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, pp. 513–521, IEEE, 2015.

[124] J.-H. Cho, A. Swami, and R. Chen, "A survey on trust management for mobile ad hoc networks," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 4, pp. 562–583, 2010.

[125] Q. Li, A. Malip, K. M. Martin, S.-L. Ng, and J. Zhang, "A reputation-based announcement scheme for vanets," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 9, pp. 4095–4108, 2012.

[126] R. Chen, F. Bao, M. Chang, and J.-H. Cho, "Dynamic trust management for delay tolerant networks and its application to secure routing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 5, pp. 1200–1210, 2013.

[127] J. Ren, Y. Zhang, K. Zhang, and S. X. S. Sacrm, "Social aware crowdsourcing with reputation management in mobile sensing," *Computer Communications*, vol. 65, pp. 55–65, 2015.

[128] K. Hwang, S. Kulkareni, and Y. Hu., "Cloud security with virtualized defense and reputation-based trust mangement," in *2009 Eighth IEEE International Conference on Dependable*, (IEEE), pp. 717–722, Autonomic and Secure Computing, 2009.

[129] J. Jiang, G. Han, F. Wang, L. Shu, and M. Guizani, "An efficient distributed trust model for wireless sensor networks," *IEEE transactions on parallel and distributed systems*, vol. 26, no. 5, pp. 1228–1237, 2015.

[130] Q. Fan and N. Ansari, "Towards workload balancing in fog computing empowered iot," *IEEE Transactions on Network Science and Engineering*, 2018.

[131] C.-H. Hong and B. Varghese, "Resource management in fog/edge computing: A survey," *arXiv preprint arXiv:1810.00305*, 2018.

[132] Q. Zhu, B. Si, F. Yang, and Y. Ma, "Task offloading decision in fog computing system," *China Communications*, vol. 14, pp. 59–68, Nov 2017.

[133] C. Fricker, F. Guillemin, P. Robert, and G. Thompson, "Analysis of an offloading scheme for data centers in the framework of fog computing," *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 1, pp. 16:1–16:18, Sept. 2016.

[134] T. He, E. N. Ciftcioglu, S. Wang, and K. S. Chan, "Location privacy in mobile edge clouds: A chaff-based approach," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2625–2636, 2017.

[135] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, 2018.

[136] N. K. Giang, M. Blackstock, R. Lea, and V. C. Leung, "Developing iot applications in the fog: A distributed dataflow approach," in *2015 5th International Conference on the Internet of Things (IOT)*, pp. 155–162, IEEE, 2015.

[137] S. Sarkar, S. Chatterjee, and S. Misra, "Assessment of the suitability of fog computing in the context of internet of things," *IEEE Transactions on Cloud Computing*, vol. 6, no. 1, pp. 46–59, 2015.

[138] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski, and P. Leitner, "Optimized iot service placement in the fog," *Service Oriented Computing and Applications*, vol. 11, pp. 427–443, Dec 2017.

[139] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[140] W. Shen, J. Yu, H. Xia, H. Zhang, X. Lu, and R. Hao, "Light-weight and privacy-preserving secure cloud auditing scheme for group users via the third party medium," *Journal of Network and Computer Applications*, vol. 82, pp. 56 – 64, 2017.

[141] Z. Wen, R. Yang, P. Garraghan, T. Lin, J. Xu, and M. Rovatsos, "Fog orchestration for internet of things services," *IEEE Internet Computing*, vol. 21, pp. 16–24, Mar 2017.

[142] P. Liu, L. Hartung, and S. Banerjee, "Lightweight multitenancy at the network's extreme edge," *Computer*, vol. 50, no. 10, pp. 50–57, 2017.

[143] K. Bhardwaj, J. C. Miranda, and A. Gavrilovska, "Towards iot-ddos prevention using edge computing," in *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*, (Boston, MA), USENIX Association, 2018.

[144] N. Wang, B. Varghese, M. Matthaiou, and D. S. Nikolopoulos, "Enorm: A framework for edge node resource management," *IEEE Transactions on Services Computing*, pp. 1–1, 2018.

[145] C. Vallati, A. Virdis, E. Mingozzi, and G. Stea, "Exploiting lte d2d communications in m2m fog platforms: Deployment and practical issues," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, pp. 585–590, Dec 2015.

[146] I. Azimi, A. Anzanpour, A. M. Rahmani, T. Pahikkala, M. Levorato, P. Liljeberg, and N. Dutt, "Hich: Hierarchical fog-assisted computing architecture for healthcare iot," *ACM Trans. Embed. Comput. Syst.*, vol. 16, pp. 174:1–174:20, Sept. 2017.

[147] E. K. Markakis, K. Karras, A. Sideris, G. Alexiou, and E. Pallis, "Computing, caching, and communication at the edge: The cornerstone for building a versatile 5g ecosystem," *IEEE Communications Magazine*, vol. 55, pp. 152–157, Nov 2017.

[148] L. Chen and J. Xu, "Socially trusted collaborative edge computing in ultra dense networks," in *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, SEC '17, (New York, NY, USA), pp. 9:1–9:11, ACM, 2017.

[149] J. Ni, K. Zhang, X. Lin, and X. S. Shen, "Securing fog computing for internet of things applications: Challenges and solutions," *IEEE Communications Surveys Tutorials*, vol. 20, pp. 601–628, Firstquarter 2018.

[150] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, and R. Ranjan, "Fog computing: Survey of trends, architectures, requirements, and research directions," *IEEE Access*, vol. 6, pp. 47980–48009, 2018.

[151] M. Iorga, L. Feldman, R. Barton, M. J. Martin, N. Goren, and C. Mahmoudi, "Fog computing conceptual model, recommendations of the national institute of standards and technology," *NIST Special publication*, pp. 500–325, 2018.

[152] V. Sarafov, "Comparison of iot data protocol overhead," *Proceedings of the Seminars of Future Internet (FI) and Innovative Internet Technologies and Mobile Communication (IITM)*, 2018.

[153] A. Sheth, "Internet of things to smart iot through semantic, cognitive, and perceptual computing," *IEEE Intelligent Systems*, vol. 31, no. 2, pp. 108–112, 2016.

[154] Logicworks, "Why Vendor Lock-In Remains a Big Roadblock to Cloud Success," September 2016 (checked out in April 2017). www.cloudcomputing-news.net/news/2016/sep/01/vendor-lock-in-is-big-roadblock-to-cloud-success-survey-finds.

[155] X. Masip-Bruin, E. Marín-Tordera, G. Tashakor, A. Jukan, and G. Ren, "Foggy Clouds and Cloudy Fogs: A Real Need for Coordinated Management of Fog-to-Cloud Computing Systems," *IEEE Wireless Communications*, vol. 5, no. 23, October 2016.

[156] A. Meola, "The Critical Role of Infrastructure in the Internet of Things," (last checked out October 2017) October 2016. uk.businessinsider.com/internet-of-things-infrastructure-architecture-management-2016-10.

[157] H. Atlam, R. Walters, and G. Wills, "Fog computing and the internet of things: a review," *Big Data and Cognitive Computing*, vol. 2, no. 2, p. 10, 2018.

[158] S. Kumar, "Ping attack–how bad is it?," *Computers & Security*, vol. 25, no. 5, pp. 332–337, 2006.

[159] K. Gao, Q. Wang, and L. Xi, "Controlling moving object in the internet of things," *International Journal of Advancements in Computing Technology*, vol. 4, pp. 83–90, 03 2012.

[160] M. J. Canet, V. Almenar, J. Marin-Roig, and J. Valls, "Time synchronization for the ieee 802.11a/g wlan standard," in *2007 IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 1–5, Sep. 2007.

[161] Y. Sahni, J. Cao, S. Zhang, and L. Yang, "Edge mesh: A new paradigm to enable distributed intelligence in internet of things," *IEEE Access*, vol. 5, pp. 16441–16458, 2017.

[162] M. A. Salahuddin, A. Al-Fuqaha, and M. Guizani, "Reinforcement learning for resource provisioning in the vehicular cloud," *IEEE Wireless Communications*, vol. 23, pp. 128–135, August 2016.

[163] G. Sun, L. Song, H. Yu, V. Chang, X. Du, and M. Guizani, "V2v routing in a vanet based on the autoregressive integrated moving average model," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 908–922, 2019.

[164] X. Jiang, H. S. Ghadikolaei, G. Fodor, E. Modiano, Z. Pang, M. Zorzi, and C. Fischione, "Low-latency networking: Where latency lurks and how to tame it," *Proceedings of the IEEE*, vol. 107, no. 2, 2019.

[165] A. Bozorgchenani, D. Tarchi, and G. E. Corazza, "An energy and delay-efficient partial offloading technique for fog computing architectures," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pp. 1–6, Dec 2017.

[166] T. Q. Dinh, J. Tang, Q. D. La, and T. Q. S. Quek, "Offloading in mobile edge computing: Task allocation and computational frequency scaling," *IEEE Transactions on Communications*, vol. 65, pp. 3571–3584, Aug 2017.

[167] M. Aazam, S. Zeadally, and K. A. Harras, "Offloading in fog computing for iot: Review, enabling technologies, and research opportunities," *Future Generation Computer Systems*, vol. 87, pp. 278–289, 2018.

[168] R. Saini and M. Khari, "Defining malicious behavior of a node and its defensive methods in ad hoc network," *International Journal of Computer ApplicationsApr;*, vol. 20, no. 4, pp. 18–21, 2011.

[169] A. Josang and R. Ismail, "The beta reputation system," in *In Proceedings of the 15th bled electronic commerce conference*, pp. 2502–2511, 5, Jun 17 2002.

[170] X. Yang, Y. Guo, Y. Liu, and H. Steck, "A survey of collaborative filtering based social recommender systems," *Computer Communications*, vol. 41, pp. 1–10, 2014.

[171] Y. Sahni, J. Cao, S. Zhang, and A. Yang L. Edge Mesh:, "new paradigm to enable distributed intelligence in internet of things," *IEEE access*, vol. 5, pp. 16441–58, 2017.

[172] A. Qureshi, *Power-demand routing in massive geo-distributed systems*. Massachusetts Institute of Technology, 2010. Doctoral dissertation.

[173] J. Son, *Integrated provisioning of compute and network resources in Software-Defined Cloud Data Centers*. PhD thesis, 2018.

[174] P. Delforge, "America's data centers consuming and wasting growing amounts of energy," *Natural Resource Defence Councle*, 2014.