# Improving Fog Computing Performance via $\mathcal{F}$og-2-$\mathcal{F}$og Collaboration

Mohammed Al-khafajiy[a], Thar Baker[a], Hilal Al-Libawy[b], Zakaria Maamar[c], Moayad Aloqaily[d], Yaser Jararweh[e]

[a] *Department of Computer Science, Liverpool John Moores University, Liverpool, UK*
[b] *Department of Electrical Engineering, University of Babylon, Babylon, Iraq*
[c] *College of Technological Innovation, Zayed University, Dubai, UAE*
[d] *Gnowit Inc., Ottawa, ON, Canada*
[e] *Jordan University of Science and Technology, Irbid, Jordan*

**Abstract**

In the Internet of Things (IoT) era, a large volume of data is continuously emitted from a plethora of connected devices. The current network paradigm, which relies on centralized data centers (*aka* Cloud computing), has become inefficient to respond to IoT latency concern. To address this concern, fog computing allows data processing and storage "close" to IoT devices. However, fog is still not efficient due to spatial and temporal distribution of these devices, which leads to fog nodes' unbalanced loads. This paper proposes a new $\mathcal{F}$og-2-$\mathcal{F}$og ($\mathcal{F}2\mathcal{F}$) collaboration model that promotes offloading incoming requests among fog nodes, according to their load and processing capabilities, via a novel load balancing known as Fog Resource manAgeMEnt Scheme (FRAMES). A formal mathematical model of $\mathcal{F}2\mathcal{F}$ and FRAMES has been fomulated, and a set of experiments has been carried out demonstrating the technical doability of $\mathcal{F}2\mathcal{F}$ collaboration. The performance of the proposed fog load balancing model is compared to other load balancing models.

*Keywords:* Internet-of-Things, Fog computing, $\mathcal{F}$og-2-$\mathcal{F}$og collaboration, Offloading

## 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), Software Defined Network (SDN), and Data Center Network (DCN). ICN shifted inter-networking to a cloud-based computing model, as reported in Cisco Cloud Index (2013-2018) [1]. Since most Internet traffic originates and/or terminates to/from the cloud [1] [2], 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] [3]. 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 of 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] [5]. The inclination to use the clouds coincides with the improvement of Network Function Virtualisation (NFV) technique which reduces cloud CAPital EXPenditures (CAPEX) and OPerating EXpenditures (OPEX), and improving the flexibility and scalability of an entire network [6]. Similarly, NFV has been used to address the problem of deploying replica servers in clouds by proposing an algorithm based on spectral

*Corresponding author: Thar Baker
*Email addresses:* M.D.Alkhafajiy@2016.ljmu.ac.uk (Mohammed Al-khafajiy), T.Baker@ljmu.ac.uk (Thar Baker), eng.hilal_al-libawy@uobabylon.edu.iq (Hilal Al-Libawy), zakaria.maamar@zu.ac.ae (Zakaria Maamar), Moayad@gnowit.com (Moayad Aloqaily), YiJararweh@just.edu.jo (Yaser Jararweh)

clustering theory [7]. The performance of the proposed solution on large-scale setup has shown lower deployment costs and improved data fusion.

Within the emerging Internet of Things (IoT), a large number of "smart" devices and objects (e.g., wearable) are, nowadays, connected to the Internet [8], generating high volumes of data every second. In the IoT area, the word *"thing"* could be anything refers to everything that can connects to a network and exchanges data over this network [9] with other stakeholders (e.g., users, applications, and peers). Cisco IBSG estimates that approximately 50 billion devices (i.e., things) will be connected to IoT networks by 2022 [10] [11]. Although cloud can provision efficient data storage and processing facilities, the ever-growing volume of data will result in a high energy consumption, heavy burden on the communication bandwidth [1] [12], and "unacceptable" latency [13] [4] [14]. In addition, since the cloud is relatively "far" from IoT devices, by the time the data reaches the cloud for storage and/or processing, its importance and freshness could deprecate [15] [16]. To address cloud limitations with focus on latency in IoT, Cisco has come up with the concept of *Fog* in 2014 [1]. Security of mobile edge and fog is one of the major challenges for successful implementation and deployment of IoT infrastructure. For example, researchers in [17] [18] [19] explore how to protect critical system (i.e., fog and edge) from unauthenticated or malicious attacks. However, the security aspect is part of our future work and has not been considered in the current work.

Simply put, fog is described as a highly virtualised platform that provides similar cloud facilities in terms of storage, processing, and communications at the edge of the network, "closer" to things compared to cloud; i.e., between things and clouds [20], making these facilities fast, secure, and reliable [21] [15]. Fog is not a substitute to cloud but a complement [1] since both are expected to work together [2] [22]. 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 and online gaming) [13]; (ii) geo-distributed services (e.g., pipeline monitoring) [23]; (iii) mobile services with high speed connectivity (e.g., connected vehicles) [24]; and (iv) large scale distributed control systems (e.g., smart energy distribution and smart traffic lights) [1]. Despite the appealing benefits of fog computing, some concerns are undermining its adoption. This includes how to specify $\mathcal{C}$loud-2-$\mathcal{F}$og ($\mathcal{C}2\mathcal{F}$) collaboration and $\mathcal{F}$og-2-$\mathcal{F}$og ($\mathcal{F}2\mathcal{F}$) collaboration. This paper addresses $\mathcal{F}2\mathcal{F}$ by promoting service offloading among fog nodes so, that, minimal latency for IoT services is achieved.

### *1.1. Problem Statement*

Although $fog$ nodes are placed "closer" to IoT things so, that, latency is "taken care" of [13] [25], these nodes can quickly become congested when the number of requests soliciting their services exceed their capabilities [13] [24]. OpenFog [26] 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 [27] [28].

### *1.2. Research Contributions*

Our contribution is threefold:

1. $\mathcal{F}2\mathcal{F}$ collaboration model that achieves a near optimal workload among the collaborating fog nodes (Section 3.1).
2. 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 [29] where each fog node hosts local computation, networking and storage capabilities. (Section 3.2).
3. A formal mathematical model that backs the decision of load balancing among fog nodes (Section 4).

### *1.3. Paper Organization*

The remainder of this paper is organized as follows. Section 2 discusses the related work of fog computing and services offloading. Section 3, describes the system architecture and management of fog nodes alongside our offloading approach in detail. Section 4 presents system design and modelling. Extensive simulation results and evaluations are presented in Section 5. Section 6 concludes the paper and discusses future work.

## 2. Related Work

Current research efforts into fog tackle the following challenges:

- Heterogeneity: fog nodes are diverse in terms of processing, storage and communication capabilities.
- Elasticity: the ever-growing number of IoT devices would trigger fog elastic.
- Federation: despite fog elasticity, making fogs collaborate is an option, when this elasticity becomes insufficient. However, heterogeneity is an obstacle to their collaboration.

Agarwal et al. [30] focus on resource allocation in a fog context. They propose a 3-layer architecture, client, fog, and cloud, that allows to distribute the 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. Agarwal et al.'s work tackles well the heterogeneity challenge but neither scalability nor federation challenge are tackled.

Beate et al. [31] 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 work does not consider offloading load among fogs; fogs are assumed able to perform computationally intensive tasks, which might not always be the case. In addition, with regard to the above challenges, the authors indicate that the approach can be employed at large scale in real world so, that, scalability is met. However, they seem overlooking heterogeneity and federation. Vehicular ad-hoc networks (VANET) have reached the maturity stage in term of communication reliability with the benefits of information transmission between vehicles and surrounding fog and edge nodes [32] [33]. The quality of communication between vehicles and those nodes is being investigated. Many researchers have found that physical factors can impact this QoS especially in the presence of buildings and other obstacles (i.e., signal fading). A candidate solution is to utilize parked vehicles for routing communication [34].

A collaborative resource sharing and utilization among fog/edge was introduced recently with very promising solution employing 5g and composition techniques [35] [36]. Abedin et al. [35] propose resource sharing among fog nodes by defining a utility metric for these nodes that accounts the communication benefits in case resources are shared among them. First, the authors determine an organised list of preferences that paires fog nodes for each node. Then, each node in the fog places a request of pairing with its preferred pairing nodes. At the reception side, depending on the preference and benefits of the previously received requests, a target 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 Quality-of-Service (QoS) (such as latency and bandwidth) in 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.

Lina et al. [37] 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. While the authors meet federation, scalability and heterogeneity are not met.

Sun et al. [38] propose Cloud-of-Things and Edge Computing (CoTEC) scheme for traffic management in multi-domain networks. CoTEC direct 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 regard to the above challenges, the federation is only met with edge-cloud, however the congestion of clouds or edges not discussed. Also, they seem overlooking heterogeneity (i.e., device's capacity) and scalability.

Despite all efforts mentioned beforehand, and to the best of our knowledge, a systematic framework for $\mathcal{F}2\mathcal{F}$ collaboration that tackles heterogeneity, elasticity, and federation is still absent. This paper serves as a starting point for defining such a collaboration model.

## 3. $\mathcal{F}$og-2-$\mathcal{F}$og Collaboration Model

Before we dive into the proposed $\mathcal{F}2\mathcal{F}$ collaboration model, we highlight the adapted fog computing architecture. This architecture is similar to other large-scale computing architectures (e.g., cloud computing) have either application specific architecture or application agnostic architecture. However, there is no a standard architecture for the systems that use fog computing [1] [12] [39]. In this paper, we adopt a general fog computing architecture that is in-line with the architectures presented in [1] [2] [4] [12] [13]. Understanding the fog computing architecture topology helps obtain a better insight into the main functionality and benefits of using fog computing. The main layers in fog architecture are *thing*, $fog$, and *cloud* (Figure 1).

**Thing layer:** also called *perception* layer, ensures data availability by hosting networked devices (e.g., heart-rate and blood-oxygen sensors). Each device has a communication protocol (such as IEEE 802.15.4, WiFi, Bluetooth, MQTT, etc.) so, that, it transmits the generated data to the fog layer in the form of data processing service request.

**Fog layer:** contains a number of distributed fog nodes that should ideally be located "next" to data sources. This layer refines and processes the data that the things layer submits. Fog has the potential to reduce the amount of things' data transmitted to the cloud layer by acting on these data. Each fog node is equipped with onboard computational resources, data storage, alongside network communication facilities to bridge things and cloud within the IoT network [13].

**Cloud layer:** enables omnipresent, convenient, and proper network access to shared resources (e.g., storage and services processing) over the IoT network.

In the following, we define an IoT network as $\{T, F, C, L\}$, *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[1] generated from the thing (e.g., *heavy-packet* and *light-packet*), and $d$ denotes the destination from $t_n$ to the

[1]A payload size of 1024 Bytes can be transported without any fragmentation through a normal not constrained network; otherwise, it is fragmented into lighter tasks [40]

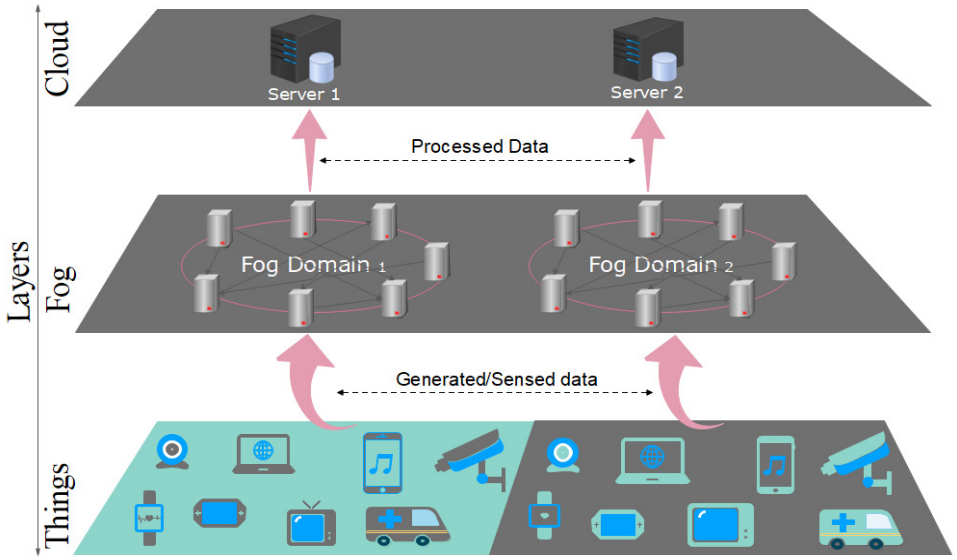



Figure 1: IoT fog architecture composed of *things*, $fog$, and *cloud* layers

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 $t_n$'s location.

- $F$ 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), $\ell$ denotes a fog node location, $s$ and $\hbar$ refer to services (e.g., image processing) provided by the fog node and hardware capability (i.e., CPU frequency) of the fog node, respectively, and $r$ is a set of all "reachable fogs"[2] from $f_i$.

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

- L is set of communication links among the layers, such that, $L \subseteq \{\langle n, \grave{n}, q\rangle | (n, \grave{n}) \in (T,T)(T,F)(F,F)(T,C)(F,C)(C,C)(C,F)(F,T)(C,T) \bigwedge (q \in Q)\}$.
  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).

The standardised approach in which IoT systems (with a fog layer) operate is as follow: $t_n$ generates and gathers data periodically from the surroundings and sends it to either the fog layer or the cloud layer for processing/manipulation. In the fog layer, $f_i$ can serve $t_n$'s request instantly or offload it to another fog node (e.g., $f_j$) in the same domain if $f_i$ is congested and may delay processing $t_n$'s request. To this end, $f_i$ (or $f_j$) 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 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 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 services with high QoS. However, this paper's focus is only on processing service's requests dispatched from the things layer to fog layer in which the latter layer adopts the proposed $\mathcal{F}2\mathcal{F}$ collaboration for efficient

[2] we focus on the processing jobs at the fog layer; therefore, "reachable" clouds are out of the scope of this work.

data processing. The following sub-sections present the fog load balance by answering when and where to offload a request, and then, discuss the fog resource management scheme.

### *3.1. Fog load 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. There are two approaches to model interactions among fog nodes. First, the centralised model, which relies on a central node that controls the offload interaction among the fog nodes. Second, in which each fog node runs 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. We envisage that the distributed model is more suitable for scenarios where things are mobile objects (i.e., Internet of moving things [41]) as to support the mobility and flexibility of data acquisition. Therefore, we adopt this model of interactions in our $\mathcal{F}2\mathcal{F}$ collaboration model.

#### *3.1.1. When to offload a request?*

The decision of a fog node to support the processing of a received service's 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's request travel time (minimal latency always preferable). The procedure of offloading a received request by a fog is as follows: once a service request(s) is received by the fog node, it checks the request payload 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 collaboration 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 neglectable in the overall service latency as the link rate among fogs is usually around 100 Mbps [13], which is very high. The collaboration 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 cloud in case no fogs are available to handle the service. Simply put, offloading happens when a fog node has heavy load. In the other extreme case when all fog nodes have heavy load, offloading becomes useless. Thereof, it is more effective when there is a high load variance among participant nodes.

#### *3.1.2. Where to offload a request?*

Each fog has a list of best-suitable nodes with whom it can collaborate (i.e., reachability features table including the estimated computing and response time), when needed. This list is generated based on nodes' locations and their neighbouring nodes. When a node is about to get or become congested[3], it can share the load with nodes from the list based on the payload size received. The list of best neighbouring nodes is maintained periodically by each node. Fog node location's privacy and fog services location's privacy are important in the network [42]. However, privacy does not fall into the scope of the current work and it will be handled as future work.

The best node selection and offloading algorithms are explained in Section 4.6. 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.

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

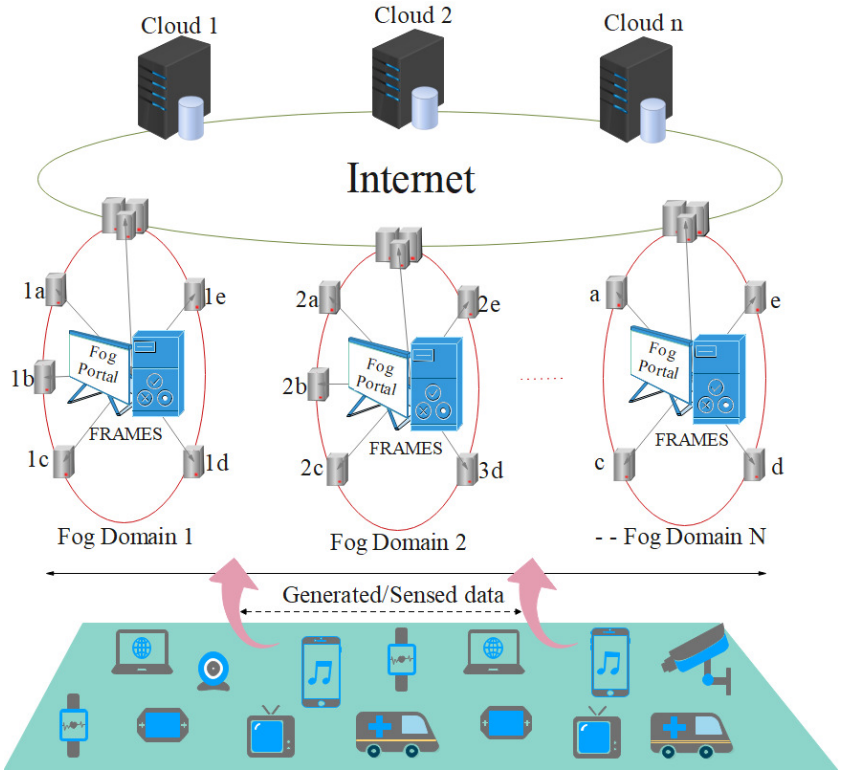



Figure 2: Overview of the Fog Resource manAgeMEnt Scheme

Therefore, the list should be updated on the following offloading occasions: (i) when the fog sends request of status updates to other nodes; (ii) adding a new node to the fog domain; (iii) removing a node from the fog domain; and (iv) node goes off-line. These interactions and management are handled by the FRAMES, which is proposed and described Section 3.2.

### *3.2. Fog resource management scheme*

The fog layer in the IoT architecture consists of heterogeneous devices clustered together and forming what so called fog domains. Each fog device has it is own coverage range where the desired fog services are provided. In fact, due to node heterogeneity, service's types and sizes (e.g., processing speed and storage capacity) vary from one fog node to another. This section discusses FRAMES, which involves managing fog resources status and provides network analysis and statistics for fog resource provisioning and consumption. Figure 2 shows the conceptual diagram of FRAMES. The main functionality is to periodically monitor fogs' statuses and network loads.

FRAMES is based on the fog node distribution architecture [2] [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 3.

- *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. The procedure of declaring new/existing fog services via the fog portal starts when the fog owner connects the actual fog

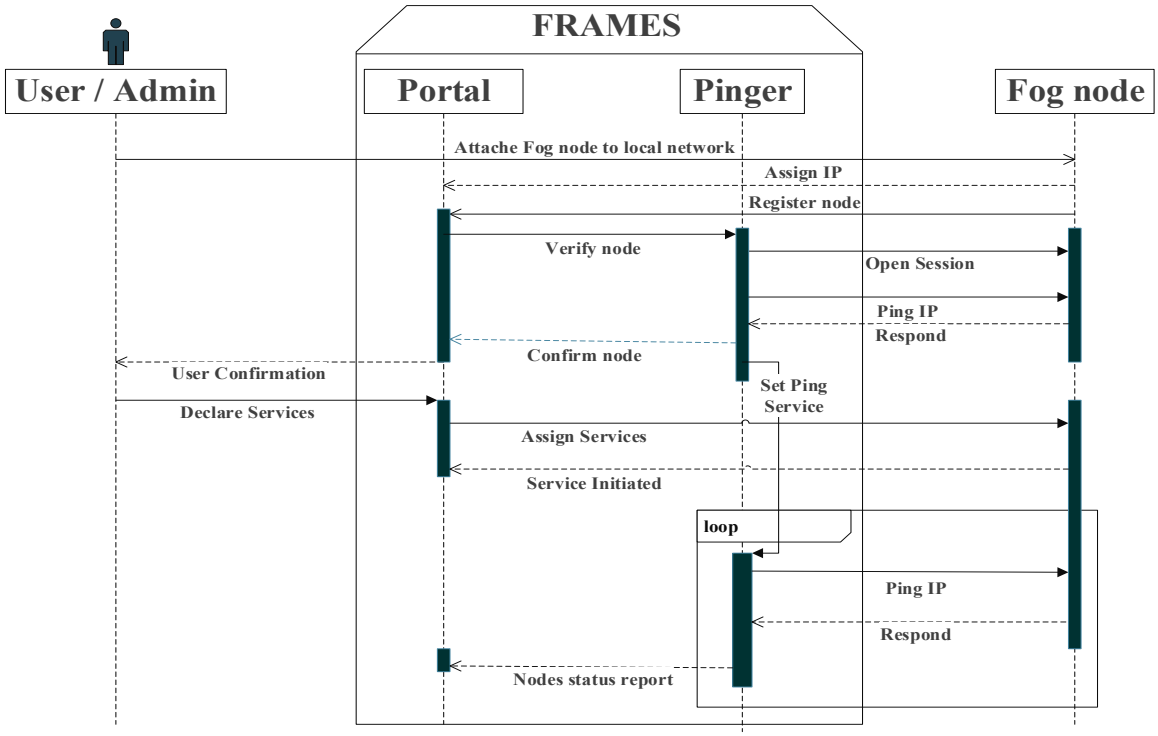



Figure 3: Sequence diagram showing FRAMES interactions

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 unique static IP address to the device, and at this point the node will ping the portal to register device 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.

- *Fog Pinger:* is an automated ping process, which runs by FRAMES on a periodic schedule 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 are taken in case of fog node is down.

- *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 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.

## 4. $\mathcal{F}$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 paper are given in Table 1.

### *4.1. Network Model*

Communication between nodes in the context of $\mathcal{F}2\mathcal{F}$ coordination is modelled as an undirected graph with all nodes are reachable from each other. Having $G = \langle N, L, W \rangle$, where $N$ is a set of thing,

Table 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 |
| $\lambda$ | service arrival rate to fog layer |
| $\mu$ | fog node service rate |
| $\rho_i^F$ | probability of sending the request to the fog |
| $\rho_i^C$ | probability of sending the request directly to the cloud |
| $\rho_i^I$ | 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 |
| $\tau_s$ | total time required to process a service |
| $\tau_{que}$ | is the queuing time |
| $\tau_{proc}$ | service processing time |
| $\rho$ | system usage |
| $\varrho_{size}$ | queue size |
| $\tau_{que}^{si}$ | 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 |

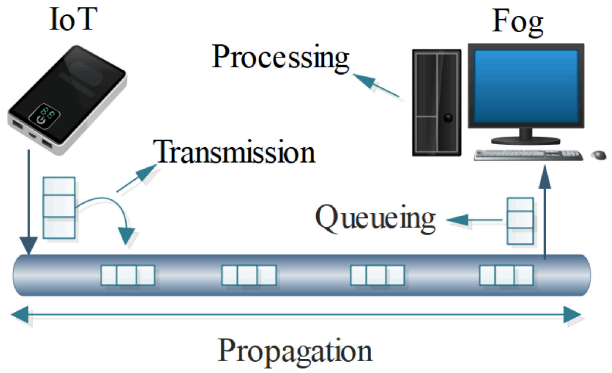



Figure 4: Four sources could delay service processing

fog, and cloud nodes. Thus, $G = N^I \cup N^F \cup N^C$; $L$ denotes the set of communication links between the nodes; $W$ is the set of edge weights between nodes, according to the distance between them; the longest the distance, the highest the weight is. Thus, $D_p$ is dependent on the edge weight between two nodes.

### *4.2. Service Delay*

A request can be defined as a set of tasks that is processed completely to meet the desired service's requirements. Processing a request (i.e., service) can happen over any of the 3 layers (i.e., thing, fog, and cloud). Hereafter, we calculate the total delay taken to process a service. Service delay ($S_d$) for $t_n$ request is expressed as follows:

$$\begin{aligned} S_d = & \rho_i^I * S_p^I \\ & + \rho_i^F * [D_t^F + D_p^F + D_c^F] \\ & + \rho_i^C * [D_t^C + D_p^C + D_c^C] \end{aligned} \tag{1}$$

Where $\rho_i^I$ is the probability that $t_n$ processes the data locally at the things layer, $\rho_i^F$ is the probability of processing the service at the fog layer, and $\rho_i^C$ 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 (see Figure 4). Similarly, $D_p^C$ is propagation delay for cloud server, and $D_t^C$ is the sum of all transmission delays to the cloud. Figure 4 shows 4 delay sources that could impact services and cause latency. To correctly calculate the delay, we should be clear about where the service will be processed and what parameters are involved in the processing. Therefore, our focus is on minimising service processing latency over the fog layer, via $\mathcal{F}2\mathcal{F}$ collaboration, based on service transmission delay ($D_t$), propagation delay ($D_p$), and computational delay ($D_c$) which includes both queuing delay ($D_{que}$) and processing delay $D_{proc}$).

#### *4.2.1. Transmission Delay*

$D_t$ is the time taken by a sender 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 $l_p$ in bits and data rate (i.e., upload bandwidth) $b \uparrow$. Thus, the sum of transmission delay $D_t^n$ for $t_n$ is calculated using Equation 2.

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

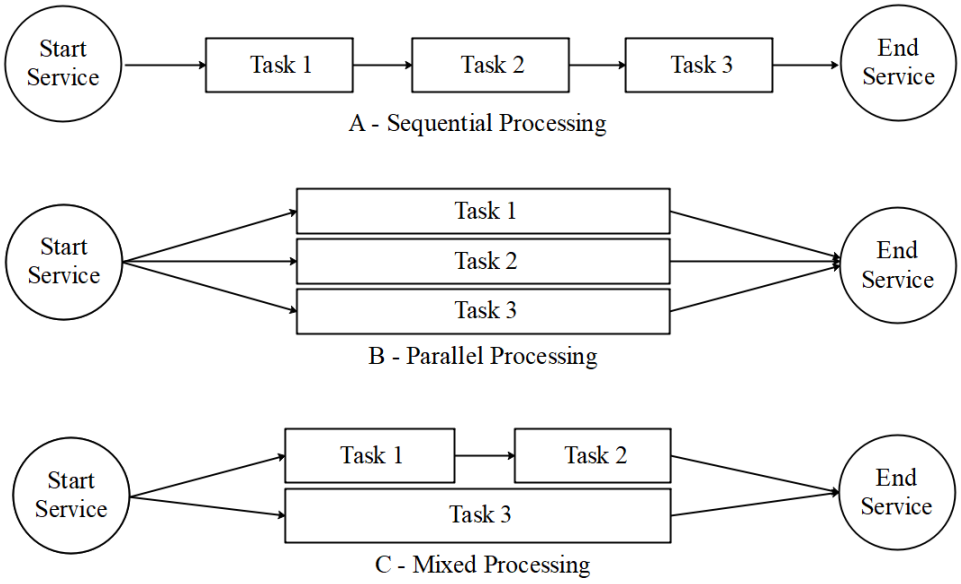



Figure 5: Three types of service processing

$$D_t^n = \sum \frac{l_p^n}{b\uparrow} \tag{2}$$

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

*4.2.2. 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 (calculated using the latitude and longitude of the thing and fog) to destination $l_d$ and propagation speed $p_s$. Thus, $D_p^n$ for $t_n$ is calculated using Equation 3.

$$D_p^n = \frac{l_d^n}{p_s} \tag{3}$$

The propagation delays between other layers, such as fog to cloud, are calculated using the same approach in Equation 3 based on $l_d$, and $p_s$.

*4.2.3. Computational Delay*

This 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 ($D_{proc}$). $D_{que}$ is the period of time spent by a packet inside the queue/buffer until it is served by the fog. And, processing delay is the time consumed to process the received data/packet(s) by the fog. 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 mix. Figure 5 demonstrates the different possible approaches for processing a service. For a service with sequential task the process delay is the sum of all tasks delay, while the process delay for a parallel processing will be the maximum latencies among all tasks. Therefore, the proceeding delay for a service that can be processed immediately without waiting in the queue will be calculated using Equation 4.

$$D_{proc}^s = max_{q \to Q_s}(\sum_{t \in Q_s^q} d_{f_i^c}^{ts}), \forall q \in Q, \forall c \in C \tag{4}$$

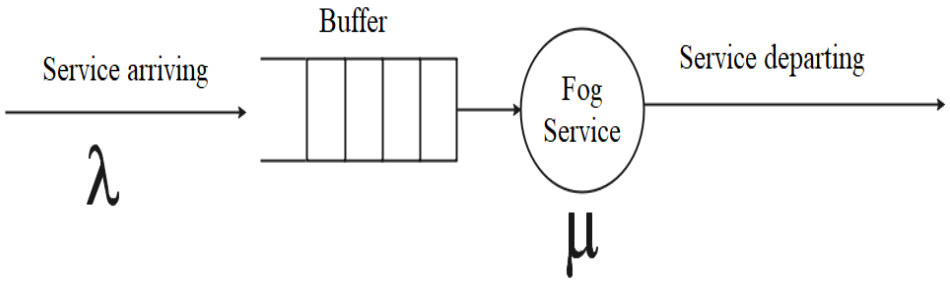



Figure 6: Queuing system

Where $d_{f_i^c}^{ts}$ 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 4 is used to calculate the total time-delay when a service is immediately processed by a fog. Next we discuss the scenario when the service arrives to 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 arrived services are queued in the fog buffer until the fog becomes available to process the service. 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 queue's, in addition, the processing time for the service as per Figure 6, where $\lambda$ is the average service arrival rate and $\mu$ is the average serving rate for a fog.

In a queuing system a network can be modelled by three parameters $A/P/n$, i.e., Erlang-C [1], where $A$ is service arrival rate, $P$ is the service time probability density, and $n$ the number of fog nodes. Therefore, we model our system as $M/M/n$. where the first $M$ is the services arrival rate according to Poisson process with average rate $\lambda_i$ 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 our 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 offloaded to the first node that becomes available through a periodic checking for the reachability table within the fog. The total time for a service $\tau_s = \tau_{que}^s + \tau_{pro}$ where $\tau_{proc}$ is service processing time, and $\tau_{que}$ is the queuing time. Thus, the total time for $\tau_s$ can be compute by Equation 5.

$$\tau_s = \left[ \sum_{x=0}^{n-1} \frac{(\frac{n}{x})!(1-\rho^2)}{(n\rho)^{n-x}} + \frac{1-\rho}{\rho} \right]^{-1} \tag{5}$$

where $\rho$ is the system utilisation, obtained using Equation 6.

$$\rho = \frac{arrivalRate}{serviceRate} = \sum_{x=1}^{n} \frac{\lambda}{\mu_x} \tag{6}$$

The $\mu$ 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 to note that 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's packets in the queue we use Equation 7:

$$\varrho_{size} = \frac{\rho(\frac{[P_s^w(n,\rho)](n\rho)^n}{n!(1-\rho)})}{1-\rho} \tag{7}$$

Where $P_s$ is the probability of number of services packets in the fog system and calculated using

Equation 8:

$$P_s^w(n,\rho) = \left[\sum_{x=0}^{n-1} \frac{(n\rho)^x}{x!} + \frac{(n\rho)^n}{n!(1-\rho)}\right]^{-1} \tag{8}$$

Equation 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 9.

$$P_s^d = 1 - (P_s(n,\rho)) \tag{9}$$

Next, we calculate the average delay for a service's packets in a fog's queue. This will help evaluate the performance of fog by the FRAMES and point out the congested node based on $\varrho_{size}$ and queuing time for a process $\tau_{que}$. Thus, the queuing time for a service computes is calculated using Equation 10.

$$\tau_{que}^{si} = \frac{\rho(\frac{P_s^w(n\rho)^n}{n!(1-\rho)})}{\lambda - \lambda\rho} \tag{10}$$

Where $\tau_{que}^{si}$ is the queuing time $\tau_{que}$ for service $s$ at the resources of fog node $i$, $\lambda$ is the service rate and $\rho$ is the system utilisation. To compute the total time for a service in our system, its by adding the processing delay to $\tau_{que}^{s}$ as per Equation 11.

$$\tau_c^{si} = \tau_{que}^{si} + \frac{1}{\mu} \tag{11}$$

*4.3. Fog Workload*

Fog workload $f_w$ refers to the overall usage of a fog node's CPU (*cycles/second*), which is consumed during the processing of a particular service. Thus, there are constraints on a node's capability. This leads to a limitation on the ability of processing different types 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 $f_i^c$.

$$f_w \leq f_i^c, \forall f \in F \tag{12}$$

A service that operates on a fog node can serve several end-users in the network. Thus, the total ratio of CPU usage by the service (or services in case of parallel processing) should not exceed the allocated resources allocated for a service because the allocated resources are considered to be the total $f_w$ can be handled by a fog. Equation 13 computes the total resources ($rs$) allocated to process all tasks $ts$ for a service $s$.

$$f_{rs}^s = s_w = \sum_{t=1}^{n} C_{ts}^{f_i}, \lceil s \rceil \leq f_c^i, \forall s \in S, \forall t \in T_s \tag{13}$$

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 13, where $C_{ts}^{f_i}$ 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, we have separate between services workloads depending on the type of service's request type, having a heavy-weight and low-weight service's request depends on a service's packets size. For instance, when a service only processes small data from sensors, this will consume low computational power, thus, the workload on fog is low. While, in services that performs heavy real-time video processing, the workload will be high on the fog node. Therefore, services workload ($s_w$) on fogs can vary for each service, depends on service's type. The $f_w$ for all services is the sum of each

service workload multipled by $\lambda$ as per Equation 14. Thus, $f_w$ should be less than the $f_c$ assignment variable (i.e., $f_w <= f_c$).

$$f_w = \sum_{x=1}^{n} s_x^w.\lambda_s, \forall s \in S \tag{14}$$

*4.4. 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 for incoming packets. Thus, the services arrival traffic $\lambda$ to fogs according to Poisson and processing rate of fog is exponentially distributed over fogs according to *light-services* processing ($\mu$) and *heavy-services* processing rate ($\mu'$). To compute the waiting for a service on a specific fog node, it will be through calculating the total time for processing the current heavy and light services in the fog queue. For instance, to obtain the average waiting time for a service $s$ that arrived at $f_i$ it will be done through the total time consumed by $f_i$ to process all current services. Having $nS_h$ refer to the number of heavy-services and $nS_l$ refer to the number of light-services, Equation 15 computes the average waiting time for a newly arrived service on $f_i$:

$$nS_h = \sum s_h^{f_i}, \forall s_h \in S$$

$$nS_l = \sum s_l^{f_i}, \forall s_l \in S$$

$$\tau_{s_w}^{f_i} = \frac{nS_h}{\mu'} + \frac{nS_l}{\mu} \tag{15}$$

It is worth to mention that, if $f_i$ queue is not empty (i.e., we have $nS_h + nS_l \neq \phi$), then we have $(nS_h + nS_l - 1)$ mix types of services in the queue and one service is currently in processing.

*4.5. 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 4 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 ($\tau_{que}$) until it gets processed. The delay for processing the service ($\tau_{proc}$) and the time to respond back ($\tau_{\downarrow}$) to $t_n$ is also added to the total latency for the service as per Equation 16. For simplification, we assume that($\tau_{\uparrow}$=$\tau_{\downarrow}$), having ([$\tau_{\uparrow}$=$\tau_{\downarrow}$]=2$\tau_{\downarrow\uparrow}$) because logically the returned packet contents normally is similar or smaller than the sent packet.

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

$$\tau_s = \tau_{que}^s + \tau_{proc} + 2\tau_{\downarrow\uparrow}, \forall s \in S \tag{16}$$

We address the problem of having an optimal workload on fog nodes alongside with achieving minimal delay for IoT services. Thus, achieving 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 services results within the required period, i.e., before service deadline ($s_d$) with the desired QoS. Therefore, the research problem can be defined as follows:

$$P: \quad min[\tau_s] \leqslant s_d, \forall s \in S \tag{17}$$

$$\text{s.t.} \quad f_c^{min} \leqslant f_w \leqslant f_c^{max} \tag{18}$$

$$\sum \lambda_s \leqslant \sum \mu_f \tag{19}$$

$$P_s^d(n,p) \geqslant serviceLevel \tag{20}$$

$$\lambda_s \xrightarrow{min[D_p]} f_i \tag{21}$$

$$\tau_s \leqslant s_d, \forall s \in S \tag{22}$$

The constraints on this research are to reduce service latency. Therefore, our constraints are written with focus on achieving minimal service delay. In constraint (18), we indicate 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 (19) imposes that the total traffic arrival rate ($\lambda_s$) to a fog domain should not exceed the service rate ($\mu_f$) of this fog domain. Constraint (20) imposes the probability of directly processed services should be greater or equal to the desired service level. Constraint (21) impose the first destination for the IoT services traffic generated at the IoT Things layer 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 (22) is strictly bound by the service time $\tau_s$ that should be within the limit of service deadline $s_d$.

### *4.6. Offloading Model*

The offloading model proposes to balance the load within the fog domain by distributing service traffics from the congested fog nodes to other fogs within the domain. To balance services traffic in fogs domain, we assume that fogs at any giving location are reachable to each other within the same fog domain as per our network model in Section 4.1, which models the fog network as a mesh network; this assumption in line with the work in [24] and [43]. In this research, we consider a real-world scenario of services flows where services arrival rates can significantly vary from one fog node to another [24] depending on fog location, since we have $\lambda_s \xrightarrow{min[D_p]} f_i$ constraint that is services are directed to nearest fog from thing for processing. For instance, in Figure 7, we demonstrate the scenario where fogs can vary in their traffic load due to their geo-location. In a similar scenario, offloading the traffic from loaded to idle fog can be crucial to mitigate the load and keep the service latency to the minimal. For example, given that only mobile vehicles are considered in traditional VANET, the authors in [34] 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, 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 system discussed in [44], the vehicular communications prove reducing the traffic congestions and, hence, the Round-trip Delay Time (RDT) thereof cutting down the fuel consumption (money variable). Another good example happens regularly in the financial market where having low-latency between placing an order and executing the order is crucial to get the order at the favourite price.

The decision factors where a node is congested and offloading is required significantly on fog workload ($f_w$), which is associated with the service traffic arrival rate ($\lambda_s$) and total processing rate (i.e., service rate $\mu$) which is down to fog CPU frequency (i.e., node capability). In addition, service processing time $\tau_s$, which ideally should not exceed service deadline ($s_d$). Therefore, to make the decision of offloading by a fog is when $\tau_s > s_d$, as per Probability 23, having ($O_s$) for offloading service decision:

$$O_s = \begin{cases} 1, & \text{if } \tau_s > s_d \\ 0, & \text{otherwise} \end{cases} \tag{23}$$

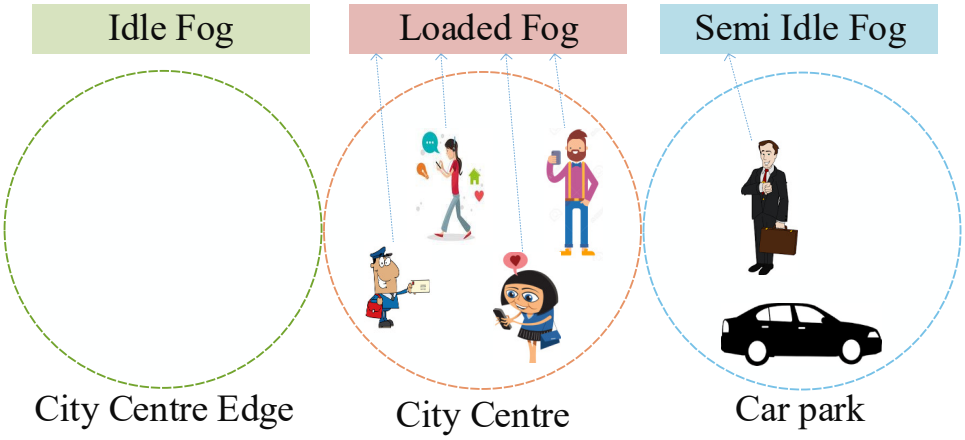



Figure 7: Loaded, idle and, semi-idle fog nodes based on $\lambda_s \xrightarrow{min[D_p]} f_i$

Thus:

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

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

In Probability 23, $O_s$ value is set to either 0 or 1, where 0 refers to no offload is required and 1 refers that the newly arrived service will suffer form latency and will not be able to meet the service deadline $s_d$. Hence, service offloading is required.

Algorithm 1 has been developed to detect the fog nodes that suffer from congestion issue, and determining the overload. 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 $\lambda$ is bigger than the outcome of the fog node $\mu$. If any of the conditions is satisfied, a flag indicates that the node is congested as per line 9. The second part of the algorithm (Procedure 2) determines the overload by computing the number of services causing the congestion as per lines 24-26. The overload $O_l$ will hold the list of services that requires offloading to other nodes as per lines 27-28.

To balance the services on fog nodes and achieve optimal workload and minimal service delay, we adopt the offloading to the best available node that 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 $\tau_s$ for the services required offloading among all available nodes using Equation 24, thus, having some constraints on the node that participate in the process to handle the overload such as load limit.

$$min[\tau_s] = min \sum_{i=1}^{n} [\tau^{f_i}_{que} + \tau^{f_i}_{proc} + \tau_{\downarrow}] \tag{24}$$

s.t.

$$f_c^{min} \leqslant f_w \leqslant f_c^{max}$$

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

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

The best available nodes are those that provide a service with minimal delay. Algorithm 2 finds the best node to handle the overload on the congested node, and than offload the overload from the

**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** $(\tau_c^{si} \geq S_d) \;||\; \lambda \geq \mu)$ **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** $(O_s == 1 \;||\; \lambda \geq \mu)$ **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**

congested 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

```
1  Procedure 1. Determine best available node by
2      F_L = list{φ}                                   ▷ F_L initiate fog list
3      F_L = list[F_n] ⟵ getFogNodes(out : (F_n, F_c))
4      F_L = sort(F_L, by F_c DESC)
5      for each F_n ∈ F_L do
6          if Fn ⟵ (F_l ≥ Fc_max) then
7              F_L = pop(Fn)                           ▷ remove busy node
8          else
9              τ_s = Σ_{i=1}^{n} [τ^i_que + τ^i_pro + τ↓]
10             if (τ_s < s_d) then
11                 list.add(F_n, τ_s)
12                 continue
13             else
14                 F_L = pop(Fn)
15             end
16         end
17     end
18     return F_L
19 End
20 Procedure 2. Handover the Overload by
21     if F_L ≠ φ then
22         F_n = min[F_L(τ_s, D_p)])
23         F^n_l = F_l + O_l
24     else
25         goto:1
26     end
27 End
```

The first part of the algorithm (Procedure 1), shows the process of finding the best available node(s) for handling the overload pointed in Algorithm 1. Lines 2-3 of the algorithm initiate the list of active fogs in the domain alongside with the node's capacity and current load (i.e., queue size). The list of available 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 to run on each of the available nodes. If the time is within the limit allowed for the service (i.e, before $S_d$), the system will be keep the node in the list and log the expected service time against the node as per lines 9-12. If the $\tau_s$ on $F_n$ is less than $S_d$, than $F_n$ will be removed from the list as per lines 13-15. The second part of the algorithm (Procedure 2), receives the list of best available nodes. If the list is not empty, that means there is at least one fog able to take the overload. However, if there is more than one node in the list, the system will direct the overload to a node that can provide minimal $\tau_s$ and has the lowest propagation delay $D_p$ as per lines 21-23.

## 5. Evaluation

In this section, we evaluate through MATLAB-based simulation the $\mathcal{F}2\mathcal{F}$ model, which is about providing optimal fog workload with minimal latency for IoT services. A scientific and comprehensive network latency [45] has been calculated (i.e., latency of heavy packets, light packets, mixed types of packets and latency per node) to demonstrate the superior performance of the proposed $\mathcal{F}2\mathcal{F}$ model. We validated the results against 2 benchmark algorithms: Random Walk Algorithm (RWA) [46] [47], and Neighbouring Fogs Algorithm (NFA) [48]. Simulation settings are presented in the following subsection followed by a discussion of the simulations results.

### *5.1. Simulation Settings*

In this section, we describe the settings we adopted during the simulations. We specify the network topology, propagation and transmission delay, with link bandwidth and fog nodes capabilities, as follows:

- Network topology modelled as an indirect graph represents fog mesh network at the fog layer. 15 fog nodes were considered ($f_n = 15$) and connected together through internal communication link. 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 one second, then the link weight between both nodes is ($fog_1 \underset{1}{\longleftrightarrow} fog_2$). Similarly, the services arrived to the fog layer are assigned to fog based on the smallest $D_p$ between the node and source, which has the smallest distance.

- Network bandwidth depends on the type of service. For light-packets (e.g., data packets from sensors) the communication bandwidth is 250 $Kbps$ [49], which is equivalent to $2.0 \times 10^6$ $hertz$. The communication protocol is IEEE 802.15.4, and ZigBee. For heavy-packets (e.g., data packets from cameras) the communication bandwidth is 54 $Mbps$ [50], which is equivalent to $4.3 \times 10^8$ $hertz$ [50]. The communication protocol is IEEE 802.11a/g. The transmission rate between the fog nodes is expected to be higher $\simeq 100$ $Mbps$ [13].

- Transmission and propagation delays, the transmission delay $D_t$ for a packet is dependent on the packet size $l_p$ alongside with the associated upload bandwidth $b\uparrow$. Therefore, we 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 $KB$ [13]. With regard to the propagation delay $D_p$, we use packet round trip time (i.e., $\tau_{\downarrow\uparrow}$) same as in [13] by $\tau_{\downarrow\uparrow} = 0.03 \times l_d + 5$, where $l_d$ is the distance with unit $km$, and $\tau_{\downarrow\uparrow}$ time unit is $ms$.

- Fog node capabilities consider the service rate $\mu$ that varies from one node to another. Fog node's capability determinants by CPU frequency, therefore nodes are variant in CPU and the rang is between 0.2 GHz to 1.5 GHz [51], which includes the CPU frequency of CISCO fog device.

### *5.2. Benchmarking*

We considered 2 algorithms for benchmarking purposes:

1. Random Walk Algorithm (RWA) [46] [47], which imposes that arriving service requests are assigned to a nearest fog node from source; if fog is congested it will offload the service randomly to another fog node. In this scenario, we assume that each fog node within the domain has the same probability of being selected.
2. Neighbouring Fogs Algorithm (NFA) [48], which is imposes that congested fog will offload the overload to the nearest fog node with bigger capacity.

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

*5.3. Results and Discussion*

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

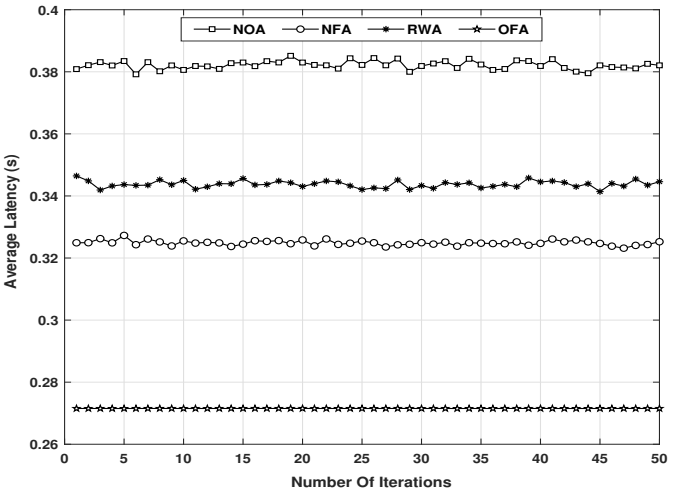
Figure 8 illustrates the performance of our OFA based on the average response time for all received services according to a service's packets type. We also show a comparison between results of OFA and the results obtained from other algorithms mentioned in Section 5.2. The simulation settings for this experiment is as follow:

- Fog nodes with different capabilities, hence, nodes vary in their service rate $\mu$.
- Fog nodes capability based on CPU frequency with a minimum of $200 \times 10^6$ hertz, incremented by 100 hertz until it get to maximum CPU capability of $15 \times 10^8$.
- Service arrival rate $\lambda = 3 \times 10^2$ packet per second as in [24], and $\lambda$ is fixed during the experiment to ensure all algorithms have the same traffic arrival rate.

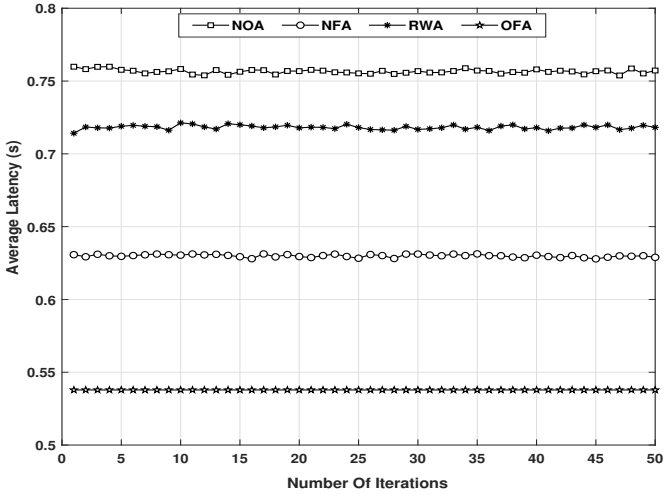
Figures 8a, 8b, and 8c are grouped by packet type (heavy-packets versus light-packets). In Figure 8a, the packets type is mixed (MTP), having a random number of heavy and light packets. However, the random number is fixed through out the experiment to ensure consistency across all algorithms. In Figures 8b and 8c, 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 8 the vertical line represent the average latency per algorithm to serve all arrived services, and the horizontal line is the number of iterations carried out to insure 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. NOA also has the lowest service time because it does not consider offloading when a 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 hither than our OFA. However, RWA has the worst performance with MTP and AHP as it randomly offloads the overload, which is relatively blind algorithm as it not considered the current load ($f_w$) and propagation delay ($D_p$).

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 $200 \times 10^6$ hertz, incremented by 100 hertz until it gets to maximum CPU capability of $15 \times 10^8$ (i.e, $F_n = 14$). In this simulation, we increment the service arrival rate so, that, the total packet received is 1 million service as in [13]. The packet type in this experiment is mixed with a random number of heavy-packets and light-packets. Figure 9 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 with NFA and RWA, on the other hand, OFA has the lowest average latency between nodes 1 to 7, but greater average latency from node 8, and thereafter. However, the average latency differences is much higher for NFA and RWA in comparison to OFA for 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 nodes considers the overall capacity and current load before 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.

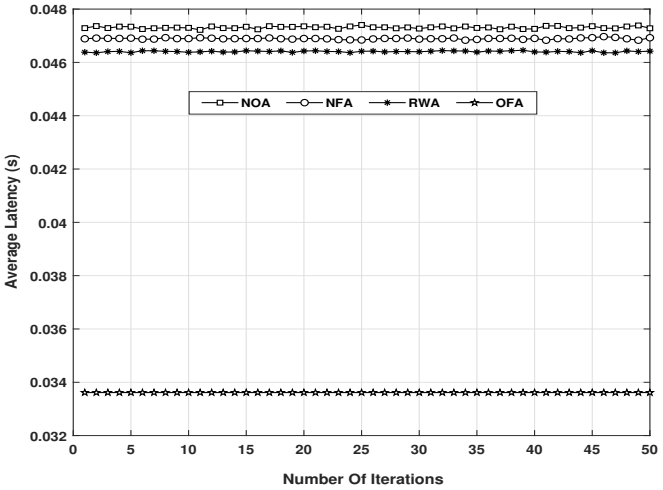
To prove the optimal distribution of packet with OFA we run a new experiment and recall the settings from the previous experiment. However, in this experiment, the vertical line represents service usage (i.e., number of packets) as per Figure 10. The nodes are sorted from smallest capacity (i.e., lowest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




(a) Mixed types of packets (MTP)




(b) All heavy packets (AHP)




(c) All light packets (ALP)

Figure 8: Average latency according to offloading model

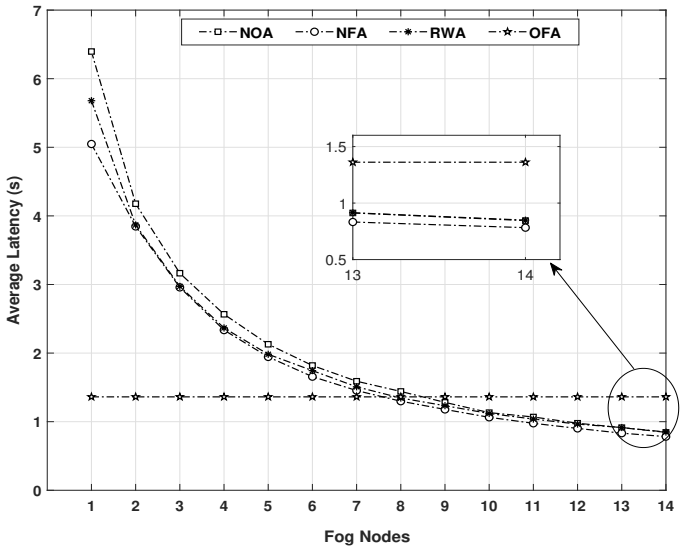



Figure 9: Average latency per node

packet distribution on average is steady among all nodes regardless of the node capacity, which causes the issue of latency as, on the one hand, nodes with small CPU frequency consumes significant time to process all received packets. While, on the other hand, nodes with large CPU frequency have already finished processing the received packets as per Figure 9.

Figure 11 shows the impact of increasing the number of packets on latency. During simulation, service's packets are varied from one packet to $10 \times 10^4$ packets in Figure 11a; thus, the packet type is fixed to heavy-packet for consistency. The service utilisation rate is 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 $\lambda$) will increase the overall latency. The total latency and performance of the algorithms vary; OFA has the lowest service latency as per Figure 11a. The service latency is stable with small delay of approximately 0.6 second for the received packets upto $6.5 \times 10^4$, 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 $10 \times 10^4$ packet. Moreover, in Figure 11b, we have increased the packets utilisation to $10 \times 10^6$ to show the continues of latency variations for the different algorithms compared to OFA. It is clear that OFA has a sustainable packets processing with the increases of service's packets (i.e., high traffic), in term of latency, as it has the lowest packet's latencies.

From previous simulations we increase the packet arrival rate $\lambda$ to $15 \times 10^4$ 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 12 shows the maximum and average latencies for the $15 \times 10^4$ packets (with type heavy) based on the offloading algorithms. In comparison between the maximum latencies for all offloading algorithms in Figure 11 and 12, it is clear that the increment of maximum latency for NFA, NOA, and RWA is significantly more than the maximum latency for OFA, as in Figure 11 the maximum latency for a packet with OFA is around 0.6 second, and in Figure 12 the maximum latency is 0.8 second. Whereas, within NFA and RWA the maximum latency is 1.5 and 2 seconds, respectively, in Figure 11, while the maximum latency is 2.7 and 3.2 seconds, respectively, in Figure 12. It is clear that the latency incremented by approximately more than 1 second with NFA and RWA offloading algorithm, while the latency increment for OFA is only 0.2 second.

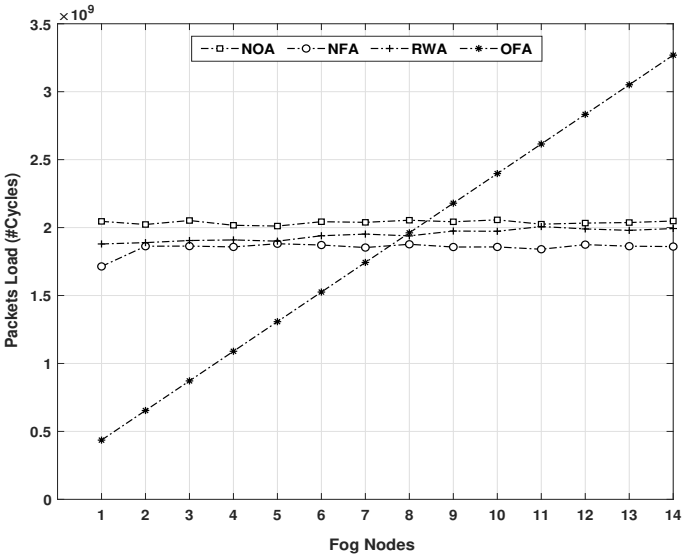



Figure 10: Average load on nodes

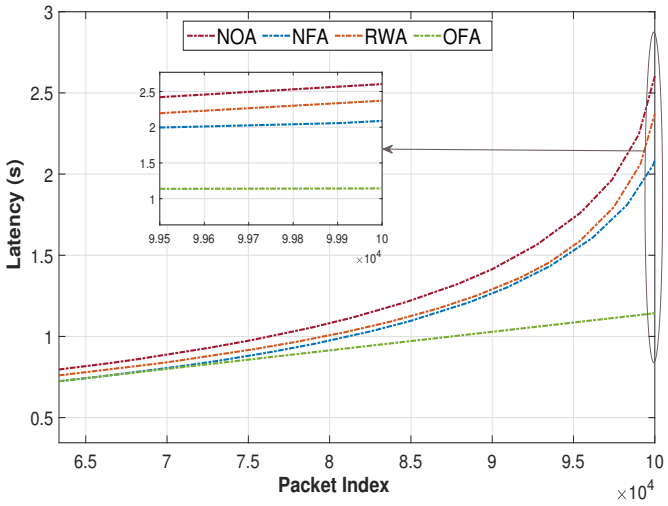
## 6. Conclusions and Future Works

Although fog computing is recognized as a computing model that suits IoT, it is still not widely used. Due to the spatial and temporal dynamics of IoT things distribution, the computations loads on fogs significantly vary. Thus, nodes could be lightly loaded, while others are not causing fog congestion then, latency. In this paper, we propose a novel Fog Resource manAgeMEnt Scheme (FRAMES) that justifies fog distributions and management with service load allocation and computing load allocation via service offloading among fog nodes to achieve minimal latency for IoT services. We propose a feasible solution that enables fog traffic management via service offloading in fog-based architecture that serves the purpose of minimizing the average response time for real-time IoT services. Through the extensive experiments, it is clear that the offloading significantly impacts the overall latency of services. A proper resources managements with accurate offloading algorithm will significantly reduce service latency and, the number of fog nodes and it's capacities will also impact service latency. Moreover, it is clear that the proposed OFA has the lowest service response time in comparison with RWA and NFA. OFA and have the potential in achieving sustainable network paradigm to highlights the significance and benefits of adopting fog computing paradigm. In our future work, we plan to extend the simulation using a data processing techniques (such as MapReduce or Spark) with a real dataset to evaluate the $\mathcal{F}$og-2-$\mathcal{F}$og model. In addition, we plan to study security issues that associate with malicious fogs since it has been considered as a critical infrastructure. Thus, develop a secure fog layer that defend users and services privacy, also perform security tests to understand the robustness against malicious attacks, particularly in the $\mathcal{F}$og-2-$\mathcal{F}$og model.

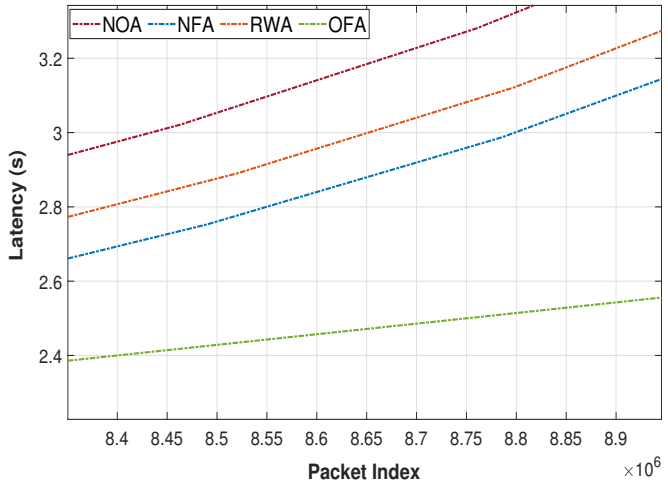
[1] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang. Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption. *IEEE Internet of Things Journal*, 3(6):1171–1181, Dec 2016.

[2] Mohammed Al-khafajiy, 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*, ICFNDS '18, pages 9:1–9:7, New York, NY, USA, 2018. ACM.

[3] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18, May 2010.




(a) Packet index upto $10^4$




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

Figure 11: Latency per packet

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

[5] Lizhe Wang, Gregor von Laszewski, Andrew Younge, Xi He, Marcel Kunze, Jie Tao, and Cheng Fu. Cloud computing: a perspective study. *New Generation Computing*, 28(2):137–146, Apr 2010.

[6] 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*, 6:18238–18250, 2018.

[7] Gang Sun, Victor Chang, Guanghua Yang, and Dan Liao. The cost-efficient deployment of replica servers in virtual content distribution networks for data fusion. *Information Sciences*, 432:495–515, 2018.

[8] Friedemann Mattern and Christian Floerkemeier. *From the Internet of Computers to the Internet of Things*, pages 242–259. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

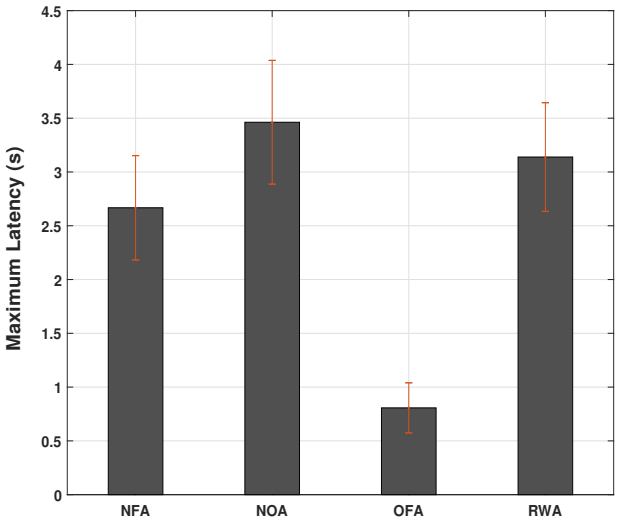



Figure 12: Maximum latency upon heavy-packets


[9] 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)*, pages 359–364, Dec 2018.

[10] Cisco Systems. Fog computing and the internet of things: Extend the cloud to where the things are, 2016.

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

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

[13] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue. On reducing iot service delay via fog offloading. *IEEE Internet of Things Journal*, 5(2):998–1010, April 2018.

[14] Saeed Khanagha, H.W. Volberda, and Ilan Oshri. Business model renewal and ambidexterity: Structural alteration and strategy formation process during transition to a cloud business model. *R and D Management*, 44, 06 2014.

[15] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh 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, pages 13–16, New York, NY, USA, 2012. ACM.

[16] 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)*, pages 71–77, Sep. 2018.

[17] Amandeep Singh Sohal, Rajinder Sandhu, Sandeep K Sood, and Victor Chang. A cybersecurity framework to identify malicious edge device in fog computing and cloud-of-things environments. *Computers & Security*, 74:340–354, 2018.

[18] Safa Otoum, Burak Kantarci, and Hussein Mouftah. Adaptively supervised and intrusion-aware data aggregation for wireless sensor clusters in critical infrastructures. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2018.

[19] Safa Otoum, Burak Kantarci, and Hussein T Mouftah. Hierarchical trust-based black-hole detection in wsn-based smart grid monitoring. In *Communications (ICC), 2017 IEEE International Conference on*, pages 1–6. IEEE, 2017.

[20] Kai Kang, Wang Cong, and Tao Luo. Fog computing for vehicular ad-hoc networks: Paradigms, scenarios, and issues. *The Journal of China Universities of Posts and Telecommunications*, 23:56–96, 04 2016.

[21] 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)*, pages 561–566, Dec 2016.

[22] 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.

[23] Zakaria Maamar, Thar Baker, Noura Faci, Emir Ugljanin, Yacine Atif, Mohammed Al-Khafajiy, and Mohamed Sellami. Cognitive computing meets the internet of things. In *Proceedings of the 13th International Conference on Software Technologies :*, pages 741–746, 2018.

[24] 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*, 14(10):4568–4578, Oct 2018.

[25] 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*, 4(2):36–45, March 2017.

[26] OpenFog Consortium Architecture Working 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.

[27] S. Ningning, G. Chao, A. Xingshuo, and Z. Qiang. Fog computing dynamic load balancing mechanism based on graph repartitioning. *China Communications*, 13(3):156–164, March 2016.

[28] 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*, 20(1):416–464, Firstquarter 2018.

[29] Hany Atlam, Robert Walters, and Gary Wills. Fog computing and the internet of things: a review. *Big Data and Cognitive Computing*, 2(2):10, 2018.

[30] Swati Agarwal, Shashank Yadav, and Arun Yadav. An efficient architecture and algorithm for resource provisioning in fog computing. *International Journal of Information Engineering and Electronic Business*, 8:48–61, 01 2016.

[31] Beate Ottenwälder, Boris Koldehofe, Kurt Rothermel, and Umakishore 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, pages 183–194, New York, NY, USA, 2013. ACM.

[32] Moayad Aloqaily, Ala Abu Alkheir, and H.T. Mouftah. Connected and autonomous electric vehicles (caevs): A service management perspective. *IT Professional*, 20, 12 2018.

[33] Moayad Aloqaily, Safa Otoum, Ismaeel Al Ridhawi, and Yaser Jararweh. An intrusion detection system for connected vehicles in smart cities. *Ad Hoc Networks*, 2019.

[34] Gang Sun, Liangjun Song, Hongfang Yu, Victor Chang, Xiaojiang Du, and Mohsen Guizani. V2v routing in a vanet based on the autoregressive integrated moving average model. *IEEE Transactions on Vehicular Technology*, 68(1):908–922, 2019.

[35] 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)*, pages 309–314, Aug 2015.

[36] Ismaeel Al Ridhawi, Moayad Aloqaily, Yehia Kotb, Yousif Al Ridhawi, and Yaser Jararweh. A collaborative mobile edge computing and user solution for service composition in 5g systems. *Transactions on Emerging Telecommunications Technologies*, 29, 06 2018.

[37] 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*, 4(5):1216–1228, Oct 2017.

[38] Jian Sun, Siyu Sun, Ke Li, Dan Liao, Arun Kumar Sangaiah, and Victor Chang. Efficient algorithm for traffic engineering in cloud-of-things and edge computing. *Computers and Electrical Engineering*, 69:610 – 627, 2018.

[39] Ranesh Kumar Naha, Saurabh Garg, Dimitrios Georgakopoulos, Prem Prakash Jayaraman, Longxiang Gao, Yong Xiang, and Rajiv Ranjan. Fog computing: Survey of trends, architectures, requirements, and research directions. *IEEE Access*, 6:47980–48009, 2018.

[40] 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.

[41] K Gao, Q Wang, and Lifeng Xi. Controlling moving object in the internet of things. *International Journal of Advancements in Computing Technology*, 4:83–90, 03 2012.

[42] Gang Sun, Victor Chang, Muthu Ramachandran, Zhili Sun, Gangmin Li, Hongfang Yu, and Dan Liao. Efficient location privacy algorithm for internet of things (iot) services and applications. *Journal of Network and Computer Applications*, 89:3 – 13, 2017. Emerging Services for Internet of Things (IoT).

[43] M. A. Salahuddin, A. Al-Fuqaha, and M. Guizani. Reinforcement learning for resource provisioning in the vehicular cloud. *IEEE Wireless Communications*, 23(4):128–135, August 2016.

[44] 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*, 107(2), 2019.

[45] V. Chang and G. Wills. A model to compare cloud and non-cloud storage of big data. *Future Generation Computer Systems*, 57:56–76, 2016.

[46] Q. Zhu, B. Si, F. Yang, and Y. Ma. Task offloading decision in fog computing system. *China Communications*, 14(11):59–68, Nov 2017.

[47] Christine Fricker, Fabrice Guillemin, Philippe Robert, and Guilherme Thompson. Analysis of an offloading scheme for data centers in the framework of fog computing. *ACM Trans. Model. Perform. Eval. Comput. Syst.*, 1(4):16:1–16:18, September 2016.

[48] 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*, pages 1–6, Dec 2017.

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

[50] 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*, pages 1–5, Sep. 2007.

[51] 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*, 65(8):3571–3584, Aug 2017.