Article

Fog Computing: Towards Dynamically Controlling the Offloading Threshold and Managing Fog Resources in Online Dynamic Systems

Faten Alenizi [1] and Omer F Rana [2]

[1] School of Computer Science and Informatics, Cardiff University, Cardiff, UK; AleniziF@cardiff.ac.uk
[2] School of Computer Science and Informatics, Cardiff University, Cardiff, UK; RanaOF@cardiff.ac.uk

Abstract: Fog computing is a potential solution to overcome the shortcomings of the cloud computing processing of IoT tasks. These drawbacks can be high latency, location awareness and security, and it is attributed to the distance between IoT devices and servers, network congestion and other variables. Although fog computing has evolved as a solution to these challenges, it is known for having limited resources that need to be consciously utilised, or any of its advantages would be lost. Computational offloading and resource management are critical concerns to be considered to get maximum benefit of the available resource at fog computing systems and benefit from its advantages. Computational offloading and resource management are important issues to be considered to get maximum benefit of the available resource at fog computing systems and benefit from its advantages. In this article, in vehicular traffic applications, we introduce a dynamic online offloading scheme that involves the execution of delay-sensitive activities. This paper proposes an architecture of a fog node that enables a fog node to adjust its offloading threshold dynamically (i.e., the criteria by which a fog node decides whether tasks should be offloaded rather than executed locally) using two algorithms: dynamic task scheduling (DTS) and dynamic energy control (DEC). These algorithms seek to solve an optimisation problem aimed at minimising overall delay, improving throughput, and minimising energy consumption at the fog layer, while maximising the use of resource-constrained fog nodes. Compared with other benchmarks, our approach can reduce the delay by up to 95.38% and reduce energy consumption by up to 67.71% in fog nodes. Additionally, this approach enhances throughput by 71.08%.

Keywords: Fog Computing; Computational Offloading; Dynamic Offloading Threshold; Resource Management; Minimising delay; Minimising energy consumption; Maximising throughputs.

1. Introduction

The number of IoT devices and their generated tasks are constantly growing, imposing a burden on the cloud infrastructure, in particular by processing these tasks within their Quality of Services (QoS) [1, 2]. The processing of these tasks in the cloud can trigger systems to suffer high communication latency, security issues, network congestion and poor QoS [3]. This is due to the gap between the IoT devices and the cloud server, and the cloud infrastructure as it is geographically centralised [4, 5]. Fog computing has emerged to address limitation of processing IoT tasks at the cloud and ensuring the processing of these tasks while meeting their QoS specifications [6].

Fog computing is an intermediate layer situated between cloud and IoT, that brings location awareness, low latency, and wide-spread geographical distribution for the IoT [7, 8]. It consists of limited-resources devices called fog nodes, which bring storage, processing, and networking resources close to IoT devices where tasks are produced [8, 9]. Fog computing made its first appearance in 2012, after being launched by Cisco [4, 10]. While Fog computing is an exciting technology for addressing the shortcomings of cloud, there are still issues yet to be addressed [10]. With limited-resources devices at the fog systems, poor utilisation of these resources would ruin the benefits of utilising fog systems and add further disadvantages rather than advantages. This reflects on the significance of tackling the problem of computational offloading in combination with resource management approaches to help leverage these tools in a sustainable technical manner [1].

Computational offloading enables workload/computational tasks to be shared among the system entities including IoT devices, fog nodes, and cloud servers [11-14]. When computational offloading occurs between fog nodes, this is called fog cooperation [15], in which overloaded fog nodes send part of their workload to other underloaded fog nodes so that they can meet their necessary QoS while still operating [16, 17]. Resource management can be discussed by multiple factors, saving energy consumption in the fog environment is one of these factors , and it has considered in the present article. Integrating computational offloading and resource management is essential to effectively utilizing fog resources and to achieve high efficiency [1].

In online dynamic fog systems, where uncertainties are imposed by multiple factors, with no prior awareness of system details such as task arrival rate, the number of connected IoT devices, and computational capacity of fog nodes, addressing computational offloading and resource management is challenging to obtain optimum outcomes [1]. To the best of the authors' knowledge, the topic of computational offloading has mostly been explored in offline fog systems, where all system data are known beforehand, and few studies have considered the online dynamic fog systems. Furthermore, it is challenging to manage fog resources in the online dynamic fog systems compared to the offline system. Additionally, no exiting work has investigated the impact of dynamically changing the offloading threshold, which is a factor that determines when a fog node begins sharing its workload with other neighbouring fog nodes within its proximity.

1.1 Contributions

This work provides the following contributions:

- We propose a fog node architecture that dynamically decides whether to process the received tasks locally or offload them to other neighbours. This is based on a dynamic threshold that considers the queuing delay of the primary fog node and the availability of its neighbours.
- The problem of computational offloading along with resource management has been investigated using an online dynamic system with the aim to solve an optimisation problem that aims to minimise delay, minimise energy consumption and maximise throughput.
- We conduct extensive experiments to evaluate the performance of our proposed scheme, and compared our proposed algorithm to various benchmarks.

This paper extends our previous work [1] by introducing a dynamic offloading threshold, made use of in an online model for evaluating service delay.

1.2 Paper Organization

The remainder of this paper is organized as follows. We overview the related works in section 2 and present the system modelling and constraints in Section 3. In Section 4, we decompose the problem into two sub-problems: the delay minimisation problem and the energy saving problem. In Section 5, we discuss the proposed scheme. In Section 6, we numerically examine the performance of our proposed scheme against other benchmarks. Finally, in section 7 we conclude this article.

2. Related Work

This section is divided into three main parts. The first focuses on computational offloading between entities within a specific system; the second addresses the impact of dynamically managing servers to enhance power efficiency. Finally, a comparison of the state-of-the-art is provided and summarised in Table 1.

2.1. Computational Offloading

Computational offloading can be implemented offline or online. In offline implementation, All the system information needed to make the offloading decision is previously known and is based on historical or predictive knowledge, such as the computational capabilities of fog nodes, the total number of IoT devices and their total workload (number of requests). This is applied during the system design stage. In online deployment, the computational offloading decision takes place at run-time and considers the current system status and process characteristics, such as the current waiting time and the current available computational resources, without prior knowledge of the system inputs that are taken into account in the offline deployment. Several studies were conducted to examine the issue of computational offloading in its offline deployment, such as [8, 11-14, 18-23].. In [19], Wang et al. investigated the optimisation offloading problem aiming to minimise task completion time given tolerable delay and energy constraints. The optimisation problem has been formulated as a mixed integer nonlinear programming problem that jointly optimises the local computation capability for IoT devices, the computing resource allocation of fog nodes and the offloading decision. To solve the problem authors decomposed it into two independent sub problems to find the optimal amount of workload that should be processed locally at IoT devices and at fog nodes. Following that a hybrid genetic simulated annealing-based algorithm has been developed to optimise the offloading decision. Based on simulation results, the proposed solutions achieve remarkable results in minimizing computation tasks delay and optimising local computation capability, the computing resource allocation of fog nodes and the offloading decision. Tang et al. [18] aimed to increase the total number of executed tasks on IoT devices and fog nodes under deadline and energy constraints. The authors identified the issue as a decentralised, partially observable offloading optimisation problem in which end users are partially aware of their local system status, including the current number of remaining tasks, the current battery power and the nearest fog node resource available. Such parameters are used to assess if tasks should be processed locally or offloaded to the nearest fog node. Their solution helps the IoT devices to make an appropriate decision based on its locally observed system.

Liu et al, [11] addressed a multi-objective optimisation offloading problem at fog environment with the aim of minimising execution delay, energy consumed at mobile devices and offloading payment cost for using fog/cloud resources. The multi-objective problem has been formulated into a single problem using Scalarization method. The proposed solution finds the optimal offloading probability that accomplish the stated objectives. Mukherjee et al, [20] designed an offloading technique focusing on jointly optimising the computing and communication resources at the fog systems to reduce end-to-end latency. Their technique considers the trade-off between transmission delay and task execution delay when making the offloading decision, in which a fog node can seek additional computational resources from either one of its neighbours or the cloud datacentre to reduce task execution delay at the expense of the transmission delay. The optimisation problem has been transformed into convex Quadratically Constraint Quadratic Programming and solved using CVX toolbar which is a MATLAB-based modelling system for convex optimization. Their simulation results demonstrated that their proposed solution offers minimal end-to-end latency in comparison to executing all tasks at end-user devices and executing all tasks at the primary fog nodes.

Zhu et al [13] proposed task offloading policy based on execution time and energy consumption. This approach helps mobile devices to make an appropriate decision on whether to process their tasks locally or offloading them to the appropriate fog node or the cloud. During the decision-making procedure, mobile devices calculate both the execution time and the energy consumed when executing the task in its device and compared that with the execution time and the energy consumed when offloading and receiving the processed task on the appropriate fog node, the energy consumed when executing the tasks on fog nodes are not considered. Based on this comparison, the IoT device selects the decision with the least cost (execution time plus energy consumption). Comparing their scheme to Random, no offloading, and only offloading when considering only execution time, their simulation results show an optimisation of the execution time of tasks and energy consumption of mobile devices. Mukherjee et al, [21] formulated the offloading problem as an optimization problem with the goal to minimize the total system cost which is the sum of the total delay of end-users' tasks and the total energy consumed at end-users devices that are consumed due to local processing tasks and uploading tasks to the fog environment for processing. Under delay and energy constraint, the optimisation problem has been transformed into a Quadratically Constraint Quadratic Programming problem and solved by semidefinite relaxation method. Within a heterogeneous environment where fog nodes have different computational resources, the proposed solution helped to define the optimal amount of workload that should be processed at end-users' devices, primary fog nodes, neighbouring fog nodes, and cloud servers. The decision on when to offload depends entirely on the availability of computational resources. The authors stated that having higher computational resources at fog nodes helps to reduce the system cost. Also, as increasing the number of end-users per fog node in the systems, fog nodes become more congested as thus preferring to send their workload to the cloud server for processing rather than the neighbouring fog nodes.

Chen and Hao [14] studied offloading problem in dense software-defined networks. The paper describes the problem as a mixed-integer nonlinear problem that is decomposed into two sub-problems; First, deciding whether the task is processed locally at the end-user device or offloaded to the edge device. Second, determining the computational resources that are dedicated to each task. The authors developed an efficient software-defined task offloading scheme to solve the sub-problems. The results of their proposed scheme, demonstrate the superiority of their approach at decreasing the end user device's energy consumption and overall task execution latency. In IoT-Fog-Cloud architecture, Sun et al, [22] presented an ETCORA algorithm that consists of two parts. The first part considers computation offloading selection, which aims to find the optimal offloading decision based on minimising time and energy and the second part optimises the resource allocation in terms of transmission power allocation. Their proposed solution helps to minimise energy consumption and completion time of tasks compared to other schemes. Zhao et al. [12] investigated the computational offloading problem in the context of radio access networks to reduce the weighted sum of total offloading latency plus total energy consumption. To improve the offloading decision and enhance the allocation of computation and radio resources, the authors have formulated the problem as a non-linear, non-convex joint optimisation problem. Their proposed solution is more effective than mobile cloud computing (MCC), which processes all end-user tasks on a cloud server, and mobile edge computing (MEC), which processes all end-user tasks in the edge computing system. The reason their approach is more effective compared to MCC and MEC is that it makes use of a combination of the available resources at the cloud and fog, compared to the only cloud as in MCC and only edge as in MEC.

Hybrid-Computational offloading optimization problem has been investigated by Meng et al., [23] where two types of models are considered; namely cloud computational offloading and fog computational offloading. The authors aimed to minimise the consumption of energy caused by transmitting and processing tasks at mobile terminals, fog, and cloud servers under deadline constraints. The authors introduced a new concept called computation energy efficiency that defined as “the amount of the computation tasks that are offloaded by consuming a unit of energy”, to solve the optimisation problem, which further divides the problem into four sub-problems. Based on the proposed solution that considers offloading tasks to fog and cloud servers for execution, simulation results show the effectiveness of the solution compared to only offloading tasks to the fog model and only offloading tasks to the cloud model. In [8], Xiao and Krunz proposed a workload scheduling method that ensures the user's response time has maximised under a power constraint. In their study, the energy spent while processing tasks have been ignored and only considering the energy consumed for offloading each unit of received workload. The decision to begin cooperating between fog nodes and offloading the workload taken by an agreement between the parties, the workload arrival rates, and the workload processing capabilities determines the amount of the offloaded workload. Their experimental results indicate that the average response time decreased due to allowing cooperation between fog nodes. Additionally, a crucial trade-off between the fog node's power efficiency and the average response time was observed. The authors proposed that the response time of end-user tasks should be set to its highest tolerable point to optimise the energy consumption at the fog computing systems, by doing that, most of the tasks will be processed at end-user devices and thus not offloading tasks and conserving energy.

In regard to online deployment of computational offloading, few studies have addressed that such as [15-17, 24-26]. In [16], Yousefpour et al. suggest a delay-minimisation approach to reduce the overall service delay. In their approach, the estimated queueing delay, which is utilised as the offloading threshold, determines whether a fog node processes its upcoming task(s), or offload it to one of its neighbours or the cloud server. If the offloading threshold has been reached, then the best neighbouring fog node in its domain is selected to offload its upcoming tasks. The best neighbouring fog node is chosen based on having the minimum total of propagation delay and queuing delay. Compared to other models, their results achieved the minimum average service delay. In [24], the problem of computational offloading has not been used explicitly, instead, the Yin et al, formulated the problem of deciding where to process end users’ tasks into task scheduling and resource allocation problem, where tasks are either processed locally at end-users devices or offloaded to fog nodes or cloud servers. In an intelligent manufacturing environment, the authors introduced fog computing and utilised the concept of the container within the fog system intending to reduce overall delay and optimise the number of concurrent tasks for the fog node. In their online model, generated tasks by end-users are transmitted to the Request Evaluator which is located at a fog node that decides whether to accept or reject the task based on its deadline requirement. If the task is accepted, then the task is transmitted to Task Scheduler which determines whether the task is processed at fog nodes or cloud servers based on the available resources and the execution time of this task which involves computation and transmission time. Finally, the Resource Manager responsible for reallocating the required resources to process the task at fog nodes. Experimental results show the effectiveness of their approach compared to other benchmarks.

Al-Khafajiy et al, [15] have proposed an offloading mechanism that allows fog-to-fog collaboration in heterogeneous fog systems intending to minimise overall service latency. Their mechanism utilises a FRAMES load balancing scheme that aims to detect congested fog devices, determine the amount of workload located at fog devices' queues that require offloading, based on their deadline requirement, and finally select the best fog node that provides the minimal service latency for the selected workload. They evaluated their proposed mechanism using a simulation. Their numerical results indicated the effectiveness of their proposed model in terms of minimising overall latency in comparison with different algorithms. In Fog-Cloud computing system, Gao et al. [17] investigate the issue of dynamic computational offloading and resource allocation. In order to reduce energy consumption and delay while having a stable queueing status, the authors formulate the problem as a stochastic network optimisation problem. They provide a predictive approach to computational offloading and resource allocation that depends on the trade-off between delay and energy use. Their approach implies that a delay reduction can be induced by increasing the allocation of computational resources at fog nodes, however, because of the processing of more tasks, energy consumption increases, and vice versa. Compared to other systems, the authors show the importance of their method. Mukherjee et al. [25] developed a scheduling strategy that manages to fulfil the deadline constraint of end-user tasks, taking into account computational resources. The deadline constraint of a given task and the availability of a neighbour, in their scheduling policy, help to decide on whether to place a given task in the fog node queue, e.g., in its high-priority queue or low-priority queue, or offload it to one of its neighbouring fog nodes. Their findings illustrate the efficacy of their suggested strategy as opposed to the no offloading and random schemes. Table 1 presents a summary of relevant articles concerning computational offloading at fog computing systems and the forms in which these systems execute.

2.2 Dynamic Server Energy Management

Dynamic Server Energy Management has been used in the area of the wireless local area network and the cloud, and it has proven to be efficient in terms of improving power quality. Although; up to the time of our study, this has not yet been implemented in the fog area. In WLANs, the energy efficiency has been enhanced by placing access points (APs) in sleep mode or turning them off. In [27], Marsan and Meo observed that in a community of APs, getting one AP in each community to control the system and service the incoming clients when all others are turned off will minimise energy consumption by up to 40 percent. Furthermore, an additional 60% of consumed energy can be saved if all APs are turned off especially during idle periods, e.g., at night. Li et al. [28] suggested an energy-saving method for state transformations in which APs are not only turned on and off based on consumer requirements, but there is also an intermediary stage that aims to reduce the frequency of switching. The authors stated that increasing the switching frequency will shorten AP's service life. In addition to that, the intermediary stage will help to avoid latency and energy overhead caused by switching on APs.

It has been suggested that servers could be periodically switched off [29, 30] or placed into sleep mode [31-33] in cloud computing systems to conserve energy resources. In [29-33], the authors examined the issue of the placement of virtual machines (VMs) to save resources concerning energy and yet retain QoS. When underutilised data centres are detected, all its VMs will be migrated to other active data centres, and these underutilised data centres are placed in sleep mode according to [31-33] or shutdown as per [29, 30]. This is intended to reduce the consumption of energy at cloud computing systems and is called 'VM consolidation'. Numerous VM migration approaches have been suggested to assess which virtual machines can be migrated from overloaded data centres. Moreover, in order to satisfy the QoS specifications of the system, a switched-off data centre may also be activated to handle the migrated VMs. According to Mahadevamangalam in [31], the energy demand for an idle-mode data centre is as almost as 70 percent of the energy generated by a fully-utilized data centre. Thus, by switching off idle-mode data centres, up to 70% of the energy consumed will be saved in the cloud system.

Table 1: Compuational Offloading State-Of-Art Comparison

Ref	Offloading Deployment	Architecture Model				Fog Cooperation	Offloading Threshold	Communication Type		Objectives						Evaluation Tool
		IoT-Fog	IoT-Fog-cloud	Fog-Cloud	Fog			Vertical	Horizontal	Delay	Energy				throughput	
											Yes	Relation with delay	Which Energy	Which device		
Tang el al [18]		✓	-	-	-	X	static	✓	X	✓	✓	Energy constraint	Energy for data processing and data transmission	IoT	X	MATLAB
Wang and Chen [19]		✓	-	-	-	X	static	✓	X	✓	✓	Energy constraint	Energy for local processing, processing at fog nodes, transmitting tasks	IoT devices and Fog nodes	X	Simulation
Liu et al [11]		-	✓	-	-	X	static	✓	X	✓	✓	Trade-off	Energy spent by local processing and transmitting tasks	mobile devices	X	Simulation
Mukherjee et al [20]		-	✓	-	-	✓	static	✓	✓	✓	X	X	X	X	X	Monte Carlo simulations
Zhu et al [13]		-	✓	-	-	X	static	✓	X	✓	✓	Offloading policy is designed to minimise task execution delay and to save energy	The energy spent by uploading and receiving tasks	Mobile devices	X	Simulation (MATLAB)
Mukherjee et al [21]	Offline	-	✓	-	-	✓	static	✓	✓	✓	✓	The goal of the study is to minimise total systems cost which includes total energy consumption plus total processing delay	Energy consumed by local computing and uploading tasks to fog nodes	End users' devices	X	MATLAB
Chen and Hao [14]		✓	-	-	-	X	static	✓	X	✓	✓	Battery capacity	Energy for transmitting tasks to edge devices and for local processing	End users' devices	X	Simulation
Sun et al [22]		-	✓	-	-	X	static	✓	X	✓	✓	Selects the least overhead cost which involves total computational time plus total energy spent by either processing task at IoT devices or transmission tasks to fog or cloud	Processing and transmission power	IoT, fog nodes, cloud servers	X	iFogSim
Zhao et al [12]		-	✓	-	-	X	static	✓	X	✓	✓	Minimises the system cost which is the total offloading latency and the total energy consumption	transmission and processing energy	Whole System: end users' devices, fogs, and cloud	X	Simulation

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 12 February 2021 doi:10.20944/preprints202102.0307.v1

Ref	Offloading Deployment	Architecture Model				Fog Cooperation	Offloading Threshold	Communication Type		Objectives						Evaluation Tool
		IoT-Fog	IoT-Fog-cloud	Fog-Cloud	Fog			Vertical	Horizontal	Delay	Energy				throughput	
											Yes	Relation with delay	Which Energy	Which device		
Meng et al [23]		-	✓	-	-	X	static	✓	X	✓	✓	Minimising energy given delay constraint	transmission + computational energy	Mobile Terminal, Fog servers and cloud servers	X	simulation
Xiao and Krunz [8]		-	-	✓	-	✓	static	✓	✓	✓	✓	Trade-off	Transmission energy	fog nodes	X	Simulation
Yousefpour et al [16]		-	✓	-	-	✓	static	✓	✓	✓	X	X	X	X	X	Simulation
Yin et al [24]		-	-	✓	-	X	static	✓	X	✓	X	X	X	X	✓	Simulation
Al-Khafajiy et al [15]		-	✓	-	-	✓	static	✓	✓	✓	X	X	X	X	X	MATLAB-based simulation
Gao et al [17]	Online	-	-	✓	-	✓	static	✓	X	✓	✓	Trade-off	Processing and transmitting tasks between fog nodes	Fog nodes	X	Simulation
Mukherjee et al [25]		-	-	-	✓	✓	static	X	✓	✓	X	X	X	X	X	simulation
Alenizi and Rana [1]		-	-	✓	-	✓	static	✓	✓	✓	✓	X	POWERING ON and processing tasks	Fog nodes	✓	iFogSim
Our Work		-	-	✓	-	✓	Dynamic	✓	✓	✓	✓	X	POWERING ON and processing tasks	Fog nodes	✓	iFogSim

2.3 Comparison of the State-Of-The-Art

Table 1 provides a summary of related papers that have discussed computational offloading in fog computing systems, highlighting the architecture model, e.g., IoT-Fog means that end-users' tasks are processed locally at IoT devices or offloaded and processed at fog nodes, fog cooperation, communication type, the stated objectives and the evaluation tools arranged by offloading deployment, both offline and online.

We find that based on previous research, computational offloading has been investigated at both online and offline stages. Both stages are important to ensure optimum results: offline deployment helps to predict the best output for the system at its design stage; and online deployment mimics various scenarios in real-world environments, involving uncertainty and unpredicted events, and helps the system to ensure a better outcome. However, most of the literature is focused on the offline deployment, while online deployment has received less attention. Additionally, the problem of computational offloading is usually investigated with the aim of minimising an overall delay in the system; managing the system resources is sometimes included, especially minimising energy consumption of IoT/end users' devices. This sheds light on the need for further investigation into the problem of computational offloading along with resource management in regard to fog resources.

Managing resources in the system is much easier within offline deployment than online deployment, especially when all the system data is known in advance compared to dealing with uncertainty. In offline deployment few works have addressed the topic of reducing energy consumption in the system. Within these works, most attention has been given to addressing the energy consumed at IoT devices, and the energy consumed at fog devices has received the least attention. Additionally, when considering energy spent at fog nodes, the research either addresses the energy spent under delay constraint or the trade-off between delay and energy, based on system requirements. Regarding the problem of resource management within online deployment, researchers address the trade-off between delay and energy at fog nodes.

In this work, we consider the online deployment of computational offloading and the potential for minimising energy consumption at fog nodes by analysing the powering-on energy of fog nodes. At present, there have been many researchers addressing the energy spent while powering on servers in cloud and WLANs environments, whereas the energy spent for powering on fog nodes has not yet been widely investigated. Computational offloading and resource management at fog environments are still at the initial phase, and thus, more research should be done in this area. In addition, when tackling the problem of computational offloading, most previous works utilised a fixed threshold that determines when to start offloading; in the current work, a dynamic threshold is investigated to address its impact on the system.

3. System Modelling and Constraints

We implement the same system model and constraints as in [1], and the new proposed Fog Node Architecture is explained in section 3.1.3 .

3.1. System Model

In this section, network diagram is presented in section 3.1.1, and in section 3.1.2, the application module is described.

3.1.1 Network Diagram

Figure.1 shows an illustration of the fog computing architecture and comprises of three layers:

- **The IoT devices layer**: This layer is composed of many mobile vehicles. An actuator and a collection of sensors is placed on the vehicle's node. Each one of these sensors emits a different type of task. There are two forms of tasks emitted by the mobile vehicle, called non-urgent and urgent tasks. The non-urgent task provides data including current position, speed, and path. The urgent task requires a quick response and is quite critical. This task may contain a video stream of the surrounding of a moving vehicle, the processing of this type in a short latency will prevent possible collisions. This is necessary, especially for self-driving vehicles.
- **Fog computing layer:** This layer is comprised of a series of fog nodes and a fog controller. Fog nodes are located in roadside units (RSU) that are installed on numerous streets. If fog nodes are situated in each other's proximity, they can interact with each other [34]. Fog nodes forms an ad hoc network to exchange and share numerous data. All fog nodes are linked to the fog controller, which is responsible for managing fog resources and controlling fog nodes. Fog nodes process two different type of tasks, for urgent tasks, fog nodes process these tasks and send the result back to the vehicle. For non-urgent tasks, fog nodes process these tasks and transfer the findings to the cloud for further analysis and storage for retrieval by traffic management organisations.
- **Cloud computing layer:** this layer cis composed of a set cloud server. It controls the traffic at the city level based on historical evidence.

3.1.2 Application module description

Three modules called Road Control, Global Road Monitor and Process Priority Tasks are part of the application model of this report. The first two modules specialise on traffic light management, while the last module is responsible for processing urgent tasks. The following is the responsibilities of all the three modules

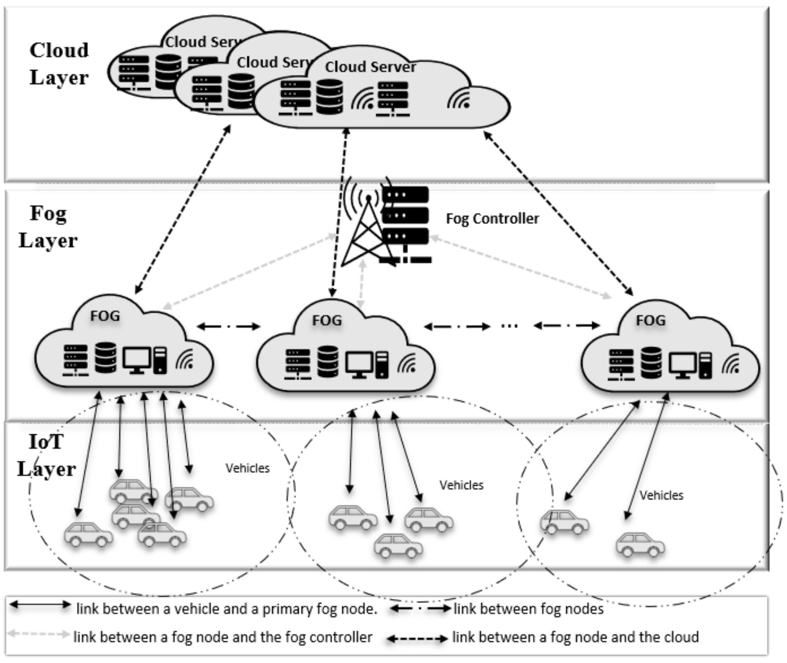


Figure 1: Fog Computing Model

• **Road Monitor**: the placement of this module is in fog nodes. When a vehicle approaches a region covered by a fog node, the sensor immediately sends data to the connected fog node for analysis. This data contains the current position, the speed of the vehicle, weather, and road conditions. After processing these data by the specified module, the results transmitted to the cloud for further processing.

• **Global Monitor**: The placement of this module is in the cloud. This module receives the data that already processed by the Road Monitor module, after receiving, these data are processed and stored for further analysis.

• **Process Priority task:** this module is placed in fog nodes and is responsible for processing the priority requests from the user. The results are then sent back to the user. The application in iFogSim is defined as a directed acyclic graph (DAG) = (M, E) where M representing the deployed application modules M = {m_1, m_2, m_3, ..., m_n}, e.g., Process Priority Task, Road Monitor and Global Road Monitor modules. E represent a set of edges, which represents the data dependencies between application modules. Figure.2 demonstrates this.

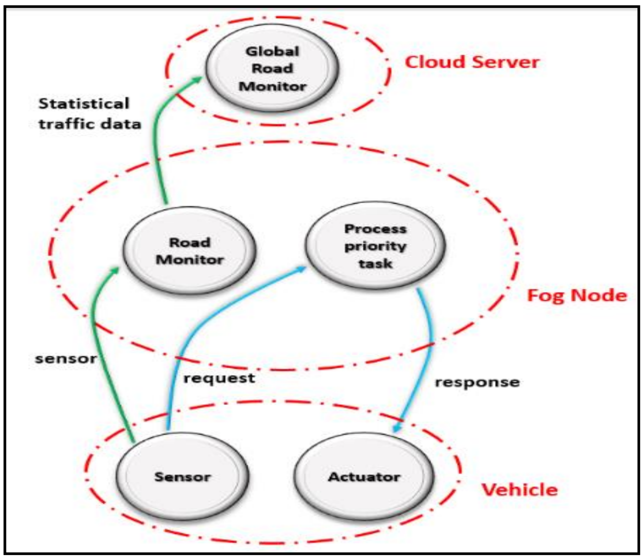


Figure 2: Directed Acyclic Graph (DAG) of the application model.

3.1.3 Fog Node Architecture

The proposed fog node architecture consists of three parts; namely, Task Scheduler, Best Neighbour Selection and Threshold Monitor (see Figure 3). Task Scheduler receives tasks generated from IoT devices within the proximity of the primary fog node and from other neighbouring fog nodes. If a fog node receives a task that is already offloaded from another neighbour, Task Scheduler immediately inserts this task in the queue to wait for processing. If the task is generated from any IoT devices, then Task Scheduler will check the offloading threshold and compare that to the fog node queuing delay. If the queueing delay reaches the offloading threshold, then Task Scheduler sends this task to the Best Neighbour Selection, which in turn decides the best neighbour node to offload this task to. The selection of the best neighbour is described in more detail in section 3.2.2. Threshold Monitor is responsible for dynamically increasing and decreasing the offloading threshold for both the primary fog node and all of its neighbours, based on the received workload and the availability of other neighbours: this is done from the perspective of the primary fog node. On the one hand, it is assumed that fog nodes are cooperative and accept tasks coming from their neighbour nodes, even if this exceeds their threshold. On the other hand, each neighbour has its own Threshold Monitor, and the primary fog node and all its neighbours do not necessarily have the same threshold value. In Figure 4, we can see that the primary fog node A in table (a) set its threshold to 9 ms for itself and all its neighbours. At the same time, primary fog node B in table (b) set its threshold to 6 ms, even for its neighbour fog node A; therefore, it can be seen that fog node A is congested and will not be selected as the best neighbour for fog node B. Determining when to increase and decrease the offloading threshold is described more in section 5.

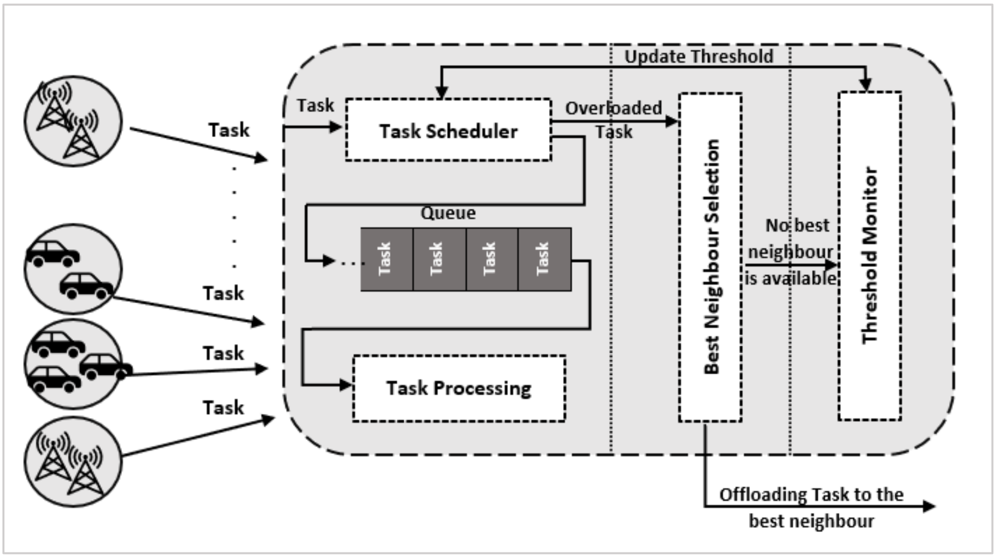
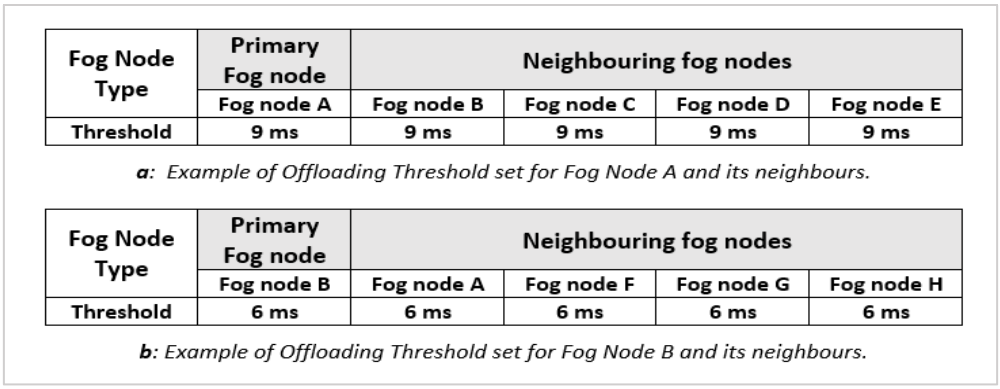


Figure 3: Fog Node Architecture Model.



Fog Node Type	Primary Fog node	Neighbouring fog nodes			
	Fog node A	Fog node B	Fog node C	Fog node D	Fog node E
Threshold	9 ms	9 ms	9 ms	9 ms	9 ms

***a**: Example of Offloading Threshold set for Fog Node A and its neighbours.*

Fog Node Type	Primary Fog node	Neighbouring fog nodes			
	Fog node B	Fog node A	Fog node F	Fog node G	Fog node H
Threshold	6 ms	6 ms	6 ms	6 ms	6 ms

***b**: Example of Offloading Threshold set for Fog Node B and its neighbours.*

Figure 4: Example of Offloading Threshold values for the Primary Fog Node and all its Neighbours at the same Time.

3.2 Types of Connections and Constraints

This section explains the relations between a vehicle and a fog node, between fog nodes, and between fog nodes and cloud. Additionally, the set of restrictions that render these relations appropriate.

3.2.1 Connection between Vehicles and Fog nodes

The interaction between a vehicle and a fog node is controlled by communication and processing restrictions.

- Communication Constraints

- Each vehicle connects to a fog node if it is located within the coverage radius of that fog node, as constraint (1).

$$D_{v,f} \leq max\ Coverage_f;\ \forall\ v \in V, \forall\ f \in\ FN \tag{1}$$

Where V represents all vehicles, v is a single vehicle, FN represents all fog nodes and f is a single fog node. $D_{v,f}$ is the distance between a vehicle v and a fog node f, is calculated as

$$D_{v,f} = \sqrt{\left(X_v - X_f\right) + (Y_v - Y_f\ )};\quad \forall\ v \in V, \forall\ f \in\ FN \tag{2}$$

Where (X_V, Y_V) and (X_f, Y_f) are the location of the coordinates of a vehicle v and a fog node f, respectively. When a vehicle is within a range of several fog nodes, it will connect to the nearest fog node. This is to decrease delay, as the propagation delay relies on the distance between the two connected items. Propagation delay (PD) is calculated as

$$PD = \frac{D_{v,f}}{PS} \tag{3}$$

Following [35], we believe that the speed of signal propagation (PS) is equivalent to the speed of light, c = 3 × 108.

- Processing Constraints

The placement of the required application modules is essential at fog nodes to process the upcoming tasks at the fog paradigm. To guarantee the placement of these application modules, fog nodes should have enough resources that meet the demand of these application modules. These resources are; CPU, Ram, and Bandwidth.

$$\sum_{i=0}^{M} Required_{Capacity}\ m_i \leq \sum Available_{Capacity}\ f;\ \forall m_i \in \mathrm{M}, \forall f \in \mathrm{FN} \tag{4}$$

The required capacity of an application module is CPU, Ram, and Bandwidth, and the available capacity in a fog node is CPU, Ram, and Bandwidth. Constraint (4) indicates that the overall needed capability of all application modules should not surpass the available capacity of the fog node in which they are installed. In iFogSim simulator,if no available capacity at the fog paradigm, the application will be placed at the cloud servers. The required CPU to place an application module is calculated as following:

$$CPU = NV * (Rate * TaskCPU) \tag{5}$$

Where NV is the number of vehicles attached to the fog node, and TaskCPU is the task CPU length which is the number of instructions contained in each task in Million Instructions Per Second (MIPS). Rate is calculated as:

$$Rate = \frac{1}{Transmission\ Time\ in\ ms} \tag{6}$$

In iFogSim, the placement of application modules happens at the design stage before running the system. Increasing the number of connected vehicles at a fog node will increase the required CPU to place the required application modules, and if the fog node has no enough CPU, these applications will be placed in the cloud. In this case the number of connected vehicles for each fog node is limited as constraint (7).

$$\sum_{i=0}^{V} v_i\, f_j \leq MAX_{vehicle\ number};\ \forall\, v_i \in V, \forall\, f_j \in FN \tag{7}$$

3.2.2 Connection between Fog nodes

In this section, we explain the waiting queue for fog nodes in which the offloading decision is determined, the criteria of the communication between fog nodes and the selection of the best neighbouring fog node.

- Fog nodes' waiting queue.

All fog nodes contain a queue where tasks are put when they arrive at the fog nodes, these tasks are served in the basis of first in first out (FIFO). Fog nodes execute one task at a time. After the task is done the fog node can process the next task according to its scheduling strategy. Queueing delay triggered the decision to begin offloading tasks to neighbouring fog nodes [16]. To begin offloading tasks, the queue waiting time should be beyond the offloading threshold, e.g., 50ms, 100ms, 150ms or 200ms.

$$T^{Queue} > Max_{threshold} \tag{8}$$

T^{Queue} is calculated as

$$T^{Queue} = \sum T_i * T_i^{process} + \sum T_z * T_z^{process};\ \forall\, i, z \in T \tag{9}$$

Where T_i and T_z are the total number of tasks of the type *i* and *z*, e.g., urgent or non-urgent. T is all tasks and is the expected execution time of a specific task and calculated as

$$T^{process} = \frac{TaskCPU}{F\_MIPS * N\ of\ PS} \tag{10}$$

Where F_MIPS is the total mips available in a fog node and N of PS is the total number of processing units allocated in that fog node.

- Coverage Method

To ensure the coverage of an area, a number of fog nodes are required. Fog nodes can overlap to achieve maximum coverage as in [36] see Figure. 5.

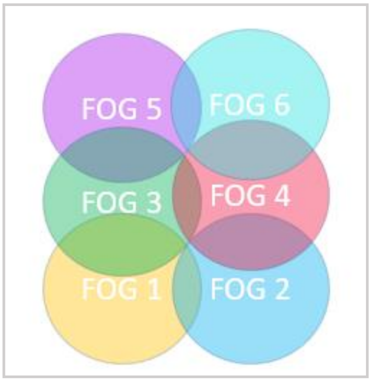


Figure 5: Overlapping Fog Nodes.

- Selecting the Best Neighbouring Fog Node

Following [16], the best neighbouring fog node is selected based on propagation delay plus transmission delay. The selection of the best neighbouring fog node begins if the offloading threshold of a fog node is reached, e.g., 50ms, the offloading threshold is determined by the waiting queue time. A Fog node can communicate with other fog nodes if they are located within the coverage radius of the fog node itself, in this case, they are called neighbours. This is shown in constraint (11)

$$d_{ij} \leq Coverage_{radius};\ \forall\, i,j\ \in\ FN \tag{11}$$

dij represents the distance between a fog node i and j. In Figure. 5, the neighbouring fog nodes for FOG 1 are FOG 2 and FOG 3. Additionally, the neighbouring fog nodes for FOG 3 are FOG 1, FOG 4, and FOG 5. The criterion for choosing the best neighbouring fog node is based upon the following considerations. The neighbouring fog node should be located within the coverage radius of the primary fog node. Besides that, the minimum sum of queueing delay plus propagation delay is required to assess the best neighbour.

$$Min \sum T^{Queue} + PD \tag{12}$$

PD is calculated as

$$PD\ =\ \frac{D_{f,f'}}{PS} \tag{13}$$

($D_{f,f'}$) is the distance between fog nodes f and f', it is calculated similar the distance between a vehicle and a fog node and propagation speed PS is equivalent to the speed of light, following [35] PS value is 3×108.

3.2.3 Between Fog Nodes and the Cloud

In the present work, we focused on sharing the workload with other neighbouring fog nodes, the cloud is the least to be considered. This is attributed to the availability of other neighbouring fog nodes as they overlap and to get maximum utilisation of the available resources in the fog system. Owing to the cloud server's efficient computing capability relative to fog nodes, queueing latency is ignored such that tasks are processed immediately upon arrival [37-39].

4. Problem Formulation

Minimising delay and energy consumption is considered an optimisation problem and decomposed into two sub-problems [1]: the delay minimisation problem and the energy-saving problem.

4.1 Delay Minimization Problem

The response time is the time required for sending the workload from a vehicle to the connected fog node and getting the results back. It consists of the transmission delay, propagation delay, queuing delay and processing delay. If the processing of the task is done at the primary fog node, then the service latency is calculated as

$$T\ =\ T^{sTv}\ + 2\ X\ (T_{vTf}^{Transmision}\ + PD_{vTf}\ ) + T^{Queue}\ +\ T^{procss}\ +\ T^{vTa} \tag{14}$$

Where T^{sTv} and T^{vTa} are the latency time between a vehicle and its sensor, and between the vehicle and its actuator, respectively. $T_{vTf}^{Transmision}$ is transmission delay between the vehicle and its primary fog node. It is based on the network length of the task and the bandwidth, and it is calculated as

$$T^{Transmision}\ =\ \frac{Network\ Length\ of\ Task}{Bandwidth} \tag{15}$$

If a neighbouring fog node is incorporated in the processing of the received task, then the latency is calculated as

$$T = T^{sTv} + 2\,x\,(T_{vTf}^{Transmsiion} + PD_{vTf}) + 2\,x\,(T_{fTf}^{Transmission} + PD_{fTf}) + T^{Queue} + T^{Process} + T^{vTa} \quad (16)$$

If the cloud is incorporated in the processing of the task, then the latency is calculated as

$$T = T^{sTv} + 2\,x\,(T_{vTf}^{Transmsiion} + PD_{vTf}) + 2\,x\,(T_{fTc}^{Transmission}) + T^{Process} + T^{vTa} \quad (17)$$

4.2 Energy Saving Problem

Minimising the power consumption of fog nodes brings many advantages, including but not limited to decreasing the overall cost of electricity and reducing the environmental impact. Two power modes are presented for each fog node: idle and busy. In the idle mode, the fog node is not performing any processing and the power is ON, and in the busy mode if the fog node is busy processing tasks and the power is ON. The energy consumed is determined by how much power the fog node consumes when processing workload and when the fog node is not processing tasks. The total energy consumption in iFogSim is calculated as in [40] as

$$E = PR + (TN - LUT) * LUP \quad (18)$$

Where PR is previous total energy consumed in this fog node, TN is the time now which is the time that the updateEnergyConsumption () is called when utilising this fog node, LUT is the last time this fog node has been utilised and finally LUP which is the fog node last utilization power status, which is idle power or busy power, the value of this is based on the predefined parameters when creating a fog node. The problem of minimizing delay and energy is formulated as follows:

Min $\sum$T & $\sum$E

s.t. (1), (7), (4)

$$T^{Queue} \leq Max_{threshold} \quad (19)$$

$$P_F + P_N = 1, P_F \,\&\, P_N = \{0, 1\} \quad (20)$$

Equation (1) ensures the connection between a fog node and a vehicle that is located within its coverage range. Equation (7) guarantees the number of vehicles connected to one fog node does not exceed the threshold number. Constraint (4) ensures the placement of the required application modules at fog nodes. Equation (19) ensures the stability of fog nodes' queues so that, to process its upcoming tasks, the waiting queue time should not exceed its threshold. In constraint (20), PF and PN mean that if the task is processed in its primary fog node, then PF = 1 and PN = 0 and vice versa. Therefore, the task is either processed in the primary fog node or one of its neighbours.

5. Proposed Algorithms

In this work, we applied the two proposed algorithms that utilised in our previous work [1], called dynamic task allocation and the second is called dynamic resource saving, both stated algorithms need to work together to achieve the intended outcome. These two algorithms were applied with a static offloading threshold. In the current work we proposed a dynamic offloading threshold, in which the offloading threshold is increased or decreased based on the workload and the availability of other neighbouring fog nodes. This is described in Section 5.1.

5.1. Dynamic Offloading Threshold

Instead of using a fixed offloading threshold, upon which the fog node determines that it is now overloaded and starts sharing its workload by offloading its upcoming tasks to the best neighbour, a dynamic threshold is proposed. This technique is performed by the Threshold Monitor, which adjust its value periodically according to the received workload and the availability of other neighbours. A dynamic offloading threshold has been developed to help the fog node decide if the received task should be processed locally or offloaded to the best neighbour, based on thresholds that are adjusted dynamically, see Algorithm 1.

The first part of the algorithm (Procedure 1) determines whether to increase the primary node's threshold or not. This starts during the process of selecting the best neighbour, and when the primary fog node reaches its offloading threshold, as per lines 1-11. If the best neighbour is available, then the primary fog node will offload its task to the best neighbour, and thus no change will be applied to the threshold. Otherwise, the threshold will increase, based on equation (22). The second part of the algorithm (Procedure 2) determines whether to decrease the threshold or not. This runs each time a new task is received, and when the fog node finishes the execution of a task. It starts by selecting the best neighbouring fog node with the lowest queueing delay, as per lines 19-23. It then checks if the queueing delay of the primary fog node is larger than a predetermined value, as per line 24. If the queueing delay of the best neighbour is lower than the current threshold, then the average queueing delay for all the neighbours is calculated and a new threshold is determined, as per lines 25-30. Parameters used in this algorithm are in Table2.

Table 2: Description of Parameters used for Dynamic Threshold Algorithm**.**

Symbol	Description	Symbol	Description
δ^n	Refers to the initial offloading threshold and the current threshold.	α	When the queuing delay reaches this threshold, the fog node might consider decreasing its offloading threshold.
VQ	Average queueing delay of all the neighbours	N	The Best neighbour fog node
δ^{n+1}	New offloading threshold.	T	The arrival task
x	x = δ^n /2.	Q	Queuing delay in the primary fog node
Ns	All neighbouring fog nodes	p	a number bigger than zero that determine how much to increase and decrease the new offloading threshold based on the current offloading threshold
Qs	Set of all queueing delay of all its neighbours	Q_N	Queueing delay of one neighbour

$$VQ = \frac{\sum_{s=0}^{Ns} Qs}{Ns} \tag{21}$$

$$\delta^{n+1} = \begin{cases} \delta^n - p, & Q \geq \alpha, VQ < x \\ \delta^n, & Q \geq \alpha, VQ \geq x \\ \delta^n, & Q < \alpha \\ \delta^n + p, & Q \geq \delta^n, VQ \geq \delta^n \end{cases} \quad \forall\, x, \alpha, p > 0 \tag{22}$$

Algorithm 1 Dynamic Offloading Threshold

```
Input:  $\delta^n$, Qs, T, Q;
Initialisation  $\delta^{n+1}$ = $\varnothing$, Best = $\varnothing$, min = Max-value;
Result/s:  $\delta^{n+1}$;

Procedure 1.
  IF (Q>= $\delta^n$)
    For each N $\in$ FN do
      IF ($Q_N$ < min) then
        min = $Q_N$
        Best = N
      end if
    end for
    IF (min < Q) then
      Offload T to Best
      $\delta^{n+1}$ = $\delta^n$
    else
      Calculate $\delta^{n+1}$ using equation (22)
    end if
  end if
  return $\delta^{n+1}$
end Procedure 1
Procedure 2.
  For each N $\in$ FN do
    IF ($Q_N$ < min) then
      min = $Q_N$
    end if
  end for
  IF (Q >= α) then
    IF (min < $\delta^n$) then
      Calculate VQ using equation (21)
      Calculate $\delta^{n+1}$ using equation (22)
    end if
  end if
  return $\delta^{n+1}$
end Procedure 2
```

6. Experimental Results

In this work, iFogSim has been used to simulate the environment. It is a toolkit developed by Gupta et. al [41], which is an extension of the CloudSim simulator. It is a toolkit allowing the modelling and simulation of IoT and fog environments and is capable of monitoring various performance parameters, such as energy consumption, latency, response time, cost, etc. Simulation settings values were used as in [1]. The simulation was run with one cloud server, seven fog nodes, the fog controller, and a total of 50 vehicles. Each vehicle transmits two different tasks every 3 ms. The metrices used to measure the performance are:

• **Service latency** as the average round trip time for all tasks processed in the fog environment. two control loops in the simulation: **Control loop A:** Sensor -> Process Priority Tasks -> Actuator. This control loop represents the path of the priority requests. **Control loop B:** Sensor -> Road Monitor -> Global Road Monitor. This control loop represents the path of the non-priority requests.

• **Throughput**, which is measured as the percentage of the processed tasks within a time window. It is calculated as

• **Total Energy Consumption** in fog environment caused by powering on and processing tasks.

6.1 Performance Comparisons with Various Computation Offloading Schemes

To evaluate the effectiveness of our proposed algorithm, the comparisons with various computation offloading schemes are provided, where the number of vehicles is set to 50 and total number of fog nodes is set to 7. Although we implement uncertain system to mimic real world scenarios, we kept the same values for all parameters including the number of total generated tasks, the capacity of fog nodes, the size of the generated tasks identical when tasting all the below stated schemes for fair comparison. In particular, the following four schemes are selected as benchmarks:

Benchmark 1: *No Offloading Scheme (NO)*: in this scheme, each primary fog node processes all the tasks in its coverage alone without cooperation with other neighbouring fog nodes.

Benchmark 2*: Joint Task Offloading and Resource Allocation Scheme (JTORA) in [20]*: in this scheme, if the primary fog node doesn't have enough computational resources that meets the delay requirement of a task, then the task will be offloaded to a neighbouring fog node within the proximity of the primary fog node that has enough computational resources. Any underutised neighbour is a candidate of processing the overload, ignoring the selection of the least utilised fog node.

Benchmark 3: *Workload Offloading Scheme (WO) in [26]*. In their work, end users offload their computational tasks to a broker node that manages the system, the broker node will send tasks to the closet fog node near to end users (primary fog node), if the primary fog node is congested (e.g. its queueing delay reaches 50 ms), then the broker node will offload the task to any underutilised neighbouring fog node.

Benchmark 4*: Static Threshold 50-ms Scheme (ST50) in our previous work [1]*: in which, when the offloading threshold is set to 50 ms, upon which the primary fog node makes the decision on whether to process the task locally or offload it to the best neighbouring fog node.

The four benchmarks are compared to the proposed offloading policy called Dynamic Threshold (DT).

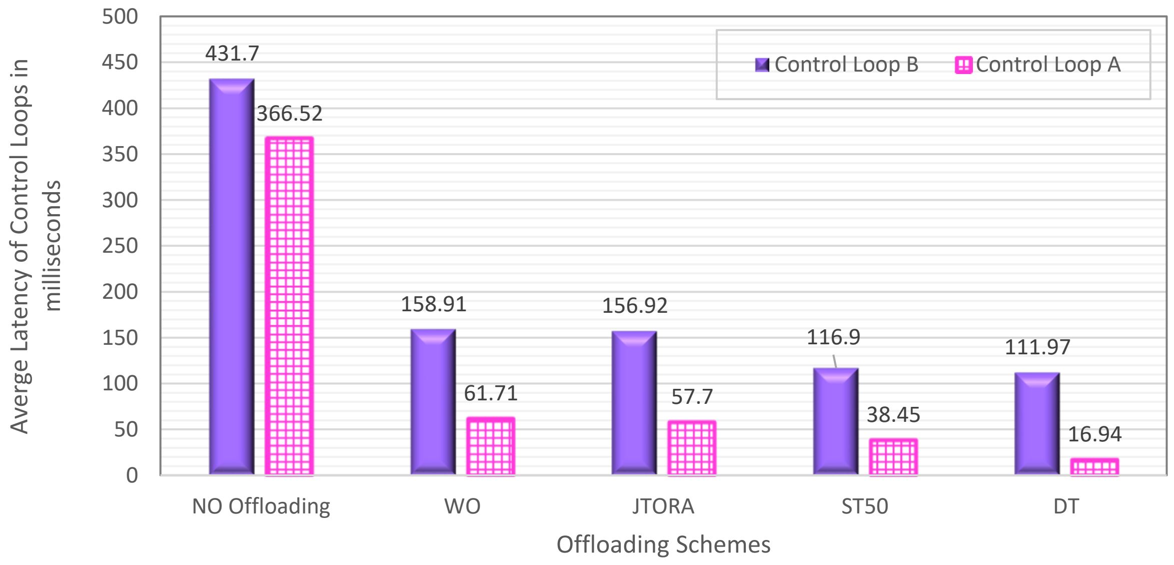


Figure 6: The comparison of the average latency with various offloading schemes.

In Figure 7, the impact of various offloading schemes on average latency is addressed. It can be seen in Figure 6 that delay is very high in the no offloading scheme; this is due to a long queueing delay as tasks are not shared by the primary node with other neighbouring fog nodes, so they are waiting to be executed by the primary fog node. The impact of allowing cooperation between fog nodes in terms of sharing workloads is shown comparing other schemes to the no offloading scheme. Additionally, the impact of selecting which neighbour to share the workload with is very clear when comparing the WO, JTORA, ST50 and DT schemes. In the WO and JTORA schemes, when the primary fog node is congested (e.g., reaching its offloading threshold), it selects any underutilised neighbour to share the workload with, rather than selecting the least utilised neighbour, as in ST50 and DT. In addition, the delay is higher in the WO scheme compared to JTORA; this is due to a communication overhead caused by sending tasks to a broker node first, which in turn decides whether to process these tasks at the primary fog node or any underutilised neighbour. The least mount of delay is achieved for both control loops when applying our proposed algorithm, DT, compared other benchmarks.

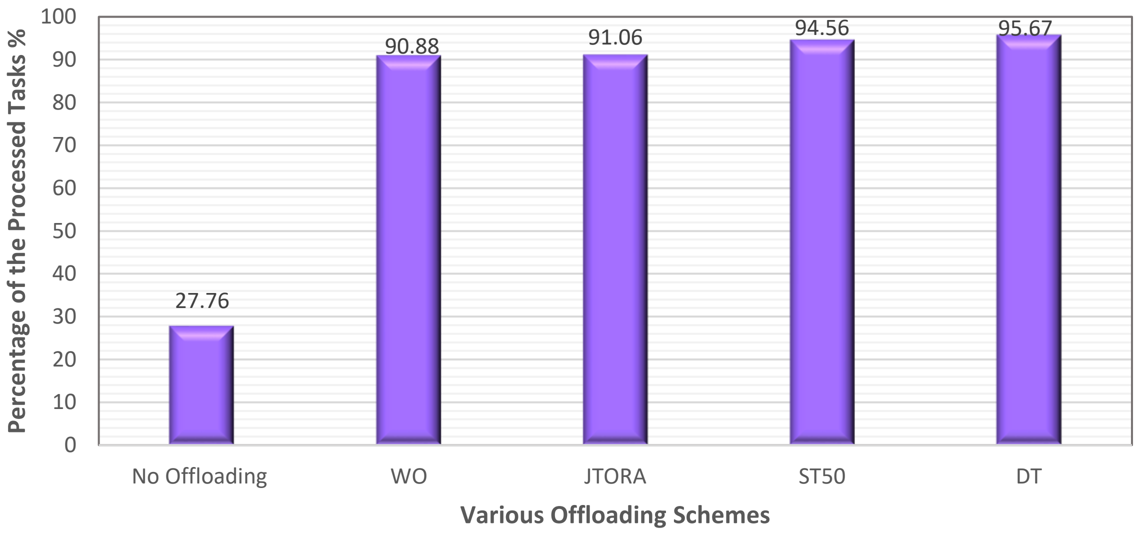


Figure 7: The comparison of the throughputs with various offloading schemes.

The impact of various offloading schemes on throughput is shown in Figure 8. It can be seen that the lowest percentage of processed task is when no offloading is applied; this is obvious as most of the tasks are waiting in the queue to be executed by the primary fog nodes. The impact of sharing workload with any underutilised neighbours is very clear in WO and JTORA schemes, resulting in processing almost 90.88% and 91.06% of tasks, respectively, compared to 94.56% and 95.67% in ST50 and DT schemes, respectively.

6.2 *Impact of increasing number of vehicles on delay and throughputs with different offloading schemes*

The impact of increasing the number of vehicles investigated to see how the delay is maintained as we increase the workload in the online system. In this experiment, the total number of fog nodes is set to seven and the number of vehicles ranges from 4 to 48. In Figure 8 we can observe that when the number of vehicles is small, ranging from 4 to 12 vehicles, the DT, ST50 and JTORA schemes exhibit almost the same pattern. This is because the generated workloads are small, resulting in the primary fog nodes processing most of these workloads themselves. When the number of vehicles increased, all three approaches ST50, JTORA and WO show a dramatic increase in delay compared to DT, which displays a stable pattern with a slight increase in delay that increases as the number of vehicles increased. The reason for the huge increase in delay for ST50, JTORA and WO is that increasing the workload makes the primary fog nodes almost reach their offloading threshold (e.g. 50 ms), but not always exceeding it, resulting in the primary nodes processing most of the workload with little help from other neighbours. The impact of selecting the best neighbour to share the workload with becomes clear when the number of vehicles is high (i.e. 28

vehicles) as the delay is low in ST50 and DT schemes compared to JTORA and WO approaches. The overall results show the effectiveness of the DT scheme even when increasing the number of vehicles.

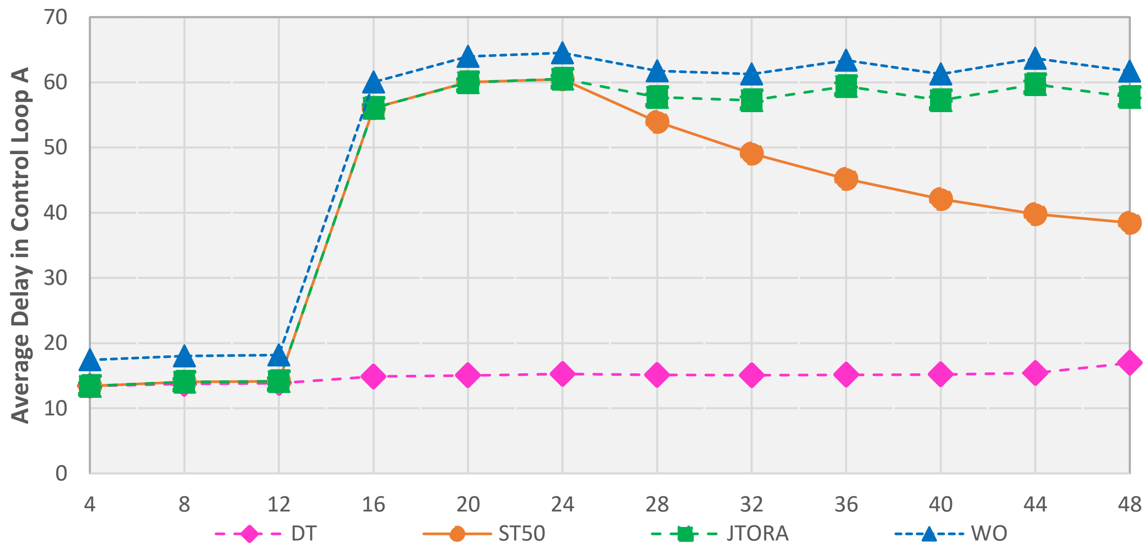


Figure 8: Impact of Increasing Number of Vehicles on avergae delay with Different Offloading Schemes

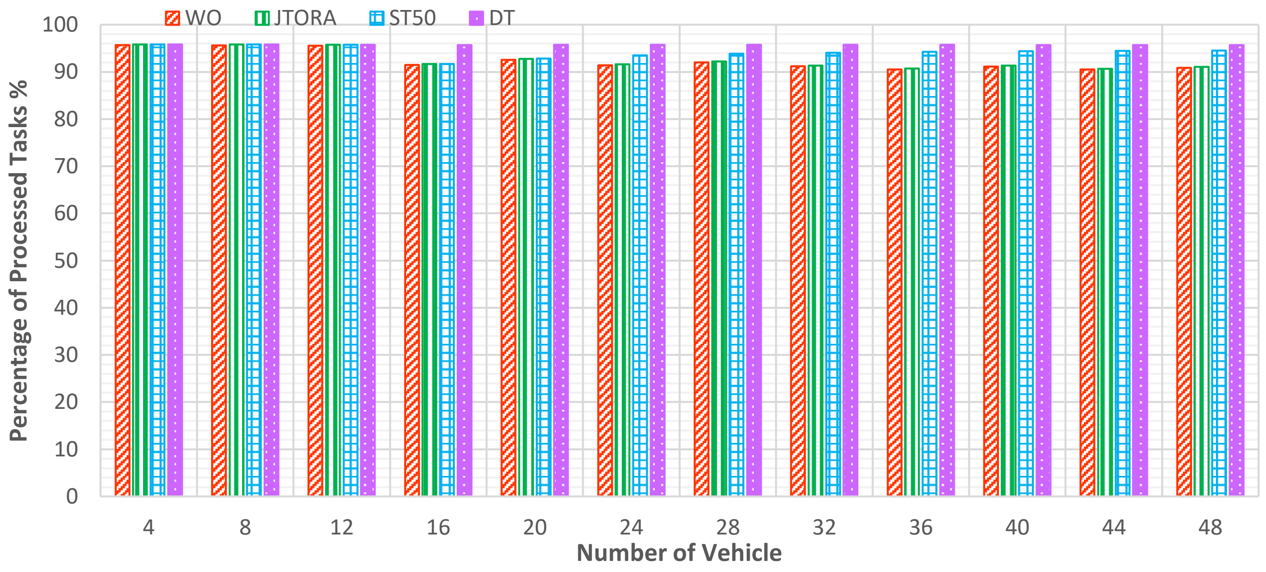


Figure 9: Impact of Increasing Number of Vehicles on throughputs with Different Offloading Schemes

The impact on throughputs has also been investigated while increasing the number of vehicles, see Figure 9. It can be seen that when there is a small number of vehicles, ranging from 4 to 12, all the offloading schemes operate in a similar way; this is because the workload is minimal and can be processed at the primary fog nodes wihout any help from other neighbours. When the number of vehicles is increased, DT acheived the highest throughput, with almost 96% compared to other schemes, which accomplished around 94.5%, 91% and 90% at ST50, JTORA and WO, respectively.

6.3 impact of incresaing number of neighbours on delay, throughputs and energy with various offloading schemes

The impact of increasing the number of neighbours has addressed to see its impact on overall system performance and to find the optimal number of neighbours that are required to obtain optimum results. From Figure 10, it can be seen that as the number of neighbours is increased, it can be seen that the delay gradually decreased in both control loops. However, when a certain number of neighbours is reached (e.g., five neighbours), the delay remains almost stable despite adding further neighbours. This means that the optimal number that is required to achieve minimum delay has been reached, and there is no futher need add more neighbours to save the energy consumption of the fog paradigm. The reason for the stable pattern is attributed to the workload, as most of the generated tasks have been processed.

Regarding the comparison of the four schemes, it can be noticed that DT accomplished the least delay for both control loops as the number of neighbours is increased, compared to the other schemes; ST50, JTORA and WO. With three neighbours, DT decreased delay by 10.80%, 13.38% and 15.29% compared to ST50, JTORA and WO respectively. When the number of neighbours is five, the DT scheme reduced delay by 55.94%, 70.64% and 72.55% in comparison to ST50, JTORA and WO, respectively.

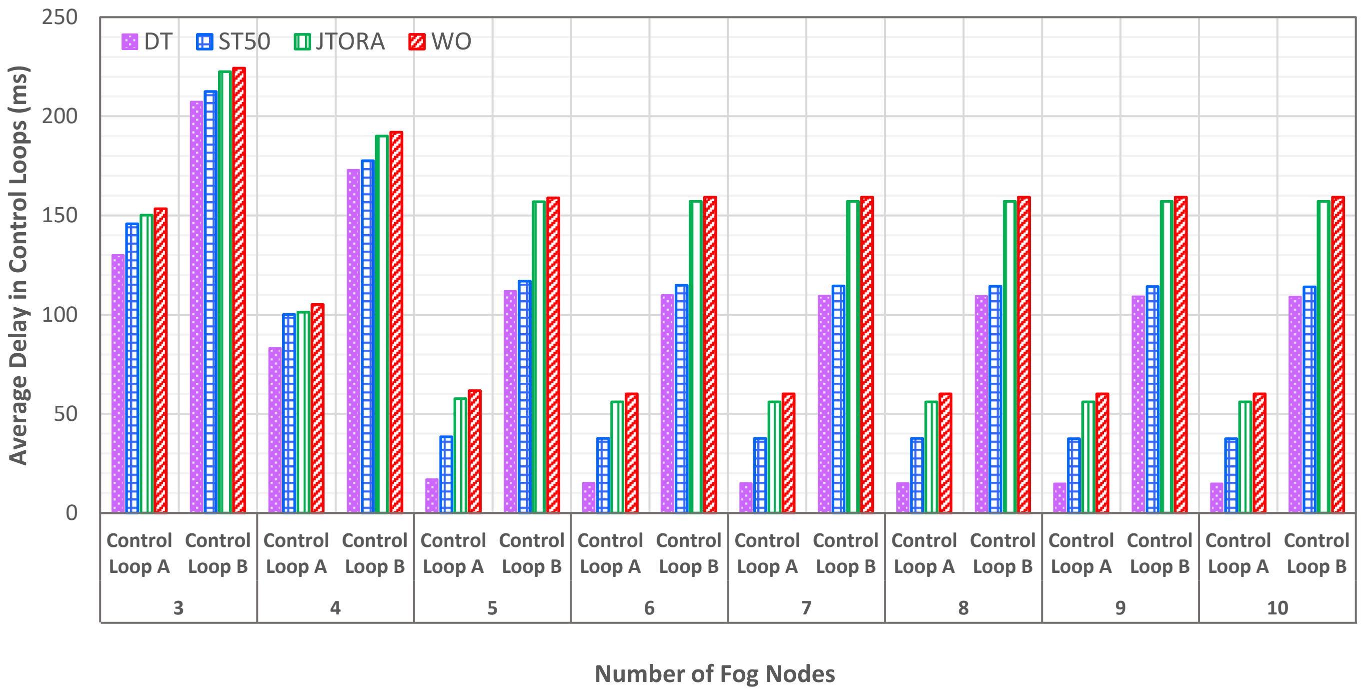


Figure 10: impact of increasing number of neighbours on average delay in control loops with various offloading schemes

Regarding the impact of increasing the number of neighbours on throughput, the pattern shows almost the same pattern as when increasing the number of neighbours to decrease delay, see Figure 11. As the number of neighbours increased, the percentage of the processed tasks increased, until a certain number of neighbours is achieved (e.g., five neighbours), after which the pattern remains almost stable. The reason behind the stable pattern is due to the workload as most of the generated tasks are processed. The reason why the percentage of processed tasks does not reach 100% is that this study implements an online dynamic system, therefore vehicles are still generating tasks until the end of the simulation; 5% of the total generated tasks are not processed because they are newly generated.

In terms of the comparison with other schemes, DT improved throughputs by 0.10% when the number of neighbours is three, 1.16% when the number of neighbours is four, and 1.11% when the number of neighbours are five, six, seven, eight, nine and ten, compared to ST50 scheme. When the optimal number of fives neighbouring fog nodes is reached, the DT processed 95.66% of the total generated tasks, while ST50, JTORA and WO processed 94.55%, 91.06% and 90.88%, respectively. The DT scheme helped to improve throughput compared to other stated schemes as the number of neighbouring fog nodes was increased.

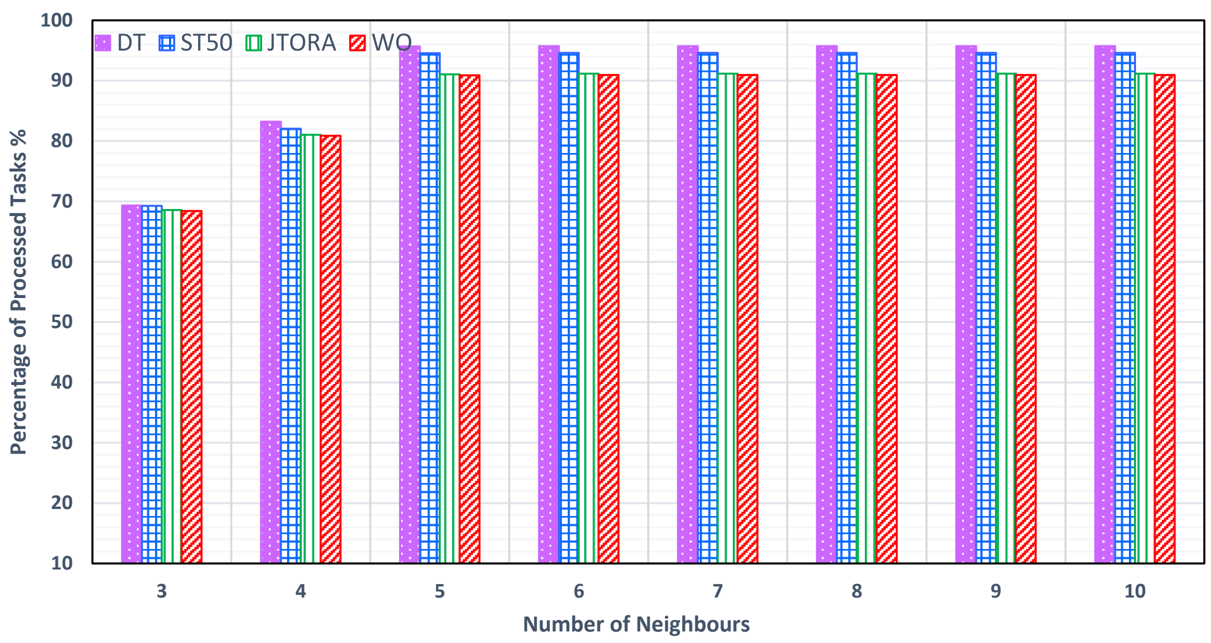


Figure 11: impact of increasing number of neighbours on throughputs with various offloading schemes

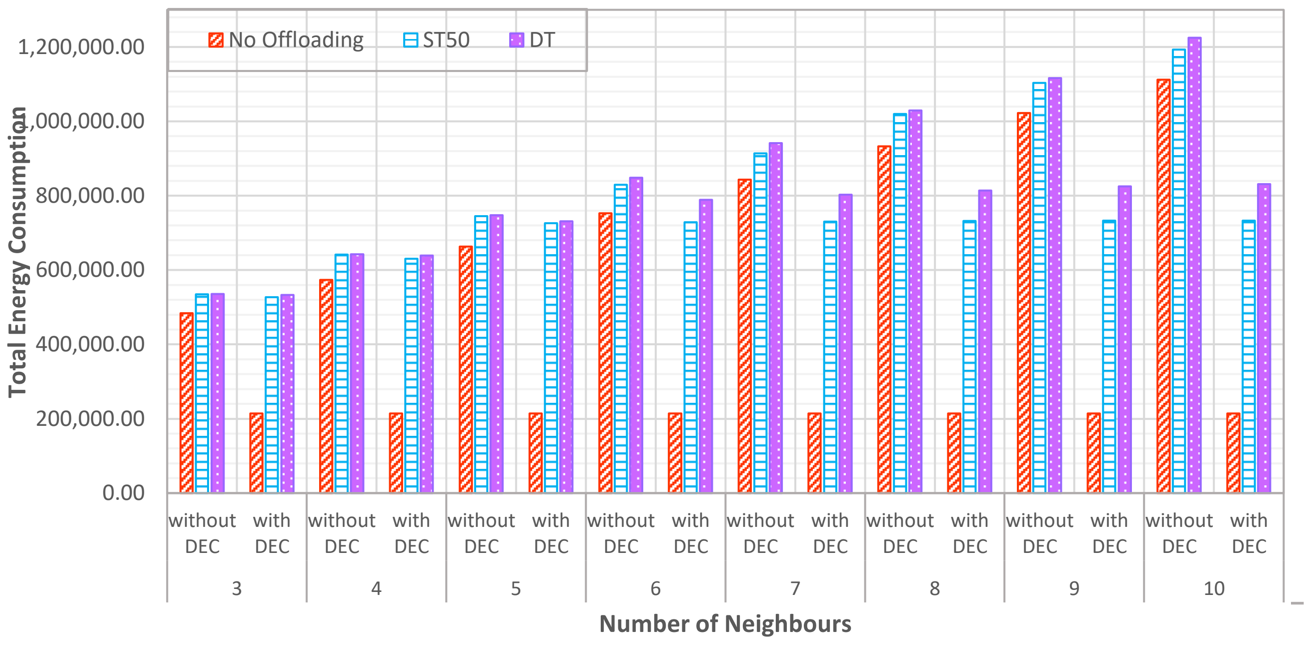


Figure 12: impact of increasing number of neighbours on Energy consumption with various offloading schemes

The impact of increasing the number of neighbours on energy consumption is investigated with various offloading schemes, as shown in Figure 12. When increasing the number of neigbours, the energy consumption in the system is increased as a result of operating additional fog nodes. Adressing the impact of increasing the number of neighbours helps to find the optimal number of neighbouring fog nodes that is necessary to acheive optimum results. This is clear, as when having five neighbours the difference between the energy consumed with and without DEC is very low; then as we increase the number of neighbours, the difference starts to increase. In the no oflloading scheme, the impact of utlising DEC is clear, reducing the wastage of energy by 55.72% when the number of neighbours is three, and up to 80.74% when the number of neighbours is ten. This method can also be applied to ST50 and DT, as DEC saved up to 38.58% and 32.16% respectively when the number of neighbours is ten. When comparing ST50 to DT after applying DEC, it can be seen that more energy is consumed with DT. This is because of the nature of this scheme, as more tasks are processed in DT than ST50, so the energy consumed by processing these tasks causes an increase in overall energy consumption in the system.

7. Conclusion

In this paper, we studied the problem of computational offloading and resource management in online fog computing systems and proposed a dynamic offloading threshold that allows a fog node to adjust its threshold dynamically, with a combination of two efficient and effective algorithms: dynamic task scheduling (DTS) and dynamic energy control (DEC). Various numerical results are included and the performance evaluations were presented to illustrate the effectiveness of the proposed scheme and demonstrate the superior performance over the existing schemes. As future work, one can consider the impact of latency and energy overhead caused by switching on/off fog nodes. Moreover, we will consider task offloading in more complicate deployment with users mobility.

References

1. Alenizi, F. and O. Rana, *Minimising delay and energy in online dynamic fog systems.* Computer Science & Information Technology, 2020. **14**: p. 139-158.
2. Arkian, H.R., A. Diyanat, and A. Pourkhalili, *MIST: Fog-based data analytics scheme with cost-efficient resource provisioning for IoT crowdsensing applications.* Journal of Network and Computer Applications, 2017. **82**: p. 152-165.
3. Sarkar, S. and S. Misra, *Theoretical modelling of fog computing: a green computing paradigm to support IoT applications.* Iet Networks, 2016. **5**(2): p. 23-29.
4. Mahmud, R., R. Kotagiri, and R. Buyya, *Fog computing: A taxonomy, survey and future directions*, in *Internet of everything*. 2018, Springer. p. 103-130.
5. Hu, P., et al., *Survey on fog computing: architecture, key technologies, applications and open issues.* Journal of network and computer applications, 2017. **98**: p. 27-42.
6. Cisco, C., *Fog computing and the Internet of Things: extend the cloud to where the things are.* Электронный ресурс]. URL: https://www. cisco. com/c/dam/en_us/solutions/trends/iot/docs/computing-overview. pdf.(дата обращения: 10.03. 2019), 2015.
7. Ma, K., et al., *An iot-based fog computing model.* Sensors, 2019. **19**(12): p. 2783.
8. Xiao, Y. and M. Krunz. *QoE and power efficiency tradeoff for fog computing networks with fog node cooperation*. in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. 2017. IEEE.
9. Jamil, B., et al., *A job scheduling algorithm for delay and performance optimization in fog computing.* Concurrency and Computation: Practice and Experience, 2020. **32**(7): p. e5581.
10. Dastjerdi, A.V., et al., *Fog computing: Principles, architectures, and applications*, in *Internet of things*. 2016, Elsevier. p. 61-75.
11. Liu, L., et al., *Multiobjective optimization for computation offloading in fog computing.* IEEE Internet of Things Journal, 2017. **5**(1): p. 283-294.
12. Zhao, Z., et al., *On the design of computation offloading in fog radio access networks.* IEEE Transactions on Vehicular Technology, 2019. **68**(7): p. 7136-7149.
13. Zhu, Q., et al., *Task offloading decision in fog computing system.* China Communications, 2017. **14**(11): p. 59-68.
14. Chen, M. and Y. Hao, *Task offloading for mobile edge computing in software defined ultra-dense network.* IEEE Journal on Selected Areas in Communications, 2018. **36**(3): p. 587-597.

15. Al-Khafajiy, M., et al., *Improving fog computing performance via fog-2-fog collaboration.* Future Generation Computer Systems, 2019. **100**: p. 266-280.
16. Yousefpour, A., G. Ishigaki, and J.P. Jue. *Fog computing: Towards minimizing delay in the internet of things*. in *2017 IEEE international conference on edge computing (EDGE)*. 2017. IEEE.
17. Gao, X., et al., *Pora: Predictive offloading and resource allocation in dynamic fog computing systems.* IEEE Internet of Things Journal, 2019.
18. Tang, Q., et al., *Decentralized Computation Offloading in IoT Fog Computing System With Energy Harvesting: A Dec-POMDP Approach.* IEEE Internet of Things Journal, 2020.
19. Wang, Q. and S. Chen, *Latency-minimum offloading decision and resource allocation for fog-enabled Internet of Things networks.* Transactions on Emerging Telecommunications Technologies, 2020: p. e3880.
20. Mukherjee, M., et al. *Joint task offloading and resource allocation for delay-sensitive fog networks*. in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*. 2019. IEEE.
21. Mukherjee, M., et al. *Computation offloading strategy in heterogeneous fog computing with energy and delay constraints*. in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. 2020. IEEE.
22. Sun, H., et al., *Energy and time efficient task offloading and resource allocation on the generic IoT-fog-cloud architecture.* Peer-to-Peer Networking and Applications, 2020. **13**(2): p. 548-563.
23. Meng, X., W. Wang, and Z. Zhang, *Delay-constrained hybrid computation offloading with cloud and fog computing.* IEEE Access, 2017. **5**: p. 21355-21367.
24. Yin, L., J. Luo, and H. Luo, *Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing.* IEEE Transactions on Industrial Informatics, 2018. **14**(10): p. 4712-4721.
25. Mukherjee, M., et al., *Deadline-aware Fair Scheduling for Offloaded Tasks in Fog Computing with Inter-fog Dependency.* IEEE Communications Letters, 2019.
26. Qayyum, T., et al., *FogNetSim++: A toolkit for modeling and simulation of distributed fog environment.* IEEE Access, 2018. **6**: p. 63570-63583.
27. Marsan, M.A. and M. Meo, *Queueing systems to study the energy consumption of a campus WLAN.* Computer networks, 2014. **66**: p. 82-93.
28. Li, F., et al., *A State Transition-Aware Energy-Saving Mechanism for Dense WLANs in Buildings.* IEEE Access, 2017. **5**: p. 25671-25681.
29. Monil, M.A.H., R. Qasim, and R.M. Rahman, *Energy-aware VM Consolidation Approach Using Combination of Heuristics and Migration Control.*
30. Mosa, A. and N.W. Paton, *Optimizing virtual machine placement for energy and SLA in clouds using utility functions.* Journal of Cloud Computing, 2016. **5**(1): p. 17.
31. Mahadevamangalam, S., *Energy-aware adaptation in Cloud datacenters*. 2018.
32. Monil, M.A.H. and R.M. Rahman. *Implementation of modified overload detection technique with VM selection strategies based on heuristics and migration control*. in *2015 IEEE/ACIS 14th International Conference on Computer and Information Science (ICIS)*. 2015. IEEE.
33. Monil, M.A.H. and R.M. Rahman, *VM consolidation approach based on heuristics, fuzzy logic, and migration control.* Journal of Cloud Computing, 2016. **5**(1): p. 8.
34. Abedin, S.F., et al. *A Fog based system model for cooperative IoT node pairing using matching theory*. in *2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. 2015. IEEE.
35. Soleymani, S.A., et al., *A secure trust model based on fuzzy logic in vehicular ad hoc networks with fog computing*. IEEE Access, 2017. **5**: p. 15619-15629.

36. Zohora, F.T., et al. *Enhancing the capabilities of IoT based fog and cloud infrastructures for time sensitive events*. in *2017 International Conference on Electrical Engineering and Computer Science (ICECOS)*. 2017. IEEE.
37. Lee, G., W. Saad, and M. Bennis. *An online secretary framework for fog network formation with minimal latency*. in *2017 IEEE International Conference on Communications (ICC)*. 2017. IEEE.
38. El Kafhali, S., K. Salah, and S.B. Alla. *Performance Evaluation of IoT-Fog-Cloud Deployment for Healthcare Services*. in *2018 4th International Conference on Cloud Computing Technologies and Applications (Cloudtech)*. 2018. IEEE.
39. Liu, L., Z. Chang, and X. Guo, *Socially aware dynamic computation offloading scheme for fog computing system with energy harvesting devices.* IEEE Internet of Things Journal, 2018. **5**(3): p. 1869-1879.
40. Rahbari, D. and M. Nickray. *Scheduling of fog networks with optimized knapsack by symbiotic organisms search*. in *2017 21st Conference of Open Innovations Association (FRUCT)*. 2017. IEEE.
41. Gupta, H., et al., *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, 2017. **47**(9): p. 1275-1296.