(Offloading) QoE-Aware Application Mapping and Energy-Aware Module Placement in Fog Computing + Offloading

Low Choon Keat, Tunku Abdul Rahman University College, Malaysia*
Ang Tan Fong, Monash University, Malaysia
Chun Yong Chong, Monash University, Malaysia
Tew Yiqi, Tunku Abdul Rahman University College, Malaysia

ABSTRACT

Fog computing is a potential solution for the internet of things in close connection with things and end-users. Fog computing will easily transfer sensitive data without delaying distributed devices. Moreover, fog computing is more in real-time streaming applications, sensor networks, IoT, which need high speed and reliable internet connectivity. Due to the heterogeneous and distributed characteristics, finely distributing the task with computation offloading is a challenging task. Developing an efficient QoE-aware application mapping policy is challenging due to the different user interests. The energy consumption would usually increase after such an algorithm and policy are implemented. In this paper, the authors enhanced the future from the previous QoE paper by proposing a computation offloading algorithm. The proposed algorithm is to prevent overloading on fog devices. The proposed solution has been evaluated and compared with other existing solutions. The results show that the proposed solution performs better in terms of execution time, energy consumption, and network usage.

KEYWORDS

Application Mapping, Energy Efficiency, Fog Computing, Module Placement, Quality of Experience

1. INTRODUCTION

Nowadays, the demand to supervise, store, share and analyze huge amounts of complex data (Verma et al., 2016) the increase in the number of end-users who are using Cloud Computing resources, which cause elevated bandwidth usage. The increasing number of end-users, bandwidth usage has been an important element where high bandwidth is required to compensate for such an increasing number of end-users. Consequently, it will lead to a high latency network that restrain other end-users from accessing and uploading data at a reasonable speed which could ultimately result in the cost of usage becoming much more expensive than before. To solve the limitations of cloud computing stated above, fog computing has been introduced. The term “Fog Computing” was coined by Cisco (Stojmenovic & Wen, 2014) which offers enormous facilities to help improve the performance and efficiency of cloud-based solutions, for example by expanding the data centers of cloud computing to the fringe of storage, and computation between end-users and sensors. Clients and associated gadgets (IoT devices included) are immensely expanding at an extremely fast rate as fog computing emerges.

DOI: 10.4018/IJWSR.299017
*Corresponding Author
This article published as an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/) which permits unrestricted use, distribution, and production in any medium, provided the author of the original work and original publication source are properly credited.
Over the years, inward-looking quantitative measures such as the percentage of packets lost and delays are used to determine the quality of services such as computer networks or cloud computing services. These measures used are significant, but with one non-trivial limitation, which is that they solely evaluate the performance of the service without considering any external factor. Therefore, many service providers have shifted their attention from Quality of Service (QoS) to Quality of Experience (QoE) that is exclusively outward-looking and focuses on the customers rather than the service performance. Besides, due to the increasing number of Internet-connected devices, the demand for global computing power is increasing by 20% a year, consuming around 3-5% of the world’s electricity in 2015. A peer-reviewed study in 2016 even shows that the ICT industry could consume 20% of electricity in the world and the carbon emissions could increase up to 5.5% by the year 2025. Therefore, we are motivated to include Quality of Experience (QoE) and an energy-efficient approach in our research.

In this paper, we proposed an application mapping policy with QoE awareness and an energy-efficient strategy to solve the limitations of cloud computing that discourages the clients (especially the end-users) from obtaining data at a faster pace or speed due to slower data speed transmit rate that caused by high latency, while maintaining the energy consumption at an optimal level through energy-efficient strategies. The QoE-aware application mapping involves the use of Fuzzy logic-based approaches and a multi-constraint single objective optimization technique. Degree of Assumption (DoA) and Capacity Class Grade (CCG) are determined to achieve application mapping. For the energy-saving strategy, the incoming tasks (application modules) are allocated to fog devices based on the remaining CPU capacity and energy required. We estimate the minimum energy needed by a module in order to place the modules into the fog devices that can handle the tasks. After that, we propose to use Dynamic Voltage and Frequency Scaling (DVFS) in order to check the available resources and adjust the MIPS value of fog devices accordingly.

2. CHALLENGES

Since QoE deals with users’ level of satisfaction for a given service or product from a service vendor, there is a challenge of developing QoE-aware policy in a real-time and heterogeneous environment (i.e. in a fog computing environment) due to the fact that the users’ interests for different services could be different from one to another and could change from time to time. Other than that, determining which applications to be mapped on which fog instances is also a challenging task as it needs to ensure that users’ QoE gain is maximized while the service QoS is observed. However, in a resource constraint and real-time environment like fog computing, it is difficult to ensure that the final objectives of the proposed policies are not obstructed where the application mapping and calculations carried out will only be performed in a short period of time with the minimum amount of computational effort.

There are additional problems such as energy-saving issues that need to be considered. Energy saving is strongly needed in a fog environment in order to embrace the green computing paradigm. There is always a need for an energy-efficient algorithm that could effectively distribute the tasks to be performed by fog devices and placement of the module in an acceptable performance in terms of network speed and minimization of energy consumption at the same time. QoE-aware policy that does not increase the demand for energy consumption will also be much preferred in the era of green computing. Other than that, due to the fact that the latency of data flows in IoT devices comprise of both communication latency and computing latency that are affected by the base stations’ traffic loads and fog nodes’ computing loads, simply offloading the traffic loads or computing loads through response time optimization is insufficient. While there were some existing studies that focus on simultaneously offloading or balancing the traffic loads and compute loads, this issue has not been addressed appropriately.

3. RELATED WORKS

3.1 Works Related to Placement

According to the paper (Nashaat et al., 2020), an IoT application placement algorithm based on the Multi-Dimensional QoE (MD-QoE) model was suggested by the author. The limitation of this algorithm is that the author did not consider the performance of the proposed algorithm for different types of IoT applications and model more influence factors for QoE.
Besides that, in the paper (Tsipis & Oikonomou, 2020), the author had proposed a distributed policy which is called QoE-Aware Rendering Service Allocation (QoERSA), which takes advantages of the strictly local network inside the fog computing network, towards an optimal delay-sensitive placement, given the QoE gaming constraints and the need for capital investment reduction.

The authors in (Aazam et al., 2019) have implemented fog computing to maximize the usage of Tactile Internet since fog computing can allocate resources based on QoS or QoE requirements. Tactile Internet transmits haptics and provides visual feedback. Therefore, near real-time network connectivity is required for most of the haptics communication in Tactile Internet. This connectivity is commonly known as ultra-reliable low latency communication (URLLC). URLLC requires fast and robust network connectivity with sub-millisecond latency for most of the tactile applications. In this paper, a QoE-based mechanism for dynamic resource allocation in fog computing for Tactile IIoT is proposed.

The author in (Jie et al., 2019) tend to focused on Game-Theoretic Online Resource Allocation Scheme on Fog Computing for Mobile Multimedia Users (MMUs) which is an online resource allocation scheme with respect to deciding the state of the servers in fog nodes separated at different zone on the premise of satisfying the quality of experience (QoE) based on a Stackelberg game. The scheduler decides the states of these fog nodes on the premise of achieving QoE. Faced with the versatility of MMUs and the continuity of requests, the original scheme is expanded by breaking the service time into several slots into an online scheme.

As for the paper(Farooq & Zhu, 2020), the author uses a pricing policy based on the QoE of the applications as a result of the allocation and obtains an optimal dynamics allocation rule based on the statistical information of the computational requests. The drawback of the revenue-maximizing competitive resource allocation and pricing system based on QoE does not take cost and completion time into consideration.

Our proposed QoE-aware application mapping policy differs from the aforementioned works. We have considered multiple users’ assumption parameters such as access rate, demanded resources, and speed. The policy is developed in a decentralized manner so that it gets less prone to the single points of failure and management overheads. Application placement requests are prioritized based on users’ expectations and compound QoE gain of users is maximized through the policy.

The summary of discussed work is presented in Table 1 and Table 2, where the former focus on the main contributions and limitations of the discussed literature, while the latter summarizes the features and strategies used by the authors.

### Table 1. A Summary of Related Work for Placement

| Paper                             | Main Ideas                                                                 | Contribution                                                                 | Limitation                                                                 |
|-----------------------------------|---------------------------------------------------------------------------|-----------------------------------------------------------------------------|---------------------------------------------------------------------------|
| (H. Nashaat et al., 2020)         | - Based on the Multi-Dimensional QoE (MD-QoE) model, an IoT application placement algorithm was proposed. | - Improve overall system output with a small improvement in fog control node power consumption. | - The efficiency of the proposed algorithm for different types of IoT applications and more QoE impact variables need to be modelled was not taken into account. |
| (Athanasios Tsipis et al., 2020)  | - Proposed a QoE-Aware Rendering Service Allocation (QoERSA).              | - Feasible to achieve significant overall service access delay cost reduction, and decrease overall deployment cost for game providers. | - Short of scalability because the need to satisfy the QoE requirements specified. |
| (Mohammad Aazam, 2019)            | - Proposed QoE-based mechanism for dynamic resource allocation in fog computing for Tactile Internet. | - Improves quality and customer satisfaction.                               | - Requires a near real-time network that is highly reliable and responsive for most of the tactile applications. |
| (Yingmo Jie, 2019)                | - Propose an online resource allocation scheme based on a Stackelberg game to determine the state of servers in fog nodes spread in various zones on the premise of satisfying the quality of experience (QoE). | - To minimize running cost used to maintain fog nodes for providing services. | - Did not consider the tradeoff between energy conservation, latency, and coverage efficiency. |
| (Muhammad Junaid Farooq et al., 2020) | - Proposed QoE-based revenue-maximizing dynamic resource allocation and pricing framework for fog-enabled MC-IoT applications. | - An implementable mechanism for allocation of different VMIs available at the fog nodes that result in varying end-to-end delay for user applications. | - Not considering cost and completion time. |
3.2 Works Related to Energy Optimization

Besides QoE, energy consumption is another important concern. Then, the cost of operation for cloud service providers will be affected as well where the cost will increase if the energy consumption is high. In short, the study of energy optimization is relevant in this paper to obtain more information and knowledge regarding the optimization of energy. Table 3 shows a synopsis of several related works and the comparison of features for energy optimization.

In the paper (Chen et al., 2019), in a fog computing environment, the authors proposed an energy optimum dynamic computation offloading system (EDCO) for IIoT. The goal is to reduce energy consumption when computational activities are performed within the desired energy overhead and delay.

Table 3. Platform, criteria involved, and application environment for works related to energy optimization

| Journal Paper              | Platform | Performance | Application Environment |
|----------------------------|----------|-------------|-------------------------|
| (Chen et al., 2020)        | X        | X           | X                       |
| Mahmoud et al., 2018       | X        |             | X                       |
| (M.V. Prakash et al., 2020) | X        | X           | X                       |
| (Tan et al., 2020)         | X        | X           | X                       |
| (Yifan Dong, 2019)         | X        | X           | X                       |
| (Hou et al., 2020)         | X        | X           | X                       |
| (T. Wang et al., 2020)     | X        | X           | X                       |
| (Shahid et al., 2020)      | X        | X           | X                       |

C = Cost, RU = Resource Utilization, ET = Execution Time, SD = Service Delay, EF = Efficiency, IoT = Internet of Things, DCN = Distributed Cloud Networking
Based on the paper (Mahmoud et al., 2018), the authors introduced energy-aware scheduling on the CloudSim which is called the Dynamic Voltage and Frequency Scaling (DVFS). The simulation was conducted to improve the power consumption of different data centers by adding the DVFS and observing the effect between the data centers. The proposed DVFS tools were used to reduce the energy required when the workload is low on the server and scale the voltage and frequency of the CPU dynamically at run-time.

In paper (Prakash et al., 2020), the author proposes a machine learning model to reduce the consumption of energy demand in fog computing Internet of Things (IoT) services. The machine learning model adopts network density, latency, and mobility as the energy constraints and designs an objective function in order to support the lower energy consumption in the network.

In paper (Tan et al., 2020), the author developed an energy-aware strategy and suggested an iterative algorithm based on a critical path that can get the optimal solution in the complexity of polynomial time. The goal is to achieve a viable solution for task allocation with minimal energy usage and restriction to the time limit. The study investigates the integer linear programming model in the fog computing systems for the task scheduling problem. A three-stage fellowship issue is then mapped to the considered issue. To obtain the optimal solution, the author proposes a critical path (CP) based process. The author also provides situations with no viable solution that can be further resolved by involving further money. Finally, the results obtained show that the proposed solution is better and can generate optimal solutions under a heavy workload.

According to paper (Dong et al., 2019), the author concentrated on a policy of energy-effective cooperation between fog nodes to boost the quality of experience (QoE) of users under fairness. A cooperative fog computing system to process offloading workload on the entire Fog layer by data forwarding is built.

Based on paper (Hou et al., 2020), the author proposes new, optimally distributed online approaches to cloud services that are fog-assisted and fog-coordinated. New distributed techniques are designed to refine the distributed programming models’ data dispatching and resource distribution, such as MapReduce, in a completely distributed way, to improve the model energy efficiency while preserving the consistency of the architecture. The proposed fog-assisted cloud platform is certainly able to improve the time-averaged energy efficiency from the experimental dataset and decrease the time-averaged queue length, compared to a fog-coordinated counterpart where fog nodes only dispatch data and do not process the data.

While in paper (Wang, Qiu, Sangaiah et al, 2020), the author designed a novel energy-efficient trustworthy protocol based on mobile fog computing. The use of this protocol is to take both the trust evaluation and data collection into consideration to effectively reduce invalid data and improve the life cycle of IoT systems. The trust evaluation method for nodes based on fog computing. Its mechanism guarantees the reliability of the network nodes and the credibility of the data in our IoT system. The node trust value is to design the trustworthy data collection path of the mobile fog node and also avoid visiting unnecessary sensors and collecting untrustworthy data.

In this literature review (Shahid et al., 2020), the author proposed the popularity-based caching method of fog networks for content delivery, this method is applied content filtration and load balancing. To address the issue of energy efficiency due to the continuous increase in the implementation of large-scale fog networks. This method helps the fog layer help to process important and time-sensitive information in real-time and decreases the delay. A lot of work has been developed to boost the fog network’s energy quality, but there is still a need for more elements to create the fog network itself energy efficient.

Our proposed energy-aware method differs from other works as we combined two optimization modules which are energy-aware module placement and dynamic voltage and frequency scaling (DVFS) technique for energy optimization. Energy-aware module placement aims to improve efficiency by placing modules to the fog device which can fulfill its requirement based on the module’s estimated minimum energy and MIPS. DVFS reduces cost and enhances resource utilization by adjusting the MIPS of fog devices as much as close to the MIPS of module requirement. In fact, by
using our proposed energy-aware method, execution time and service delay can also be reduced due to the improvement of overall efficiency.

3.3 Works Related to Offloading

Offloading is also considered important in improving performance. Therefore, offloading will be considered for research in this section as well. Table 4 shows an overview of several literature reviews and related work comparisons for offloading in different computing paradigms.

### Table 4. A Summary of Related Work for Offloading

| Paper | Main idea | Improved criteria | Limitation |
|-------|-----------|-------------------|------------|
| (M. Yang et al, 2020) | Propose an adaptive multi-objective optimization task scheduling method for fog computing (FOG-AMOSM) | To solve the non-inferior solution set of the utility function index of fog computing task scheduling to try to solve the multi-objective cooperative optimization problem in fog computing task scheduling | A large number of processing machine operations in the resource center will consume large amounts of electric energy which also cause the problem of heat dissipation |
| (S. Wang et al, 2020) | For the characteristics of tasks and resources, a new task scheduling system (I-FASC) is suggested, including an improved fireworks explosion radius detection mechanism | To improve the high resource and balances load | Does not recognize the energy usage of the fog devices processing activity |
| (Z. He et al, 2020) | Proposes two Integer Linear Programming (ILP) models to solve the fog planning issue under the integrated Cloud-Fog (iCloudFog) framework | To efficiently organize the leisure network devices in the network edge to form fog networks (fogs), which are then integrated with the cloud to provide storage and computing resources | Very few works have addressed the fog planning issue |
| (M. Barzegaran et al, 2020) | Implementing Fog Computing Platform (FCP) to bring commutation and communication close to the edge of the network. Proposed a Simulated Annealing metaheuristic algorithm to determine the schedule and partition to solve optimization problems | Provide isolation among mixed-criticality task, guarantee timing requirements | The deviation among the QOC of the control application is minimized |
| (Z. Wang et al, 2020) | Focus on advancing the development of autonomous driving and cooperative traffic management systems. Provide the required computational capabilities to efficiently execute these applications | Improve the computational capacities of the vehicle itself | Is costly |
| (N. D. Nguyen et al, 2020) | To improve the throughput and network latency by proposing a suitable solution called ElasticFog. Propose resource provisioning for container-based fog computing. | Improves the system performance and latency compared to using the default Kubernetes features | The experiment can only be conducted on a small and medium scale but not a larger scale. |
Based on this paper (Yang et al., 2020), the main idea of this paper is to propose an adaptive multi-objective optimization task scheduling method for fog computing (FOG-AMOSM). This algorithm is used to solve the non-inferior solution set of the utility function index of fog computing task scheduling to try to solve the multi-objective cooperative optimization problem in fog computing task scheduling. The FOG-AMOSM algorithm performs in different fog computing situations in which it processes the task scheduling goals of execution time and service cost, it can make the two goals reach the optimal values at the same time.

In this paper (Wang, Zhao, & Pang, 2020), the article is the purpose of a new algorithm that was a firework algorithm to reduce the processing time of the task and provide better overall load balancing for cloud-fog computing. The author proposed the method as a firework algorithm to improve the vulnerabilities. An improved firework algorithm (I-FA) is proposed that implements the fireworks explosive radius prediction function to prevent the absence of an ideal solution. A job scheduling system (I-FASC) is introduced, this approach takes into account job and resource characteristics like an enhanced firework algorithm.

In this paper (He et al., 2020), to optimally provision the huge volume of IoT services with significant diversity, this paper proposes to efficiently organize the leisure network devices in the network edge to form fog networks (fogs), which are then integrated with the cloud to provide storage and computing resources. Specifically, this paper proposes two Integer Linear Programming (ILP) models to solve the fog planning issue under the integrated Cloud-Fog (iCloudFog) framework.

In this paper (Barzegaran et al., 2020), the main objective is implementing the Fog Computing Platform (FCP) as a set of interconnected multicore computing nodes and bringing computation and communication closer to the edge of the network. The author proposed a Simulated Annealing metaheuristic algorithm to determine the schedule and partition to solve optimization problems. These proposed strategies use an Earliest Deadline First simulation.

In this paper (Wang, Zheng, Ge et al., 2020) considers a three-layer VEC architecture and proposes an online offloading scheduling and resource allocation (OOSRA) algorithm to improve the system performance. The author proposed an online computation offloading scheduling algorithm, which is online with low computational complexity, distributed to execute with a small number of control signals needed to be transferred and adaptive to dynamic traffic environments and different applications. The author built an online resource allocation algorithm modified from the First Fit algorithm to allocate computation resources for applications, which can save energy for edge servers. Finally, by extending the network and traffic simulator Veins, the author developed a simulation tool for the VEC system and simulated the OOSRA-VEC system.

Based on this paper (Nguyen et al., 2020), the author implemented ElasticFog that runs on top of the Kubernetes platform and allows resource provisioning for container-based fog computing. By using Kubernetes’ network traffic information and scheduling features in real-time, the author proposed ElasticFog as the elastic resource provisioning method for the applications in container-based fog computing. The goal is to automatically distribute the appropriate quantity of IoT device pods at each location, based on the distribution in real-time of incoming network traffic at each location. In conclusion, combining the features of Kubernetes and ElasticFog can improve the real-time network traffic status and other policies supported by Kubernetes. It also improves the system performance and latency compared to using the default Kubernetes features.

Our proposed computation offloading method can achieve the shortest execution time and least network usage compared with other work. This is because analysis and testing are carried out on the simulation repeatedly to evaluate the performance of our proposed algorithm. In fact, several simulation parameters are followed during the testing such as processing capacity of fog devices, RAM of fog devices and network latency, fog device upstream capability, fog device downstream capability, module size, and tuple size.
4. USE CASE OF FOG ARCHITECTURE

Figure 1 shows the Smart Grid’s Design, is able to compromise power networks by generating power that may execute at the outskirts of devices such as smart meters. Smart Grid also allows the usage of miscellaneous replenished energy sources such as wind power energy, solar energy, biomass energy, hydroelectric, radiant energy, and geothermal energy to supply heterogeneous Internet of Everything (IoE) devices; for example, home appliance, micro-grids, sub-station, and sensor nodes (Baccarelli et al., 2017). Fog computing plays an important role in the Smart grid as the fog layer has the responsibility to manage the large volume of metered data. The fog nodes work in the closeness in terms of geographical distances between each of the edge devices and are rigged with both networking and computing capability. (Baccarelli et al., 2017) In order to assure that the energy has the exact amount and is being stored at the right location seasonably, the flow of the energy must be trustworthy. The fog nodes monitor the quality-of-energy real-time for a quick route and quick exchange of the energy flow.

Figure 1. Internet of Energy application reference architecture (Smart Grid)

5. PROPOSED SOLUTION

5.1 QoE-Aware Application Mapping

Figure 2 shows the architecture for QoE-aware application mapping. QoE-aware application will be the first process in phase one of a proposed solution which reduces the network usage and execution time. In the architecture, For computational fog nodes (CFN), the computational nodes are equipped with resources such as memory, bandwidth, and CPU to run various applications. In computational nodes, the resources are virtualized among MSs, microservices, where an assignment of applications for execution is conducted. Dynamic provision on the additional resources for a microservice can be conducted from either unassigned resources or other MSs without affecting the service quality. In CFN, all configured MSs can be operated independently. The controller node is in charge of monitoring and controlling the overall activities of CFN. There is data stored in the controller node that stores metadata that is related to the running application and State Criteria parameters of the MSs. In the controller node, a Capacity Grade Unit is proposed to define a capacity index for each MS based on the State Criteria parameters to ensure that MSs are ranked in accordance with their competence.
Sometimes, the computation of data signals transmitted from IoT devices is facilitated by edge fog nodes, EFNs. For certain Fog-enabled IoT systems, it is assumed that the corresponding EFNs run the Client Module and aid in placing the subsequent module to CFNs in the upper level. In this approach, the connections are established between EFNs and IoT devices. The Client Module is initiated by the Application Initiation Unit of EFNs, through which a user expresses assumptions related to the application to EFNs. EFN services are used to obtain and collect the capacity index of MSs and it is stored in a data storage. Moreover, the data storage keeps user Assumption Criteria and Quantity of Service (QoS) attributes related to the application for further processing. In EFN, there are two individual units which are, Application Mapping Unit and Assumption Degree Unit. For each application mapping request, the Assumption Degree Unit calculates a priority value by considering user Assumption Criteria. Other than that, the Application Mapping Unit of EFN carries out mapping of applications to appropriate Fog instances according to the priority value of application mapping requests and the capacity index of MSs respectively.

5.1.1 Pseudocode for QoE-Aware Application Mapping

**Algorithm 1** QoE-Aware Application Mapping

1:  \( \text{function } \text{DoA}(\text{bandwidth}, \text{demandedResources}, \text{latencyAcceptability}) \)
2:  \( \text{NormalizedBW} = \text{bandwidth} \) → Normalized bandwidth within \((-1, 1)\)
3:  \( \text{NormalizedDR} = \text{demandedResources} \) → Normalized demanded resources within \((-1, 1)\)
4:  \( \text{NormalizedLA} = \text{latencyAcceptability} \) → Normalized latency acceptability within \((-1, 1)\)
5:     FuzzyBW[i] = NormalizedBW \quad \neg \text{convert}
6:     FuzzyDR[i] = NormalizedDR \quad \neg \text{convert}
7:     FuzzyLA[i] = NormalizedLA \quad \neg \text{convert}
8:     \quad \text{for each} \quad \text{FuzzyBW}[i] = \min..\max \text{ do}
9:     \quad \quad \text{for each} \quad \text{FuzzyDR}[i] = \min..\max \text{ do}
10: \quad \quad \quad \text{for each} \quad \text{FuzzyLA}[i] = \min..\max \text{ do}
11: \quad \quad \quad \quad \text{take the largest among the 3 values then}
12: \quad \quad \quad \quad \text{store in}
13: \quad \quad \quad \quad \text{FuzzyInterference}[i]
14: \quad \quad \quad \quad \text{\text{DefuzzySingleton} +=}
15: \quad \quad \quad \quad \text{\text{FuzzyInterference}[i] * Singleton}
16: \quad \quad \quad \quad \text{\text{Defuzzy} +=}
17: \quad \quad \quad \quad \text{\text{FuzzyInterference}[i]}
18: \quad \quad \quad \quad \text{end}
19: \quad \quad \text{end}
20: \quad \text{return} \quad \text{doaValue} = \frac{\text{DefuzzySingleton}}{\text{Defuzzy}}
21: \text{end function}
22: \text{\text{function}} \quad \text{CCG}(\text{circulationTime}, \text{resourceRequirement}, \text{processingSpeed})
23: \quad \text{NormalizedCT} = \text{circulationTime} \quad \neg
24: \quad \text{\text{Normalized circulation time within (-1, 1)}}
25: \quad \text{NormalizedRR} = \text{resourceRequirement} \quad \neg
26: \quad \text{\text{Normalized resource requirement within (-1, 1)}}
27: \quad \text{NormalizedPS} = \text{processingSpeed} \quad \neg
28: \quad \text{\text{Normalized processing speed within (-1, 1)}}
29: \quad \text{FuzzyCT[i]} = \text{NormalizedCT} \quad \neg \text{convert}
30: \quad \text{FuzzyRR[i]} = \text{NormalizedRR} \quad \neg \text{convert}
31: \quad \text{FuzzyPS[i]} = \text{NormalizedPS} \quad \neg \text{convert}
32: \quad \text{\text{for each} \quad \text{FuzzyCT}[i] = \min..\max \text{ do}}
33: \quad \quad \text{\text{for each} \quad \text{FuzzyRR}[i] = \min..\max \text{ do}}
34: \quad \quad \quad \text{\text{for each} \quad \text{FuzzyPS}[i] = \min..\max \text{ do}}
35: \quad \quad \quad \quad \text{take the smallest among the 3 values then}
36: \quad \quad \quad \quad \text{store in}
37: \quad \quad \quad \quad \text{FuzzyInterference}[i]
38: \quad \quad \quad \quad \text{\text{DefuzzySingleton} +=}
39: \quad \quad \quad \quad \text{\text{FuzzyInterference}[i] * Singleton}
40: \quad \quad \quad \quad \text{\text{Defuzzy} +=}
41: \quad \quad \quad \quad \text{\text{FuzzyInterference}[i]}
42: \quad \quad \quad \quad \text{end}
43: \quad \quad \text{end}
44: \quad \text{end}
Algorithm 5.2: The Pseudocode for QoE-aware application mapping

Algorithm 5.2 shows the pseudocode for QoE-aware application mapping. There are two sections in the code where from line 1 to 19 is used to find the roeValue while from line 20 to 38 is used to find the css Value. Degree of Assumption (DoA) value is important to calculate the application requirement based on three user assumption criteria which are the bandwidth, demanded resources, and latency acceptability while the Capacity Class Grade (CCS) value is used to calculate the Micro Computing Instance Capability based on three state criteria which are the circulation time, available resource and processing speed.

The users end device compromise to the $A_{em} \in \{K_{\alpha}, K_{\beta}, K_{\gamma}\}$ regarding an application $e_m$ to the system through the Application Initiation Unit. The data storage contains the $A_{em}$ and is sent to the Assumption Degree unit of EFN $m$. $A_{em}$ which contains three parameters and the range. The units of the values vary. The values of each parameter are normalized to simplify the further calculation. The result of the normalization will fall in between -1 and 1 by using:

$$K_{em} = 2 \left( \frac{K_{em} - x_{\omega}}{y_{\omega} - x_{\omega}} \right) - 1$$  \hspace{1cm} (1)

$K_{em}$ is the normalized value for criteria $\omega$ within the range $[x_{\omega}, y_{\omega}]$. Each criteria in $[x_{\omega}, y_{\omega}]$ is defined based on the scope for every criteria of Algorithm 5.2 offered in the Fog Environment. In the other words, $x_{\omega}$ refers to the minimum value of the range of parameters, $y_{\omega}$ refers to the maximum value of the range of parameters. In the Assumption Degree Unit, a Fuzzy logic-based approach is used to calculate the $\sigma_{em}$ of each application from the normalized parameter in $A_{em}$.

Fuzzification is the process of converting the crisp input values into fuzzy values by using the information in the knowledge base. In fuzzification, it takes the crisp inputs which are $x$ and $y$ to determine the degree of whether they belong to which of the appropriate fuzzy sets. The standardized value $K_{em}$ of any $A_{em}$ parameter $\omega$ is transformed into an equivalent fuzzy dimension through the associate membership function $\tau_{\omega}$. This work involved membership functions of different Assumption Criteria from three different fuzzy sets.

Figure 3 shows the membership function of fuzzy sets and Figure 4 shows how Fuzzy logic separates the area for Bandwidth fuzzy sets. It is assumed that the value of parameter $\propto$ in application 1 is 2, before the normalization process. Fuzzy sets can have many shapes. However, triangles or trapezoids can usually fully express expert knowledge and can greatly simplify the calculation process.

Figure 4 shows the fuzzy rules used to calculate DoA while the results of fuzzy inputs (Assumption Criteria parameters) are compared based on the fuzzy rules. Fuzzy inference is the process of using fuzzy logic to formulate a mapping from a given input to an output. Then, the mapping will provide the basis from which decisions can be made or patterns can be identified. During fuzzy interference, corresponding fuzzy outputs are determined by mutually comparing fuzzy inputs with the help of fuzzy rules. The fuzzy rules are set in such a way that approximately stringent assumption parameters like large resource demand are given higher weight. As a result, the DoA value for the requests will be more aligned with the stringent assumption parameters compared to flexible parameters like moderate latency acceptable and medium bandwidth. After that, the system needs to be tuned and
evaluated to see if the fuzzy system meets the requirements specified at the beginning. The surfaces can be generated by using the fuzzy logic toolbox to analyze the performance of the system.

The membership degree for each of the fuzzy outputs refers to the highest membership degree of each compared parameter from the corresponding fuzzy set. Figure 5 is based on the range of the parameters for DoA. The equation used to determine the membership degree is shown in Eq. 2:

$$\tau_\alpha(f_{\alpha}) = \left( \tau_\alpha\left(\frac{K_{x}}{K_{y}}\right), \tau_\beta\left(\frac{K_{x}}{K_{y}}\right), \tau_\gamma\left(\frac{K_{x}}{K_{y}}\right) \right)$$  \hspace{1cm} (2)

Figure 5 shows that the bandwidth will be divided into 5 types which are ‘EH’, ‘H’, ‘M’, ‘L’, ‘EL’ and represent ‘Extremely High’, ‘High’, ‘Medium’, ‘Low’ and ‘Extremely Low’. The resource demand is divided into 3 types which are ‘L’, ‘M’, ‘S’ and represent ‘Large’, ‘Medium’, ‘Small’. Using the comparison of the first fuzzy set as an example, with the Extremely Low Bandwidth, Small Demanded Resource, and Low Latency Acceptance, the fuzzy output will be on an Extremely High level.

Figure 6 shows the illustrative explanation on getting fuzzy output for DoA calculation. Any number of fuzzy rules can be triggered based on the Assumption Criteria parameters.

The maximum rating of the application for that fuzzy output is represented by a value called singleton value which is set in a way that could make the logical difference on having fuzzy outputs obviously visible. In this case, the singleton value for fuzzy output is set as ‘Extremely High’,
‘High’, ‘Medium’, ‘Low’ and ‘Extremely Low’ as 10, 8, 6, 4, and 2 respectively. Fuzziness helps us evaluate the rules, but the final output of the fuzzy system must be a clear number. The input of the defuzzification process is the aggregate output fuzzy set, and the output is a single number. For
defuzzification, Fuzzy logic is applied to different parameters of the Assumption Criteria to obtain the exact DoA for application. The equation used to obtain DoA is shown in Eq.3:

\[
\sigma_{e_m} = \frac{\sum_{i=1}^{n} t_a \left(f_{i}^a \right) \times \bigwedge_{i=1}^{m} f_{i}^e}{\sum_{i=1}^{n} t_a \left(f_{i}^a \right)}
\]  

(3)

Each membership degree will be multiplied with the corresponding singleton value, then all the values resulted from the multiplication will be summed up and lastly, be divided by the sum of all membership degrees. \( \sigma_{e_m} \) refers to the exact DoA obtained for application \( e_m \). The Application Mapping Unit will then use \( \sigma_{e_m} \) to map the application to a suitable Fog computing instance.

Similar to the calculation of DoA, corresponding fuzzy output is determined by mutually comparing fuzzy inputs with the help of fuzzy rules during fuzzy interference. However, the fuzzy rules for calculating CCG give higher weight to those approximately impediment state parameters such as long Circulation Time. As a result, the CCG value of the instances indicates more on the limitation instead of the convenience such as adequate Resources Available and fair Processing Time.

Figure 8 shows the fuzzy rules for CCG calculation. Figure 9 shows the calculation time will be divided into 5 types which are ‘ES’, ‘S’, ‘A’, ‘L’, ‘EL’ and represent ‘Extremely Short’, ‘Short’, ‘Average’, ‘Long’ and ‘Extremely Long’. The resource availability is divided into 3 types which are ‘S’, ‘A’, ‘L’ and represent ‘Sufficient’, ‘Adequate’, ‘Limited’. The membership degree for each
Figure 8. Fuzzy rules for CCG calculation

Three-dimensional plots for CCG

Figure 9. Fuzzy rules for CCG calculation

Cube FAM of CCG

Processing Speed
- 127 - 140
- 112 - 127
- 100 - 112
- 87 - 100
- 75 - 87
- 60 - 73
- 47 - 60
- 30 - 47
- 20 - 30
of the fuzzy outputs refers to the lowest membership degree of the compared parameters from the corresponding fuzzy set. The equation used to determine the membership degree is shown in Eq.5:

$$
\tau_c(f_{j_0}) = \left( \tau_0 \left( K_{p_0}^{j_0} \right), \tau_1 \left( K_{p_1}^{j_0} \right), \tau_c \left( K_{c}^{j_0} \right) \right)
$$

(5)

Using the result of the comparison of the first fuzzy set as a reference, with the short Circulation Time, limited Resources Available, and minimal Processing Time, the fuzzy output will be Lower. The illustrative explanation is shown in Figure 10.

Figure 10 shows the Illustrative explanation on getting fuzzy output for CCG calculation. In. The singleton value represents the maximum rating of the application for that fuzzy output. The singleton values for fuzzy output are set as ‘Extremely High’, ‘High’, ‘Medium’, ‘Low’ and ‘Extremely Low’ as 10, 8, 6, 4, and 2 respectively. For defuzzification, the membership degrees generated are combined with an equation to obtain the exact CCG $\Omega_{j_0}$ of the instance. The equation used to obtain CCG is shown in Eq.6:

$$
\Omega_{j_0} = \frac{\sum_{z=1}^{z=\mu} \tau_c(f_{j_0}^{z}) \times \wedge f_{j_0}^{z}}{\sum_{z=1}^{z=\mu} \tau_c(f_{j_0}^{z})}
$$

(6)

Each membership degree will be multiplied with the corresponding singleton value, then all of the values resulting from the multiplication will be summed up and latterly be divided by the sum of all membership degrees. The CCG obtained is then forwarded to the querying EFN to carry out the following application mapping process.

Figure 10. Illustrative explanation on getting fuzzy output for CCG calculation
5.2 Energy-Aware Module Placement

Analytical Modeling of Energy-aware Module Placement

Figure 11 shows the process of placing the incoming modules and it shows clearly the example of the process. Table 5 shows the used parameters. To place the modules, we estimate the minimum energy needed by a module after that place it into the fog device that can handle the modules. As we can see that the module minimum energy needed and MIPS is $M_{\text{mips}}$ and there are 1, 2, 3, ..., n fog devices in which are represented by different maximum energy and MIPS. If the fog devices do not have enough maximum energy and MIPS for the incoming module, the fog device will forward the module to the next fog device until it is able to find another one that is capable of fulfilling the requirement of the module.

Figure 11. Processing of energy-aware module placement

Table 5. Notation of the used parameters

| Parameter     | Definition                                      |
|---------------|-------------------------------------------------|
| $P_{\text{max}}$ | Maximum power                                   |
| $P_s$         | Static power                                     |
| $P_c$         | Constants power                                  |
| $U$           | Utilization                                      |
| $M_{\text{mips}}$ | The MIPS of the incoming module                 |
| $F_{\text{mips}}$ | The available fog devices MIPS                   |
| $\text{minEnergy}$ | The get the minimum energy of incoming module   |
| $\text{frequency}$ | The available frequency value                     |
| $\text{maxFre}$   | The max frequency number (HOST in GHz)           |
5.2.1 Energy-Aware Module Placement Algorithm Design

We used the following formula to perform the energy-aware module placement.

\[
P_{\text{max}} - \frac{P_s}{100}
\]

In order to obtain the constants power value, we use the \( P_{\text{max}} - P_s \) and divided by

\[
100 \text{minEnergy} = P_s + P_c \cdot U
\]

(2)

In this formula, we are using the formula above to do the calculation and get the estimated minimum energy of the module.

\[
U = \text{Min} \left( \frac{M_{\text{mips}}}{F_{\text{mips}}} \right)
\]

(3)

The parameter U shown above is the parameter put inside the Eq.2. The formula for the parameter U uses the function Math.min and compares the 1 and the result in which is Mmips Fmips. In other words, we are using Equation (3) to obtain the current utilization of CPU from U and use the minEnergy to compare with the available energy of the fog device. The process will run iteratively until the module identifies a suitable fog device, which is capable of satisfying the MinEnergy and MIPS of the module.

After the modules are placed in the fog devices, DVFS is performed in order to check if there is any remaining available resource in the fog device. If the fog device still has plenty of remaining MIPS, then DVFS will adjust the MIPS of the fog device into a value that is close to the used MIPS value.

Pseudocode for energy-aware module placement

Algorithm 2 Energy-Aware Module Placement

1: for each FogDevice = closestDevice do
2:     for each AppModule = modules do
3:         estimateConsumedEnergyAfterAllocation
4:         if device suitable for module placement then
5:             Add module to device
6:             print device name
7:             Deployed modules
8:             end if
9:     end if
10:     if devicesname = m-VRGame AND module name = client then
11:         Add module to device
12:         print device name
13:         Deployed modules
14:         end if
15: end
16: for each FogDevice = closestDevices do
17:     if device name contains “cloudlet” then
Algorithm 5.12: Pseudo code for energy-aware module placement

Based on the Algorithm 5.12, the pseudo code for the above energy-aware module placement can be explained in the way that for any fog device that has the closest distance, it will perform the module of estimation of the consumed energy after allocation which is the third line function of the pseudo code. If the device is suitable for the module placement, then it will add the module to the device and print the device name before the modules are deployed, else it will find the upper level of devices for suitable module placement. If the device name equals to m-VRGame and the module name equals to client, then it will add the module to the device and print the device name before the modules are deployed. Lastly, the pseudo code assigns the device as dvfs if the device name contains “cloudlet”.

5.2.2 Dynamic Voltage and Frequency Scaling (DVFS)

Figure 12 shows the processing of DVFS. After the modules which require b MIPS are placed in the fog device with a MIPS, the fog device will have a-b MIPS remaining. If there are no more modules to be placed into this fog device, DVFS will be performed in order to minimize the MIPS remaining in the fog device.

Table 6 shows the frequency values. It will use the five frequency values from the table above to perform DVFS to find out the most suitable MIPS. The five frequency values are 1.70, 2.00, 2.133, 2.60, and 3.00. It uses a frequency value of 1.70 first to perform DVFS. If the result is greater than the used MIPS which is MIPS b, the greater MIPS device will be assigned as the new MIPS. Else, if the result is lesser than using MIPS, then it will use the next frequency value, which is 2.00, and so on to perform DVFS again until the condition is met. For example, in Figure 5.13, the fog device’s new MIPS will be adjusted by using the following formula:

| HOST (GHz) | 1.70 | 2.00 | 2.133 | 2.60 | 3.00 |
|------------|------|------|-------|------|------|
| Frequencies|      |      |       |      |      |
Adjusted MIPS = \( \frac{F_{\text{mips}}}{\text{MaxFre}} \times \text{frequency}[x] \) \hspace{1cm} (4)

DVFS will adjust the MIPS of the selected fog device in which the Fmips are divided by MaxFre and then multiply the frequency [1, 2...x] based on the frequency table. After that, the selected fog device MIPS is minimized as close as possible with the incoming module MIPS.

Pseudocode for DVFS (Dynamic Voltage and Frequency Scaling)

Algorithm 3 DVFS (Dynamic Voltage and Frequency Scaling)

1: \hspace{1cm} set 5 values into frequency array
2: \hspace{1cm} set mips[4] = fog device total MIPS
3: \hspace{1cm} used mips = fog device total MIPS - fog devices available MIPS
4: \hspace{1cm} for \( x < 4 \)
5: \hspace{2cm} set mips[x] = (mips[4] / frequency[4]) \times \text{set frequency}[x]
6: \hspace{1cm} end
7: \hspace{1cm} for \( x \leq 4 \)
8: \hspace{2cm} if used mips < mips[x] then
9: \hspace{3cm} New_mips = mips[x]
10: \hspace{2cm} end if
11: \hspace{1cm} end

Algorithm 5.14: Pseudo code for DVFS

Furthermore, Algorithm 5.14 projects clearly the pseudo code for energy-aware module placement for the third algorithm which is Dynamic Voltage and Frequency Scaling (DVFS). First, set 5 frequency values in orders. Next, set those MIPS values by using the formula \( \frac{F_{\text{mips}}}{\text{MaxFre}} \times \text{frequency}[x] \). After that, if the used MIPS for that particular fog device is lesser than MIPS set, then the new available MIPS for that fog device is equal to MIPS set minus used MIPS. The fog device total MIPS is also set to equal the new MIPS.

5.2.3 Offloading Techniques

Figure 13 shows the network topology simulation. There are proposed two load balancing algorithms intended to complete different tasks respectively which is proximity algorithm Cluster algorithm. The proximity algorithm is to place the application modules on the nearest possible available fog device in order to reduce the delayed transmission. The second algorithm will be the Cluster algorithm that is responsible for placing multiple modules together in the same device to decrease the burden of the network infrastructure due to the intercommunication between the application modules.

The Cluster algorithm’s main goal is minimizing the network bandwidth consumption. Each load balancing algorithm receives the application model as input that contains multiple dependent modules. The algorithm also receives network topology as input which include network devices, sensors and actuators in Figure 13. The network topology consists of Fog devices that have arranged properly into layers depending on their connections with other devices. As a result, the computational power of the devices increases in the network topology.

A tree-like structure has been used for The network topology where Cloud will be the root node and the leaf nodes represent IoT devices which may involve sensors or actuators. The pathway will be a combination of devices from leaf node devices along to the destination, the root node which is
the Cloud. For an example of a pathway from the network topology: EdgeDevice_0_0 → Gateway_0 → ProxyServer → Cloud. The Cluster algorithm has been given as Algorithm 5.16.

Pseudo code for Cluster Algorithm

Algorithm 4 Cluster Algorithm

1: Input: List of Application modules and Fog devices
2: Output: Application module to Device mapping
3: for $p \in \text{PATHS}$ do across all the paths
4:     placedList = {}
5:     load = 0
6:     for module $w \in \text{Application}$ do
7:         if $w$ is already placed on $f \in p$
8:             if $CPU_w^{\text{req}} \leq CPU_f^{\text{avail}}$ then
9:                 place $w$ on device $f$
10:                add $w$ to placedList
11:                break
12:         end if
13:     end if
14:     load += $CPU_w^{\text{req}}$
15: end for
16: for fog device $d \in p$ do leaf to root traversal
17:     if $load \leq CPU_d^{\text{avail}}$ then
18:         for module $w \in (\text{Application} - \text{placedList})$ do
19:             place $w$ on device $d$
20:         end for
21:     end if
22: end for

Figure 13. Network Topology for Simulation
6. RESULTS

The results of the experiments were obtained via testing the Energy Aware Algorithm in iFogSim simulation. We used several scenarios to test the QoE-aware Application Mapping policy and Energy-aware Module Placement algorithm in deploying the application modules to the fog layer. There are three parameters collected, which are the execution times, the energy consumption, and the network usage used to evaluate and compare the performance of several algorithms in a fog computing environment. Table 7 shows the arrangement of the application module of each test scenario and the fog device is arranged with increasing MIPS from the end user towards the cloud.

Table 7 shows some of the simulation parameters, which include processing capacity of fog devices, RAM of fog devices, network latency, fog device upstream capability, fog device downstream capability, module size, and tuple size.

### Table 7. Test Scenario with their respective Application Module Arrangement

| Scenario | Application Module Arrangement |
|----------|-------------------------------|
| Scenario 1 | Module’s MIPS requirements increases from client towards last module |
| Scenario 2 | Module’s MIPS requirements decreases from client towards last module |
| Scenario 3 | Module’s MIPS requirements is in random order between client and last module |

### Table 8. Simulation parameters for QoE-aware application mapping testing

| Parameter                          | Value               |
|------------------------------------|---------------------|
| Processing capacity of fog devices | 350 - 1000 MIPS     |
| RAM of fog devices                | 256 - 512 Mb        |
| Network latency                   | 2 - 100 ms          |
| Fog Device Upstream capability    | 1024000 Mbps        |
| Fog Device Downstream capability  | 1024000 Mbps        |
| Module Size                        | 100 - 600 MIPS      |
| Tuple Size                         | 1000 – 6000 MIPS    |

6.1 Analysis of Algorithms in term of Execution Time

Figure 14 shows the execution time of four algorithms. On the y-axis represents the total execution time to complete the application’s tasks whereas on the x-axis indicates different types of scenarios. The value for each of the algorithms is using the result of execution time of four applications. Based on Figure 14 In scenario 1, the Edgeward algorithm is the second longest to execute, because all modules will be placed on the fog device which is closest to the user, unless the fog device runs out of resources. However, the fog devices which are closest to the user do not have the high specification as the fog devices which are nearer to the cloud. Therefore, more time needed to process. The QoE-aware algorithm allocates the module to the most suitable fog device for processing, so that the task
can be completed within the shortest time in Scenario 1. QoE-aware with energy-aware algorithms uses the third longest execution time, it needs to calculate the lowest power consumption of the fog device then assign the task to the lowest power consumption and suitable fog device. Within scenario 2, all four (4) of the algorithms have the highest value of readings of execution time among other scenarios which module size in scenario 2 will be 600, 500, 400 and 300 separately. While scenario 3 will randomly generate different module sizes unlike scenario 2 with big fixed module size values. In all of the scenarios, QoE-aware and energy-aware placement with Computation Offloading algorithms used the longest execution time whereas QoE-aware algorithms used the shortest execution time. It is because the QoE-aware algorithm first calculates which fog device is suitable to handle the task instead of one by one checking in different fog devices and sends the task to the particular fog device to process. Besides, QoE-aware and energy-aware placement with Computation Offloading algorithms used the longest execution time because the algorithm offloaded the task to other fog devices when the incoming module exceeded the load limit of the current fog device set to prevent the fog device overload. In other words, although a fog device has enough capability to process the specific module size, based on the offloading algorithm, it will set the load limit to the fog device to ensure the fog device would not be occupied fully. This is because the offloading algorithm takes responsibility to ensure every fog device will not be overloaded in order to increase the lifespan of the fog devices. Nevertheless, in these scenarios it has up to 4 apps running. The offloading method does not help to shorten the execution time because all devices are full of jobs. Therefore, in this case fog devices are offloading the tasks but the tasks are offloaded to other busy fog devices and resulting in longer execution time.

6.2 Analysis of Algorithms in term of Total Energy Consumption

Figure 15 shows the total energy consumption of four algorithms. On the y-axis represents the total energy consumption needed to complete the application’s tasks whereas on the x-axis indicates different types of scenarios. The value for each of the algorithms is using the result of energy consumption of four applications. Based on the Figure 15, Edgeward used the highest energy consumption whereas QoE-aware with energy-aware algorithms used the lowest energy consumption. Based on the scenarios, Edgeward has the highest energy consumption among the others. It is caused by the lower layer of the fog devices which the closest to the end devices cannot process the task, it will pass to another that above them, the middle and upper layer fog devices. Therefore, the
characteristics of Edgeward that transfer the modules from one to upper fog device will undoubtedly increase the power consumption.

In scenario 2, every algorithm has higher value compared to other scenarios, which also included Edgeward. It is because the module size in scenario 2 is getting distributed in 600, 500, 400 and 300 respectively. In Edgeward’s perspective, there is a large size module needed to be executed. Although the lower layer of the fog devices has many fog devices, the large module size still cannot be executed completely by the lower layer. The lower layer of fog devices has lower specification, therefore the tasks will be distributed to the middle or upper layer of fog devices which have better specification. In the other cases, the higher the task distributed to, the lesser the fog devices, and the number of the fog devices in the upper layer is limited. By the task distribution of the Edgeward, it has the highest execution time and also the highest energy consumption. Unlike scenario 2 is giving fixed module size value, scenario 3 makes the module size irregular.

For the QoE-aware and QoE-aware with energy-aware algorithms, the energy consumption of the QoE-aware algorithm is higher than the QoE-aware energy-aware algorithm. However, both of the algorithms include QoE application mapping which will transfer the task to the most suitable fog devices directly, which saves up the energy consumption in passing the task to one and another. In scenario 2 that consists of large module size, both of the algorithms are performed well and will pick for the most suitable task distribution of fog devices to process but because of the large module size, the value readings are larger than scenario 1. Although QoE-aware and QoE-aware with energy-aware algorithms are quite similar, due to the combination of energy-aware algorithms, QoE-aware with energy-aware algorithms will be the best algorithm in total energy consumption.

With QoE-aware with energy-aware algorithm and computation offloading algorithm, for the task that can be executed by three (3) fog devices, because of the offloading logic, the task needs to be distributed in even smaller or equal to more fog devices. Therefore, as the transfer rate between fog devices increases, the power consumption of the QoE-aware with energy-aware algorithm and computation offloading algorithm will be increased also.

6.3 Analysis of Algorithms in term of Total Network Usage

Figure 16 shows the total network usage of four algorithms. On the y-axis represents the total network usage needed to complete the application’s tasks whereas on the x-axis indicates different types of scenarios. The value for each of the algorithms is using the result of network usage of four applications.
In scenario 2, all four (4) of the algorithms with module size 600, 500, 400 and 300 severely make them become the highest network usage between all of the scenarios. While scenario 3 will create different module sizes randomly.

Based on the Figure 16, Edgeward and QoE-aware with energy-aware and computation offloading algorithms obviously used more network usage compared to QoE-aware algorithm and QoE-aware algorithm. In Scenario 1 and 3, the QoE-aware with energy-aware algorithm has the least network usage to complete the task. While in Scenario 2, the QoE-aware algorithm has slightly lower network usage than the QoE-aware with energy-aware algorithm. On the other hand, the QoE-aware with energy-aware and computation offloading algorithm used the most network usage in every scenario. Behind the highest network usage algorithm is Edgeward. It has only a little lower network usage behind the QoE-aware with energy-aware and computation offloading algorithm. The main reason why algorithms get higher network usage is because when the task is assigned further away from the sender, more network usage is needed. Exactly for Edgeward and the QoE-aware with energy-aware and computation offloading algorithm, these 2 algorithms will forward the task to other fog devices if the receiver device is unable to process the task. On the other hand, the QoE-aware algorithm and the QoE-aware with energy-aware algorithm will send the task to suitable fog devices to process instead of passing around in the fog environment.

7. CONCLUSION

In cloud computing, optimizing energy consumption is one of the main problems in order to improve its efficiency. In order to address the problem, the fog computing paradigm was proposed by Cisco for providing such services at the network edge. In this paper we proposed a QoE-aware application mapping and energy-aware module placement for placing the incoming module to the fog device. QoE-aware application mapping involves the use of Fuzzy logic-based approaches and a multi-constraint single objective optimization technique. The proposed energy-aware module placement involves the use of Dynamic Voltage and Frequency Scaling (DVFS) algorithm to determine the minimum MIPS in order to reduce the energy consumption. The results show that the execution time of applications with QoE-aware policy was reduced compared to applications without QoE-aware policy. The simulation results show that energy consumptions were reduced with the aid of the proposed
energy aware algorithm, when compared to other existing algorithms. These experimental results demonstrated the effectiveness of the proposed QoE-aware policy in reducing execution time and also showed the importance of analyzing the environment in which DVFS is used. Without having a QoE-aware policy, the DVFS might not necessarily always be efficient in terms of energy saving and will negatively impact the total network usage.

FUNDING AGENCY

Publisher has waived the Open Access publishing fee.
REFERENCES

Aazam, M., Harras, K., & Zeadally, S. (2019). Fog Computing for 5G Tactile Industrial Internet of Things: QoE-Aware Resource Allocation Model. *IEEE Transactions on Industrial Informatics, 15*(5), 3085–3092. doi:10.1109/TII.2019.2902574

Baccarelli, E., Vinueza Naranjo, G. P., Scarpiniti, M., Shojaifar, M., & Abawajy, J. (2017). Fog of Everything: Energy-Efficient Networked Computing Architectures, Research Challenges, and a Case Study. *IEEE Journals & Magazines*. Available at: https://ieeexplore.ieee.org/document/7921687

Barzegaran, M., Cervin, A., & Pop, P. (2020). Performance Optimization of Control Applications on Fog Computing Platforms Using Scheduling and Isolation. *IEEE Access: Practical Innovations, Open Solutions, 8*, 104085–104098. doi:10.1109/ACCESS.2020.2999322

Chen, S., Zheng, Y., Lu, W., Varadarajan, V., & Wang, K. (2019). Energy-optimal dynamic computation offloading for industrial IoT in fog computing. *IEEE Transactions on Green Communications and Networking.*

Dong, Y., Guo, S., Liu, J., & Yang, Y. (2019). Energy-Efficient Fair Cooperation Fog Computing in Mobile Edge Networks for Smart City. *IEEE Internet of Things Journal, 6*(5), 7543–7554. doi:10.1109/JIOT.2019.2901532

Farooq, M. J., & Zhu, Q. (2020). QoE Based Revenue Maximizing Dynamic Resource Allocation and Pricing for Fog-Enabled Mission-Critical IoT Applications. *IEEE Xplore. ieeexplore-ieee-org.tarcez.tarc.edu.my/document/9108583*

He, Z., Zhang, Y., Tak, B., & Peng, L. (2020). Green Fog Planning for Optimal Internet-of-Things Task Scheduling. *IEEE Access: Practical Innovations, Open Solutions, 8*, 1224–1234. doi:10.1109/ACCESS.2019.2961952

Hou, Ni, Zhao, Cheng, Chen, & Chen. (2020). Frequency-Reconfigurable Cloud Versus Fog Computing: An Energy-Efficiency Aspect. *IEEE Transactions on Green Communications and Networking, 4*(1), 221-235. doi:10.1109/TGCN.2019.2953891

Jie, Y., Li, M., Guo, C., & Chen, L. (2019). Game-Theoretic Online Resource Allocation Scheme on Fog Computing for Mobile Multimedia Users. *China Communications, 10.*

Mahmoud, Rodrigues, Saleem, Al-Muhtadi, Kumar, & Korotaev. (2018). *Towards an energy-aware fog-enabled cloud of things for healthcare.* ScienceDirect. Available at: https://www.sciencedirect.com/science/article/pii/S0045790618300399

Nashaat, H., Ahmed, E., & Rizk, R. (2020). IoT Application Placement Algorithm Based on Multi-Dimensional QoE Prioritization Model in Fog Computing Environment. *IEEE Access: Practical Innovations, Open Solutions, 8*, 111253–111264. doi:10.1109/ACCESS.2020.3003249

Nguyen, N. D., Phan, L.-A., Park, D.-H., Kim, S., & Kim, T. (2020). ElasticFog: Elastic Resource Provisioning in Container-Based Fog Computing. *IEEE Access: Practical Innovations, Open Solutions, 8*, 183879–183890. doi:10.1109/ACCESS.2020.3029583

Prakash, M. V., Porkodi, V., Rajanarayanan, S., Khan, M., Ibrahim, B. F., & Sivaram, M. (2020). Improved Conservation of Energy in Fog IOT Services Using Machine Learning Model. *2020 International Conference on Computing and Information Technology (ICICTT-1441)*, 1-4. doi:10.1109/ICICTT-144147971.2020.9213719

Shahid, M., Hameed, A., ul Islam, S., Khattak, H., Din, I., & Rodrigues, J. (2020). Energy and delay efficient fog computing using caching mechanism. *Computer Communications, 154*. 10.1016/j.comcom.2020.03.001

Stojmenovic, I., & Wen, S. (2014). *The Fog computing paradigm: Scenarios and security issues.* IEEE Conference Publication. Available at: https://ieeexplore.ieee.org/document/6932989

Tan, H., Chen, W., Qin, L., Zhu, J., & Huang, H. (2020, August). Energy-aware and Deadline-constrained Task Scheduling in Fog Computing Systems. In 2020 15th International Conference on Computer Science & Education (ICCSE) (pp. 663-668). IEEE. doi:10.1109/ICCSE49874.2020.9201710

Tsipis, A., & Oikonomou, K. (2020). QoE-Aware Rendering Service Allocation in Fog-Assisted Cloud Gaming Environments. *IEEE Xplore. ieeexplore-ieee-org.tarcez.tarc.edu.my/document/9221778*

Verma, M., Bhardwaj, N., & Kumar Yadav, A. (2016). *Real Time Efficient Scheduling Algorithm for Load Balancing in Fog Computing Environment.* Available at: https://www.mecs-press.org/ijitcs/ijitcs-v8-n4/IJITCS-V8-N4-1.pdf

Wang, S., Zhao, T., & Pang, S. (2020). Task Scheduling Algorithm Based on Improved Firework Algorithm in Fog Computing. *IEEE Access: Practical Innovations, Open Solutions, 8*, 32385–32394. doi:10.1109/ACCESS.2020.2973758
Wang, T., Qiu, L., Sangaiah, A. K., Xu, G., & Liu, A. (2020, May). Energy-Efficient and Trustworthy Data Collection Protocol Based on Mobile Fog Computing in Internet of Things. *IEEE Transactions on Industrial Informatics, 16*(5), 3531–3539. doi:10.1109/TII.2019.2920277

Wang, Z., Zheng, S., Ge, Q., & Li, K. (2020). Online Offloading Scheduling and Resource Allocation Algorithms for Vehicular Edge Computing System. *IEEE Access: Practical Innovations, Open Solutions, 8*, 52428–52442. doi:10.1109/ACCESS.2020.2981045

Yang, M., Ma, H., Wei, S., Zeng, Y., Chen, Y., & Hu, Y. (2020). A Multi-Objective Task Scheduling Method for Fog Computing in Cyber-Physical-Social Services. *IEEE Access: Practical Innovations, Open Solutions, 8*, 65085–65095. doi:10.1109/ACCESS.2020.2983742

Chun Yong Chong received the M.Sc. and the Ph.D. degrees in computer science from the University of Malaya, Malaysia, in 2012 and 2016, respectively. He is currently a Lecturer with the School of Information Technology, Monash University Malaysia. His current research interests include software maintenance, software clustering, software remodularization, software fault prediction, and metamorphic testing.