electronics

MDPI

Article

Green Demand Aware Fog Computing: A Prediction-Based Dynamic Resource Provisioning Approach

Dk. Siti Nur Khadhijah Pg. Ali Kumar [1], S. H. Shah Newaz [1,2,*], Fatin Hamadah Rahman [3], Gyu Myoung Lee [4], Gour Karmakar [5] and Thien-Wan Au [1]

1 School of Computing and Informatics, Universiti Teknologi Brunei, Jalan Tungku Link, Gadong BE 1410, Brunei; khadhijahphak@gmail.com (D.S.N.K.P.A.K.); twan.au@utb.edu.bn (T.-W.A.)
2 KAIST Institute for Information Technology Convergence, 291 Daehak-ro, Yuseong-gu, Daejeon 34141, Korea
3 Laksamana College of Business, Bandar Seri Begawan BA 1712, Brunei; fatin@laksamanacollege.edu.bn
4 Faculty of Engineering and Technology, Liverpool John Moores University, Liverpool L3 3AF, UK; g.m.lee@ljmu.ac.uk
5 School of Engineering, Information Technology and Physical Sciences, Federation University Australia, Ballarat, VIC 3350, Australia; gour.karmakar@federation.edu.au
* Correspondence: shah.newaz@utb.edu.bn; Tel.: +673-2461020 (ext. 5235)

Abstract: Fog computing could potentially cause the next paradigm shift by extending cloud services to the edge of the network, bringing resources closer to the end-user. With its close proximity to end-users and its distributed nature, fog computing can significantly reduce latency. With the appearance of more and more latency-stringent applications, in the near future, we will witness an unprecedented amount of demand for fog computing. Undoubtedly, this will lead to an increase in the energy footprint of the network edge and access segments. To reduce energy consumption in fog computing without compromising performance, in this paper we propose the Green-Demand-Aware Fog Computing (GDAFC) solution. Our solution uses a prediction technique to identify the working fog nodes (nodes serve when request arrives), standby fog nodes (nodes take over when the computational capacity of the working fog nodes is no longer sufficient), and idle fog nodes in a fog computing infrastructure. Additionally, it assigns an appropriate sleep interval for the fog nodes, taking into account the delay requirement of the applications. Results obtained based on the mathematical formulation show that our solution can save energy up to 65% without deteriorating the delay requirement performance.

Keywords: broker; energy efficiency; fog computing; computational demand; prediction

Citation: Pg. Ali Kumar, D.S.N.K.; Newaz, S.H.S.; Rahman, F.H.; Lee, G.M.; Karmakar, G.; Au, T.-W. Green Demand Aware Fog Computing: A Prediction-Based Dynamic Resource Provisioning Approach. *Electronics* **2022**, *11*, 608. https://doi.org/10.3390/electronics11040608

Academic Editor: George Angelos Papadopoulos

Received: 31 December 2021
Accepted: 9 February 2022
Published: 16 February 2022

1. Introduction

Green computing has been the ultimate goal in the information and communications technology (ICT) sector for decades. As of 2021, the worldwide use of ICT contributes to 4.7% of electricity consumption [1,2]. With the significant increase in the number of connected devices, the sector's energy consumption will increase gradually. The paradigm shift to cloud computing enables the development of ecologically-friendly internet technology through efficient server utilization, large-scale virtualization, and software stacking [1,2]. In recent years, user applications are also experiencing a shift from simple web browsing, file transfer and email, to low-latency applications, such as audio communication and video conferencing, and currently we are in ultra low-latency era, such as Internet of Things (IoTs), augmented reality, and virtual reality [3]. The delay requirement of these ultra low-latency applications must be kept within the order of milliseconds [4,5]. This is impossible to achieve with the conventional cloud computing architecture due to the centralized internet-based nature of its paradigm. Furthermore, the unprecedented amount of data being generated simultaneously by user equipment (UE) and IoT devices in the

access network every second, all of which will be pushed to the cloud, may cause network bottlenecks and in turn incur additional delays and performance degradations.

The emergence of fog computing is a promising enabler for ultra low-latency applications. Fog computing lies between the cloud and the UE. By leveraging devices at the edge network (called fog nodes) with computational, storage, and networking capabilities, fog extends the functionality of cloud computing to the edge, reducing the distance between each end-user and their "cloud". With fog computing, latency-sensitive tasks can be processed at the fog level, whereas delay-tolerant tasks can be pushed to the cloud. In this way, traffic in the core network and backhaul is reduced and the energy consumption at the cloud is lessened. The same applies to UE and the IoT, especially mobile nodes; because of their close proximity to the fog nodes compared to the cloud, they can offload resource-intensive applications to the fog and only require low transmitting power to send and receive data packets.

Fog devices are not energy-efficient compared to data centers in the cloud. This raises concern, especially for potential battery-powered fog nodes, such as vehicles, portable computers, and mobile phones. Battery is depleted more quickly as a node runs more quickly. Once the battery level reaches zero, a fog node can no longer provide services and thus is considered a failure. Node failure will reduce the overall fog availability and may impose additional delays, which eventually violates the service-level agreement (SLA) of ultra low-latency applications. With the ever-increasing demand for fog computing, the energy consumption in access and edge network segments will undoubtedly rise. This provided our motivation for proposing a green fog computing framework in this paper.

The main objective of this paper is to reduce the energy consumption within the fog network without violating the SLA-specified quality of service (QoS) requirements, particularly the delay requirement of low-latency applications. Our solution in this paper relies upon three principles: traffic prediction, resource allocation and energy conservation. This prediction approach involves building a prediction model based on an autoregressive integrated moving average (ARIMA), which is a supervised learning method, to predict the future number of request arrivals in the deployed fog computing facilities. Resource allocation describes the dynamic allocation of fog nodes for the serving of computational requests from the customer's premises. Inspired by the concept presented in Hadoop [6] which considers two types of nodes, namely, active and standby nodes, this study further extends the idea of fog nodes as either working nodes, standby nodes, or idle nodes. This is based on a prediction output and QoS metrics to ensure that only the required ones are deployed to meet QoS requirements while minimizing power consumption. Energy conservation implements a sleep mechanism on fog nodes with the sleep interval varying depending on their node groups and certain parameters to further reduce energy consumption without violating the QoS requirements of applications with stringent latency requirements.

Numerous studies in the literature have highlighted the significance of ensuring energy efficiency in fog computing, as seen in [7,8]. In [7], the authors have proposed a tree-based fog computing model with the objective of ensuring the energy-efficient distribution of data to servers and fog nodes. Their results show that the total electric energy consumption of nodes in their proposed model is lower than the cloud computing model. In [9], the authors have proposed a trust-based task mapping solution in a fog computing infrastructure. In their solution, based on a trust score, a set of fog nodes forms a logical cluster. A task is assigned to a logical cluster based on its trust level requirement. In cases when the logical cluster is unable to accept any new task due to its capacity saturation, the task is assigned to another logical cluster with a lower trust score (these other logical clusters perform as a backup cluster, catering for the task requests). Meanwhile, the authors in [8] have introduced optimization techniques for fog-assisted and fog-coordinated cloud platforms to improve energy saving performance.

Additionally, realizing the benefits of implementing the trust concept, several studies have included trust in their energy-efficient fog-assisted solutions with the aim of preventing the collection of untrustworthy data from sensor nodes. These studies have been

conducted in body area networks (BANs) [10], wireless sensor networks (WSNs) [11], and IoT applications [12]. However, the focus of trust in the these studies is on the sensor nodes. Although these works have incorporated trust as part of their solutions, they have not considered categorizing the fog nodes into different types to further enhance the energy efficiency of the fog nodes.

In this paper, we propose a prediction-based approach to identify fog nodes as working nodes, standby nodes and idle nodes. Unlike cloud computing, fog computing is more heterogeneous and dynamic in nature, which may lead to security vulnerabilities. Hence, it is important to consider trust in this study. Here, the fog nodes' trust values are evaluated and nodes with high trust values are given priority to serve incoming requests. Our main contributions are as follows:

- We propose a dynamic allocation of fog nodes into three separate groups: working fog nodes, standby fog nodes, and idle fog nodes. Based on predicted load, only the necessary fog nodes are deployed during a specific period to meet the delay requirement of the tasks, whereas the remaining fog nodes are put into a low-power state, i.e., to sleep, to conserve energy.
- Unlike the existing works (e.g., [9]), in which the number of fog nodes deployed for backup is static, this solution proposes to deploy backup fog nodes (standby fog nodes) dynamically.
- Furthermore, to maximize energy savings, this solution proposes separate sleep mode management for the communication interface and processing unit of a fog node; the duration of the sleep interval for the communication interface/processing unit of a fog node depends on its assigned role (i.e., working fog nodes, standby fog nodes, or idle fog nodes) and on the imposed delay requirement.

The rest of the paper is organized as follows: Section 2 explains the related works and summarizes the limitations of the existing studies. Section 3 elaborates our proposed work, system model and algorithms. Section 4 presents the performance evaluation of our proposed solution. Section 5 discusses our observation based on the results and open research issues. Finally, Section 6 concludes the study and states our future avenues of research.

2. Literature Review

In this section, we review the existing works on energy management in fog computing and their energy-saving modes in Section 2.1 and the forecasting of network traffic through machine learning in Section 2.2.

2.1. Energy-Efficient Fog Computing

Various studies have proposed energy-efficient solutions in the fog computing platform. Several energy-efficient techniques for fog computing have been identified for the purpose of energy-aware task offloading, energy-aware fog node placement, energy-aware device control, and energy-aware energy harvesting [13]. A number of studies have focused on energy-aware load balancing in fog environments [14,15]. In [14], the authors proposed an energy-aware load-balancing and scheduling method for equipment in a smart factory using fog nodes. Meanwhile, the load-balancing solution in [15] selects a fog node which has a high energy and high computation availability in order to process a request.

The authors in [8] have proposed optimization methods for fog-assisted and fog-coordinated cloud platforms in order to improve energy conservation performance. In case of a fog-coordinated cloud platform, the raw data among the data centers can be dispatched by the fog nodes. Conversely, in a fog-assisted cloud platform, apart from being able to dispatch raw data among the data centers for parallel processing, the fog nodes take part in processing part of the raw data. Finally, at one of the data centers, the results are gathered to deliver inferences and analytics. In [16], an energy-aware cloud-fog task offloading mechanism has been proposed. During run-time, the solution retrieves information on the real-time applications of all fog devices and determines whether tasks should be offloaded

or not. The authors in [17] aim at optimizing service response time and reducing energy consumption in fog nodes. According to their solution, the fog nodes need to measure the optimal amount of workload to be forwarded to other nearby fog nodes and processed by each other so as to improve performance.

In [18], the authors have proposed an energy and trust-aware virtual machine (VM) allocation solution for vehicle fog computing. Their work aimed to minimize the number of VM migrations and reduce energy consumption in VM processing. Additionally, several studies have focused on categorizing a fog node based on its state in order to improve energy efficiency. The study presented in [19] has proposed a reinforcement-learning-based duty cycling approach to put a set of fog nodes into sleep mode in a cyclic manner without violating QoS constraints. This aimed to achieve a trade-off between power consumption and latency with a lower overhead. The study presented in [20] has introduced an approach in which nodes enter a sleep state once all pending tasks have been processed. When a new task arrives, only then do the sleeping components transition back to their active state. However, immediately switching to a sleep state when no task arrives and to an active state when a new task arrives cause frequent active/sleep transitions, which may lead to the unnecessary consumption of a significant amount of energy. Additionally, during the transition time between the states, a node can neither transmit nor receive a message. This can incur a high latency overhead [21]. Request batching can solve this issue by putting devices in a sleep state until enough requests are present. Once the number of requests reaches the maximum point, devices will enter the active state and resume sleeping when all requests have been processed [22]. However, this will incur additional latency for requests that arrived early and is therefore not feasible for latency-sensitive applications.

Although the above studies have proposed various solutions aiming to ensure an energy-efficient fog environment, including ones that use sleeping mechanisms, none of the studies have considered using idle states for fog nodes. There are cases where keeping the nodes in idle mode is more energy-efficient compared to transitioning them to a sleep state [23,24].

2.2. Forecasting Network Traffic with Machine Learning

As highlighted previously, dynamic provisioning of resources would be ideal in order to avoid QoS violations due to insufficient resources and to prevent the wastage of energy due to under-utilizing the available resources. Determining the right amount of resources, specifically the number of nodes, to be deployed at a given time without violating QoS requirements and energy efficiency is the primary concern. There are two approaches to overcome this issue: either with a reactive response or a proactive response. With the reactive response, the allocated resources change when predefined thresholds are reached or exceeded. On the other hand, the proactive response allocates resources based on the predicted future workload.

In dynamic provisioning, the authors in [25] have introduced a technique using Bayesian probability to predict the number of packets accumulating in an optical line terminal (OLT) when in a sleep or doze state and minimize this number through dynamic bandwidth allocation in a passive optical network. The authors in [26] have applied a k-nearest neighbour regression-based model to predict CPU utilization of VMs and hosts and perform VM migration and consolidation to prevent the system from being overloaded. The authors in [27] have used a seasonal auto-regressive integrated moving average (SARIMA) model for traffic prediction in base stations. The advantage of using SARIMA is that it acknowledges data seasonality, which is a common trait found in network traffic. This way, prediction accuracy is higher. The authors in [28] also have proposed a sleep mode system in a base station using two methods: (1) predicting the traffic load with a prediction window scale corresponding to minutes and activating a resource each time a frontier is reached and (2) pre-determining the number of transmitters to be activated for a predicted average traffic arrival scenario. In the energy-saving area, the work presented in [24]

predicts the length of future CPU idle lengths and selects appropriate sleep states to avoid wasting energy.

3. Proposed Green-Demand-Aware Fog Computing (GDAFC) Solution

In this section, we explain how the proposed GDAFC solution maximizes energy savings while meeting the service requirements of the assigned tasks. Section 3.1 elaborates the system model, whereas Section 3.2 presents the workflow of GDAFC.

Our solution has three main functions. First, based on historical time-series data on task processing request arrival, GDAFC predicts the task processing load in a fog computing infrastructure (FCI), which comprises several fog nodes (see Section 3.2.1). Next, taking account of the predicted task processing load and the delay requirements of services, the required number of working fog nodes (WFNs), standby fog nodes (SFNs) and idle fog nodes (IFNs) are decided for a given time (see Section 3.2.2). Finally, GDAFC sets the appropriate power-saving modes for WFNs, SFNs and IFNs to maximize energy savings, while not compromising with the service requirements of end-users. In our solution, to maximize energy conservation in fog nodes, we propose a sleep mode for the communication interface (CI) and processing unit (PU), which includes the processor and memory, of a fog node. We elaborate this in more detail in Section 3.2.3.

3.1. System Model

Figure 1 illustrates the proposed system model of GDAFC. In this solution, we assume that a broker function, which is located in one of the fog nodes or in the cloud, is responsible for coordinating the overall operation of the FCI. We further assume that the broker, fog nodes, and end-users (e.g., IoTs) communicate via wireless connections, whereas the broker and the cloud communicate through wired or wireless connections. Next, we explain the operational assumptions of the broker, fog nodes, and end-users in the GDAFC solution.

- **Broker**: In this study, a broker is used as an intermediary entity to facilitate the collaboration between the core network and the fog nodes in the edge network. The broker acts as the coordinator for the fog computing, responsible for providing fog services to the interested customers (IoTs) by renting resources from the eligible fog nodes. In particular, the rented resources are used to execute latency-sensitive cloud services that the cloud itself cannot perform without violating the QoS requirements.
- **Fog nodes**: We assume that computing resources of fog nodes are purchased by the broker to deliver fog services to the UEs and IoTs. The fog nodes need to meet a set of requirements to be part of an FCI. We consider that the prime criteria for a fog node to be part of an FCI are having sufficient storage and computing resources, as well as the capability to communicate through communication interface(s) and move to a low-power state to conserve energy. The authors in [29] implemented Cloudy software (http://cloudy.community/, accessed on 31 December 2021) technologies in home gateways in order to facilitate FCI. Similarly to [29], in order to model end-users' computing resources, e.g., personal computer (PC), as part of an FCI, we assume that Cloudy software is installed on all the fog nodes, enabling cloud computing to be conducted at the edge; hence, the PCs are transformed into fog nodes.
  As the fog nodes need to be able to sleep to reduce energy consumption, having the ability to switch to low-power mode is a must. A system in an idle state consumes a lot of energy despite doing nothing; thus, in our fog scenario, we use the C-States low-power mode, whereby the processor can be put to sleep when idle [30]. This is the standardized power-saving mode of the ACPI specification (http://www.acpi.info/spec50a.htm, accessed on 31 December 2021) for computer systems. The reason we choose C-States is because the wake-up latency (transition overhead) is well below the delay requirement of low-latency applications and the power reduction is significant. Here, we assume that the fog nodes will only go to sleep when there is no incoming request that will not exceed the delay requirement. While being deployed to provide fog service, each fog node will monitor and record its own request traffic, calculating how

many requests it received from the customer premises before the next re-configuration period (RECON). When no request arrives, the fog node will transition to sleep mode for a specific amount of time before waking up and listening for new request. Once a new request arrives, it will remain in an active state, process the request and listen for another requests.

- **End-users**: IoTs and UEs are collectively referred to as end-users in this paper, catering to different sorts of applications including those that have very stringent latency requirements. To be connected to a fog service, these devices must first send a connection request and define their expected level of service to the broker. Once SLA has been achieved, devices are permitted to send tasks to and receive responses from the nearby fog nodes of an FCI.

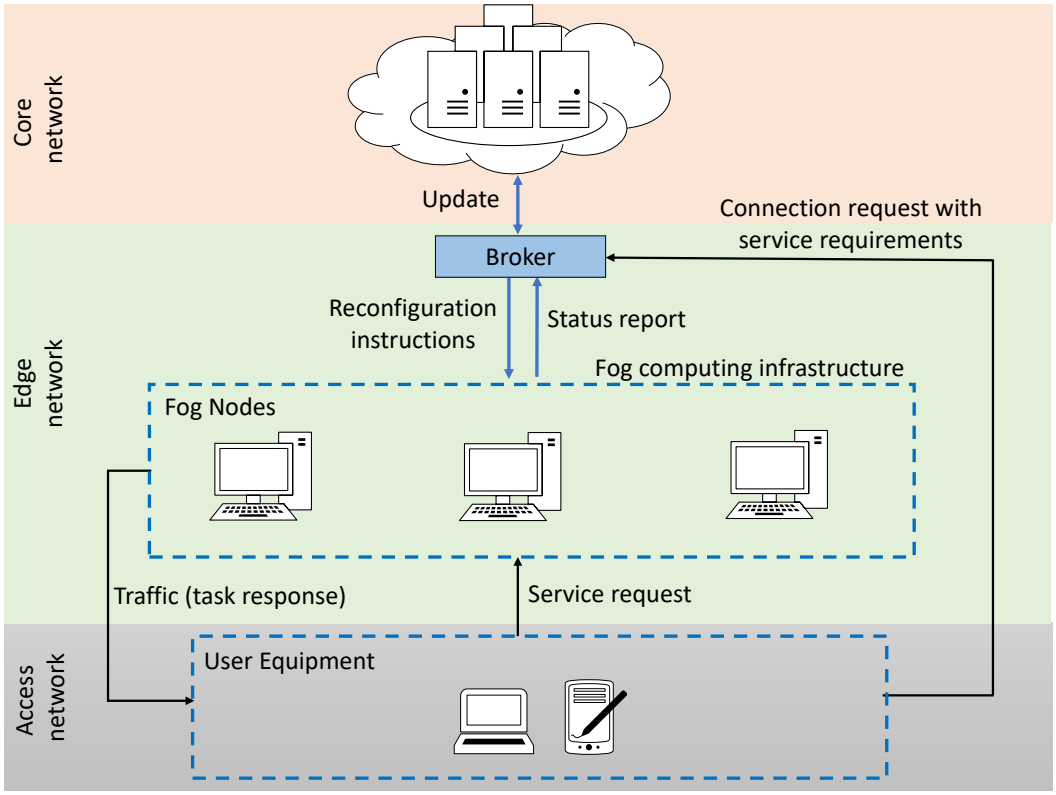


Figure 1. System model of GDAFC.

3.2. Proposed Workflow in GDAFC

GDAFC aims to reduce energy consumption in a fog paradigm without violating the SLA-specified QoS requirements. To simplify our research, we only consider two QoS metrics: the trustworthiness and delay requirements of application. The trustworthiness of a fog network addresses several QoS factors, such as, but not limited to, availability and reliability. A fog network must have high trust in order to meet the performance agreed to in the SLA. An FCI made up of fog nodes with low trust will bring down the overall fog performance and will not be able to meet the QoS. The authors in [9,18,31] proposed a broker-based trust evaluation approach to finding trustworthy fog nodes. Since our system model is also broker-based, we apply a similar approach in our proposed algorithms. For queuing delays, we address the amount of time spent for each request arriving at one fog node to wait in a queue before being served. The queuing delay is directly affected by the number of available fog nodes and the request arrival rate; the higher the request arrival rate and the fewer fog nodes are available, the higher the queuing delay, and vice versa.

Figure 2 illustrates the overall proposed operational procedures in GDAFC. In our solution, the number of WFNs, SFNs and IFNs are decided after every inter-reconfiguration period (I_{re}), which is known by both fog nodes. After I_{re}, the reconfiguration period starts,

which has a duration of t_{re}. After I_{re}, all fog nodes must be in an active state in order for them to send their current status and traffic load, and receive their new assigned role and sleep interval ①. In the reconfiguration process during t_{re}, the broker polls each of the fog nodes for their traffic load and current status ②. Next, fog nodes send a reply message containing the traffic load (e.g., computation resource availability) and their current status ③. With this information, the broker then executes its algorithms, which we explain in the subsequent part of this section, and allocates to each fog node their respective role and power-saving operation. In particular, at this stage, the broker has three major roles: load prediction, estimating the number of required WFNs and SFNs, and appropriating power-saving techniques for the WFNs, SFNs and IFNs. We explain these roles of the broker at this stage in more detail below:

- **Network load prediction:** Using time-series analysis, historical network traffic traces are analyzed and processed to train our prediction model. Using the model, the broker predicts the total incoming requests for the next I_{re}. This will be elaborated further in Section 3.2.1.
- **Dynamic allocation of fog nodes:** Based on the predicted number of request arrivals in the next I_{re}, the broker allocates the fog nodes available in the system into three possible role groups: WFNs, SFNs and IFNs. This will be elaborated further in Section 3.2.2.
- **Determining appropriate power-saving techniques:** We assume that the fog nodes can selectively turn on/off their CI and PU whenever required. Once fog nodes are allocated to their respective groups, the broker determines the appropriate sleep mode for the CI and PU of the WFNs, SFNs and IFNs. Next, it notifies the fog nodes their respective role and power-saving-related parameters (e.g., sleep interval length and sleep state of the processing unit) ④; see an example illustration in Figure 3. In reply, the fog nodes send an acknowledgment (ACK) message to the broker ⑤. The energy conservation operation of the fog nodes will be elaborated further in Section 3.2.3.

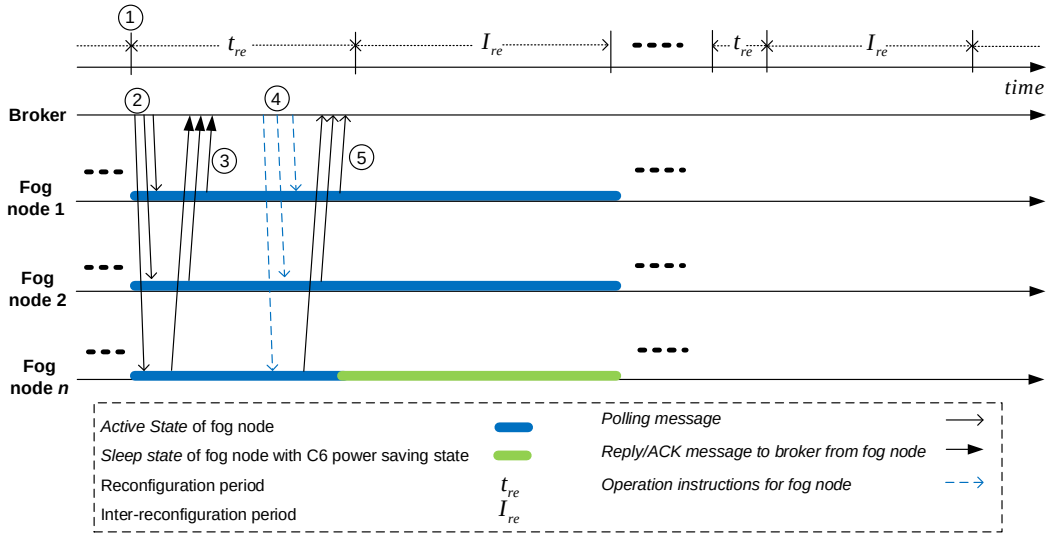


Figure 2. The processes occurring between fog nodes and the broker in an FCI.

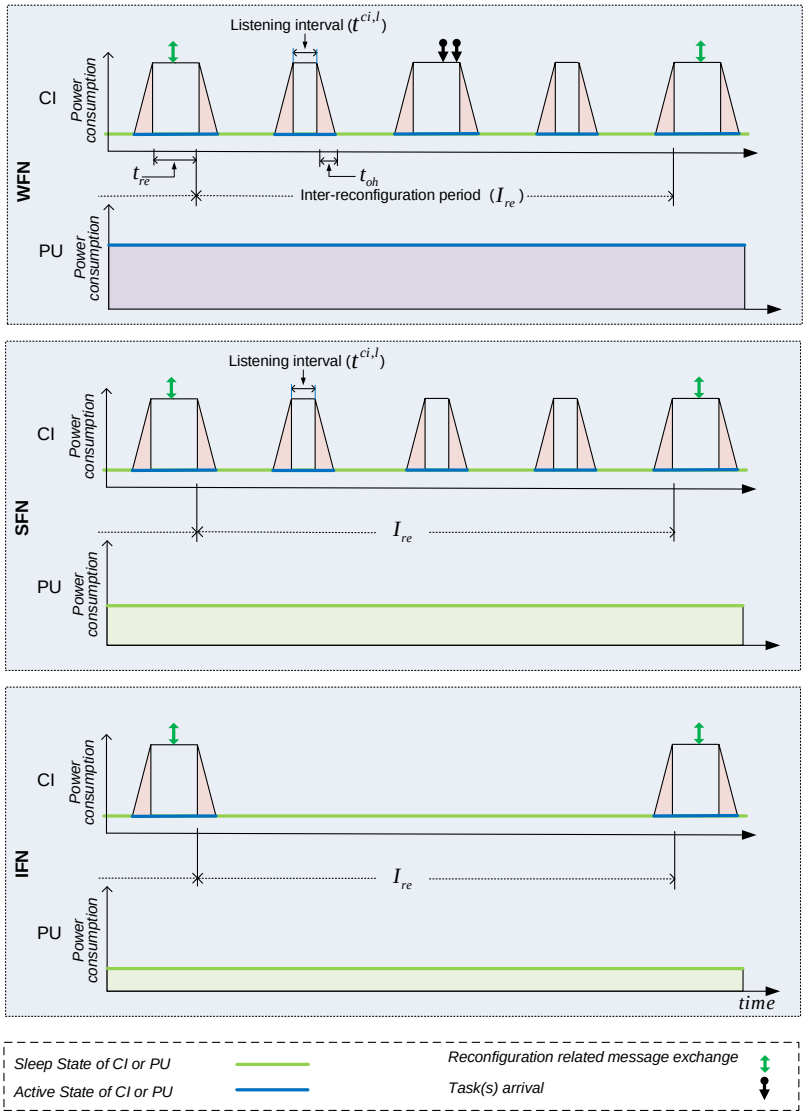


Figure 3. Energy-saving operations of a WFN, SFN and IFN in the proposed GDAFC.

3.2.1. Network Load Prediction

As the static allocation of fog nodes would be inefficient, excessive resources available during low demand results in energy wastage, whereas insufficient resources during high demand will violate the delay requirements of applications. Dynamic provisioning of resources is an ideal approach as network behavior tends to fluctuate throughout the day. Dynamic provisioning actively or passively adapts the provisioning of machines based on the current or predicted future workload [32]. Our focus is on the latter; hence, we implement proactive dynamic provisioning in our solution. In order to realize this, an ARIMA-based prediction technique is used here to estimate upcoming workloads, which in turn is used to determine the amount of required resources at a given time and the most appropriate energy-saving technique of the CI and PU of each fog node in relation to the request arrival rate. With the location-awareness characteristics of fog computing, service demands of mobile users are usually predictable [33]. For simplicity, we assume that all requests arriving at our fog network require the same amount of resources.

3.2.2. Dynamic Allocation of Fog Nodes

GDAFC deploys a certain number of fog nodes, called WFNs, that can serve the task-processing load without violating the QoS requirements. The remaining fog nodes will

be further assigned to be either standby nodes or idle nodes (i.e., WFNs and IFNs). These three types of fog nodes are explained further below. The SFNs are necessary to tackle the issue of under-estimating the actual network load. We define the beginning of the current reconfiguration period as T_i and the next one as T_{i+1}, where $T_{i+1} = T_i + t_{re} + I_{re}$.

Based on task processing request arrivals between T_i and T_{i+1}, GDAFC predicts the future workload for time T_{i+1} to T_{i+2}. The total number of fog nodes in the system is denoted as FN_{max}. Each fog node in the network is assigned with an identifier. F is a set of fog nodes' identifiers where $F = \{f_1, f_2, \ldots, f_n, \ldots, f_{FN_{max}}\}$, where f_n indicates the n-th fog node in the network. Next, we assign a node score, ranging from 0 to 1, to each fog node in the system by applying a fuzzy logic technique [34,35]. Following the work presented in [31], we evaluate our fog nodes based on their trustworthiness. Nodes with node scores close to one indicate that they have higher trust levels, whereas those with scores close to zero have lower trust levels. These nodes are then ranked from the best to the worst according to their scores. This ensures that nodes with high scores are given more priority to be deployed and serve incoming requests. $FN_{rank} = \{f_{x_1}, f_{x_2}, \ldots, f_{x_n}, \ldots, f_{x_N}\}$, where fog node f_{x_1} has the highest score and fog node f_{x_N} has the lowest score, $f_{x_n} \in F$ and $1 \leq x_n \leq FN_{max}$. $FN_{score} = \{S_{f_{x_1}}, S_{f_{x_2}}, \ldots, S_{f_{x_n}}, \ldots, S_{f_{x_N}}\}$ contains the scores corresponding to their respective nodes. That is, $S_{f_{x_n}}$ is the score for node f_{x_n}.

- **Working fog nodes**: Working fog nodes (WFNs) are the fog nodes that are expected to serve the requests arriving at the fog network. Here, our objective is to find the minimum number of fog nodes from FN_{rank} that are sufficient to meet the demand for task-processing requests, measured using the prediction model based on ARIMA. In order to attain this objective, we first measure the total delay that a request may experience for a given task arrival rate.

$$D_{cal} = D_{perNode} + D_{forward}, \tag{1}$$

where $D_{perNode}$ is the service delay per fog node, which is measured using (2), and $D_{forward}$ is the task-forwarding delay to a user.

$$D_{perNode} = \frac{\lambda \times (\frac{1}{\mu_i})^2}{2(1 - \frac{\lambda}{\mu_i})}, \tag{2}$$

where, λ is the request arrival rate and μ is the task-processing service time at a fog node. To find the optimal WFN, we propose Algorithm 1, which returns an optimal number of WFNs from FN_{rank} that are sufficient to meet the imposed delay requirement (DR) for a given task request arrival.
In Algorithm 1, initially, the D_{cal} of current $WFN_{T_{i+1}}$ is measured using (1), taking into account the λ and μ. Next, the $WFN_{T_{i+1}}$ is either incremented or decremented depending on the current workload trend ($TREND_T$) and the predicted workload trend ($TREND_{T+1}$). If both trends are either increasing or fluctuating and $D_{cal} > DR$, then $WFN_{T_{i+1}}$ is incremented until $D_{cal} < DR$ to ensure optimal performance. Otherwise, if decreasing trends are observed from T to $T+1$, and $D_{cal} < DR$, the algorithm will attempt to reduce $WFN_{T_{i+1}}$ to minimize the number of working nodes in order to conserve energy, while not exceeding DR. Since our objective is to deploy enough working nodes for the incoming rate of requests, we assume that the request arrivals at each fog node during a time period are independent and follow a Poisson distribution [36]. We apply the $M/G/1$ system in (2) to calculate the waiting time in the queue.
- **Standby fog nodes**: As the provisioning of fog nodes is highly dependent on the output of the prediction model, having 100% prediction accuracy is essential to ensure that the right number of fog nodes are deployed in the network to meet the SLA-specified QoS requirements. However, our prediction model is based on ARIMA, which is a statistical model, and statistical models generally produce an average level

performance, indicating that they are not able to yield optimum performance at all times [37]. Note that the prediction model could over-estimate or under-estimate the actual workload. Addressing the latter is our main concern as a lack of resources available in the network when demand is higher than expected could degrade performance in relation to delay requirements. Therefore, deploying extra fog nodes (SFNs) alongside the WFNs would be an ideal solution. We use Algorithm 3 to measure the number of SFNs required.

- **Idle fog nodes**: After the working nodes and the standby nodes have been determined, the remaining fog nodes, if any, are set as idle nodes. This is measured simply using Algorithm 4.

Algorithm 1: $find\_WN$: determining the number of WFNs.

Data: $\lambda, FN_{max}, FN_{rank}, FN_{score}, WN_T, TREND_T, TREND_{T+1}$
Data: $DR, D_{prop}, LR, P_{size}$
Result: $WFN_{T_{i+1}}$
initialize $FN_T = WFN_{T_i}, A_T = 0, FN_{min} = 0, D_{cal} = 0$;
$WN_T = 0$;
$D_{cal} = find\_D_{cal}(\lambda, FN_{rank}[1 : WFN_{T_{i+1}}], \mu)$ /*calling Algorithm 2, which implements (2)*/;
while $D_{cal} \neq DR$ **do**
  $D_{cal} = find\_D_{cal}(\lambda, FN_{rank}[1 : WFN_{T_{i+1}}], \mu)$ /*calling Algorithm 2*/;
  if $TREND_T = 0$ *and* $TREND_{T+1} = 0$ **then**
    /*0 indicates decreasing trend*/;
    if $D_{cal} < DR$ **then**
      decrement $WFN_{T_{i+1}}$ by 1;
    end
  else
    if $D_{cal} > DR$ **then**
      increment $WFN_{T_{i+1}}$ by 1;
      if $WFN_{T_{i+1}} > FN_{max}$ **then**
        $WFN_{T_{i+1}} = FN_{max}$;
      end
    end
  end
end
return $WFN_{T_{i+1}}$

Algorithm 2: $find\_D_{cal}(\lambda, FN_{curr}, \mu)$: calculate the expected delay in the system with current number of fog nodes.

Data: λ, FN_{curr}, μ;
Result: D_{cal};
initialize $D_{cal} = 0, D_{perNode} = 0$;
$D_{perNode} = \frac{\lambda \times (\frac{1}{FN_{curr} \cdot \mu})^2}{2(1 - \frac{\lambda}{FN_{curr} \cdot \mu})}$;
$D_{cal} = D_{perNode} + D_{forward}$;
return D_{cal}

Algorithm 3: $find\_SFN$: determining the number of SFNs.

Data: $ERR_T, WFN_{T+1}, FN_{max}, Num_{SFN}, FN_{rank}, FN_{score}$;
Result: SN_{T+1};
$SFN_{T+1} = 0, Num_{SFN} = 0, D_{cal} = 0$;
$Potential_{SFN} = FN_{max} - WFN_{T+1}$;
if $WFN_{T+1} == FN_{max}$ *or* $ERR_T < 0$ **then**
  $SFN_{T+1} = 0$;
else
  λ_{err} = request arrival rate of ERR_T using Poisson;
  while $Num_{SFN} \neq Potential_{SFN}$ **do**
    $Num_{SFN} = Num_{SFN} + 1$;
    $D_{cal} = find\_D_{cal}(\lambda_{err}, Num_{SFN}, \mu)$ /*calling Algorithm 2*/;
    if $D_{cal} < DR$ **then**
      $SFN_{T+1} = Num_{SFN}$;
      break;
  end
end
return SFN_{T+1}

Algorithm 4: $find\_IFN$: determining the number of IFNs.

Data: $WFN_{T_{i+1}}, SFN_{T_{i+1}}, FN_{max}$
Result: $IFN_{T_{i+1}}$
initialize $IFN_{T_{i+1}} = 0$;
$IFN_{T_{i+1}} = FN_{max} - (WFN_{T_{i+1}} + SFN_{T_{i+1}})$;
return $IFN_{T_{i+1}}$

3.2.3. Energy Conservation Operation in Fog Nodes

The motivation behind introducing a sleep mode in an FCI is to minimize energy consumption as much as possible. However, sleep mode introduces a trade-off between energy savings and the QoS requirements of applications. A longer sleep length could save more energy but at the cost of increasing delays, which may degrade service performance. On the other hand, a shorter sleep interval could reduce service delays but increase energy consumption. Therefore, finding an optimal sleep interval length is imperative. We propose that the PU and CI of a fog node can maintain a sleep mode. Therefore, we consider that the CI of a fog node triggers a sleeping PU of the fog node in the case that a task processing request arrives. Furthermore, we assume that the PU of a fog node uses a C-States sleep mode, as this mode has a short wake-up latency and can provide significant reductions in energy consumption.

We assume that the FCI provider sets an optimal delay requirement for a set of fog nodes in order to cater to certain types of applications that require similar QoS requirements. Let DR be the delay requirement set by an FCI provider. To ensure that the service delay is not more than DR, the maximum sleep interval (SL_{max}) should be less than the DR. When a fog node maintains sleep mode, it needs to be periodically available to its broker to obtain instructions. The amount of time a fog node spends listening to the broker's instruction is referred to as $t^{ci,l}$. Both CI and PU require to a certain amount of time to move into an active state. We refer to this required transition time for PU and CI as t_{oh}. Note that when the interface of the fog node is in a sleep state and transiting from a sleep to an active state, the broker cannot reach it. We measure the sleep interval of the communication interface of a fog-device as follows:

$$t^{ci,s} = DR - 2 \cdot t_{oh} - t^{ci,l}. \tag{3}$$

The number of working nodes deployed are the fog nodes expected to serve the predicted request arrivals within R; therefore, it is plausible to use (3) to determine their sleep interval. As for the standby fog nodes, the number is determined by how far the

prediction model under-estimated the actual load during the previous RECON. With that knowledge, there is a possibility that during the current RECON, the prediction model may under-estimate the load as well. Previous work by [38] tackled the under-estimated issue by adding a 10% safety margin that keeps some nodes in idle state. This enabled them to avoid extra latency due to the need to wake up machines upon the arrival of any unexpected requests. Therefore, it is necessary to deploy standby nodes and working nodes together. Hence, we determine the sleep interval of the communication interface for standby and working nodes using (3).

As for IFNs, these are the nodes that are not required for computation purposes in an FCI. Therefore, we propose to put their CI into a sleep state during the entire RECON period and for the PU to remain in the sleep state as long as there are no arrivals of computational requests. Furthermore, we expect that SFNs are sufficient to accommodate the unexpected request arrivals.

3.3. Energy Consumption Measurements in GDAFC

In order to quantify the total energy consumption of each WFN, we first measure the total energy consumption of the communication interface of a WFN during a I_{re}. The number of times a fog node switches between its sleep and active state during I_{re} is $v = \frac{I_{re}}{DR}$. Therefore, the energy consumption of the CI of a j-th WFN during I_{re} is given by

$$E_j^{WFN,ci} = \sum_{k=1}^{v} \left(t^{ci,s} \cdot p^{ci,s} + \left(2t_{oh} + t^{ci,l} \right) p^{ci,a} \right), \tag{4}$$

where $t^{ci,s}$, $p^{ci,s}$, t_{oh}, $t^{ci,l}$ and $p^{ci,a}$ indicate the sleep interval of the CI, the power consumption of the CI during its sleep state, the transition overhead, the listening interval for receiving an instruction and the power consumption of the CI during its active state, respectively. Next, we formulate the equation for calculating the total energy consumption of all WFNs, M_{WN}, during I_{re}, taking into account their energy consumption at CI and PU.

$$E_{WFN} = \sum_{i=1}^{M_{WN}} \left(I_{re} \cdot p_i^{PU,a} + E_i^{WFN,ci} \right), \tag{5}$$

where $p^{PU,a}$ is the power consumption of the PU of the i-th WFN when it is in an active state. In our solution, the WFNs and SFNs need to maintain the same sleep interval at their communication interface during I_{re}. Therefore, we assume that the energy consumption of the i-th SFN, $E^{WFN,ci} = E^{SFN,ci}$. Thus, we formulate the equation for calculating the total energy consumption during I_{re} of all SFNs, M_{SN}, taking into account their energy consumption at CI and PU.

$$E_{SFN} = \sum_{i=1}^{M_{SN}} \left(I_{re} \cdot p_i^{PU,s} + E_i^{SFN,ci} \right), \tag{6}$$

where $p^{PU,s}$ is the power consumption of the PU of the i-th SFN when it is in the C-state. Now, we formulate the equation for measuring the energy consumption of all IFNs under the broker during the I_{re}. Note that the IFNs keep their communication interface in a sleep state during $I_{re} - 2t_{oh}$. Therefore, the energy consumption at the CI of each IFN during I_{re} is expressed as follows:

$$E^{IFN,ci} = (I_{re} - 2t_{oh}) p^{ci,s} + 2t_{oh} \cdot p^{ci,a}. \tag{7}$$

The total energy consumption during I_{re} of all the IFNs, M_{IN}, taking into account their energy consumption at CI and PU, is as follows:

$$E_{IFN} = \sum_{i=1}^{M_{IN}} \left(I_{re} \cdot p_i^{PU,s} \cdot \varsigma + E_i^{IFN,ci} \right), \tag{8}$$

where ς is the weightage of the power consumption reduction in a low-power state compared to $p_i^{PU,s}$. Finally, by summing (5), (6) and (8), we can obtain the total energy consumption of FN_{max} under a fog broker.

4. Performance Evaluation

In this section, we present a comprehensive evaluation of our proposed solution to demonstrate whether energy consumption in fog paradigm can be reduced using our proposed GDAFC without violating the delay requirement. Since we built the prediction model based on already-exist ARIMA time series technique, we do not evaluate the performance of our prediction model in this section. We assume, in order to conserve energy, that our fog nodes transition from a *C0 state* (active state) to a *C6 state* (sleep state) when no request arrives during a $t^{ci,l}$. For simplicity, we only consider homogeneous fog nodes; we use immobile personal computers (PCs), as considered in [29]. Table 1 summarizes the parameters used in the performance evaluation of proposed GDAFC.

Table 1. Parameters used in performance analysis.

Parameters	Values
$D_{forward}$	1.5 ms
FN_{max}	600 fog nodes
$p^{PU,a}$	53 W
$p^{PU,s}$	15 W
$p^{ci,a}$	2.22 W
$p^{ci,s}$	1.28 W
t_{oh}	2 ms
$t^{ci,l}$	1 ms
I_{re}	600 s

Here, we observe how the number of deployed fog nodes (WFNs, SFNs and IFNs) changes as the requests for task processing in an FCI change. In our performance evaluation, we assume the task processing request arrival rate depicted in Figure 4. This figure depicts the trend of request arrivals (actual load) in a day; the number of requests increases during daytime (between the 7th and 15th hours) and gradually decreases towards night time. This figure also portrays the ARIMA-based one-day and one-hour prediction of the task requests.

The main challenge in any prediction area is achieving good accuracy. For time-series analysis, prediction window size affects the prediction accuracy. Figure 5 illustrates the prediction errors for different prediction window sizes. The error percentage is calculated by subtracting the actual load from the predicted load. Positive values indicate an underestimate, i.e., the actual load is higher than the predicted load, whereas negative values indicate an over-estimate i.e., the prediction load is higher than the actual load. We can note from this figure that as the prediction window increases, the prediction error also rises.

Figure 6a–c illustrate the number of WFNs, SFNs and IFNs, respectively, in different parts of a day in the proposed GDAFC (evaluated based on the request arrivals depicted in Figure 4). It can be seen that the number of working nodes deployed and the number of idle nodes are highly influenced by the request arrival rate. When the demand decreases, the number of working nodes decreases as well. This is because fewer fog resources are required to serve the requests during that time. That is, some of the working nodes are no longer required; therefore, they are invoked to become IFNs. For example, during the 7th hour (see Figure 4), the fog demand is the lowest. We can see that the number of working

nodes during that same hour is also the lowest, whereas the number of idle nodes is the highest (see Figure 6c). As the demand increases, more idle nodes are being deployed as working nodes to serve the increasing requests within the FCI operator-defined delay requirement (DR).

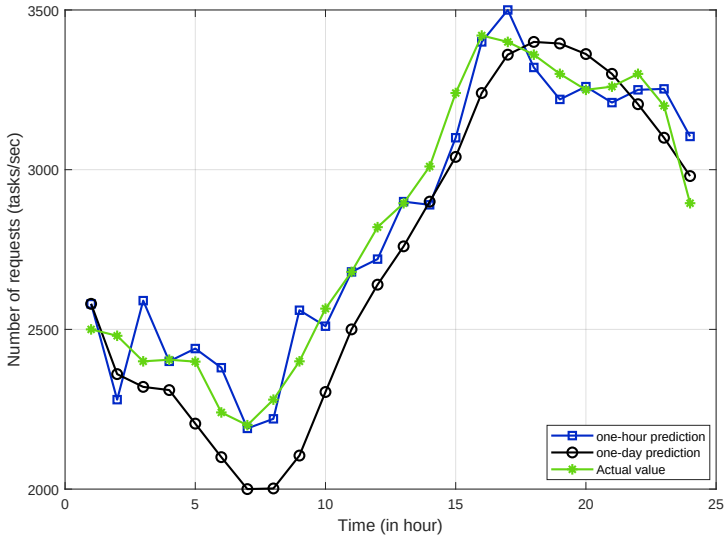


Figure 4. Total task processing requests based on actual, one-hour and one-day prediction.

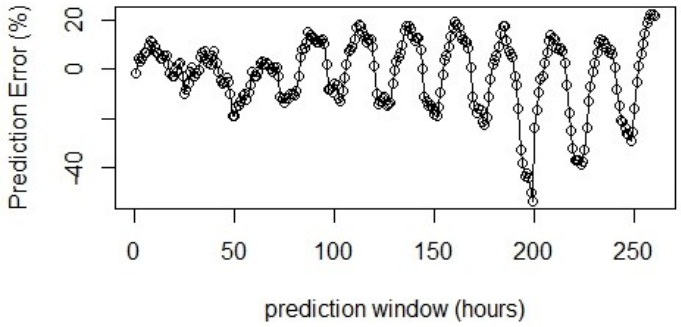
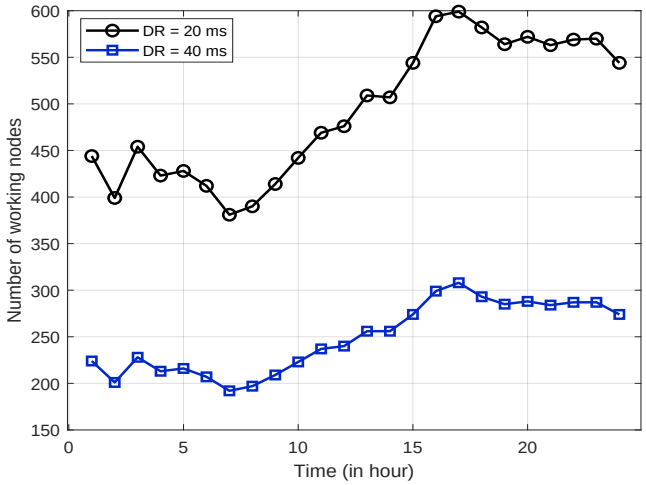


Figure 5. Prediction error by varying prediction window size from 1 h to 260 h.



(**a**)

Figure 6. *Cont.*

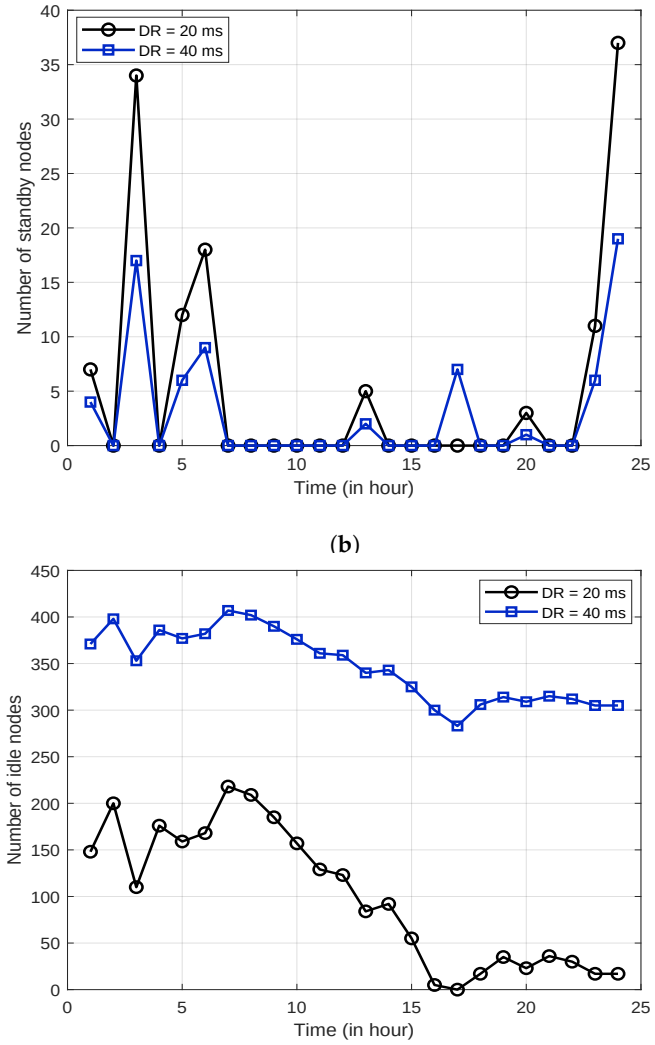


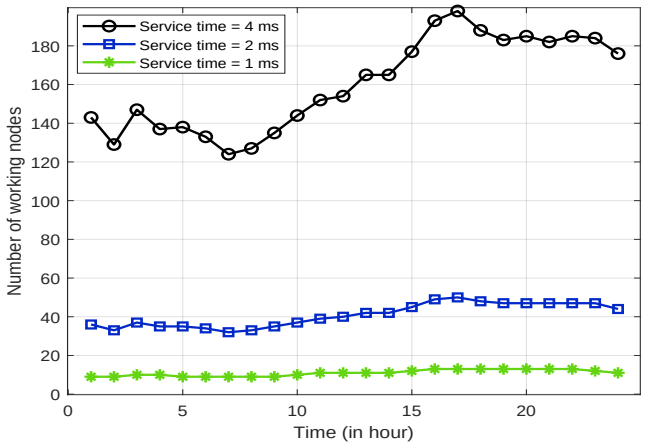
Figure 6. The influence of the task completion *DR* on the number of selected WFNs, SFNs and IFNs in GDAFC. (**a**) The number of WFNs; (**b**) the number of SFNs; (**c**) the number of IFNs.

It can be observed from Figure 4 that there are times when the prediction model over-estimated and under-estimated the actual load. We can expect to see the presence of standby nodes during those hours where the prediction model under-estimates the actual load (see Figure 4). Referring to Figure 6b, we can see that the number of deployed SFNs in the proposed solution is relatively higher during those hours where the request arrival is higher than predicted (e.g., see the predicted and actual load results for the 2nd hour in Figure 4). At the 3rd hour in Figure 4, more requests were expected to arrive than the actual load; hence, no standby nodes were deployed during that time. This shows that despite having a prediction error, our GDAFC is able to deploy sufficient fog nodes to ensure that all incoming requests are well served below *DR*. We also investigated how the *DR* value influences the number of WFNs, SFNs and IFNs in our GDAFC. In Figure 6a,b we can note that when the *DR* value is relatively relaxed, the number of deployed WFNs and SFNs is lower. In Figure 6c, we can also note unsurprisingly that in case of relaxed *DR* (e.g., 40 ms), the number of IFNs is far higher than that of the 20 ms case.

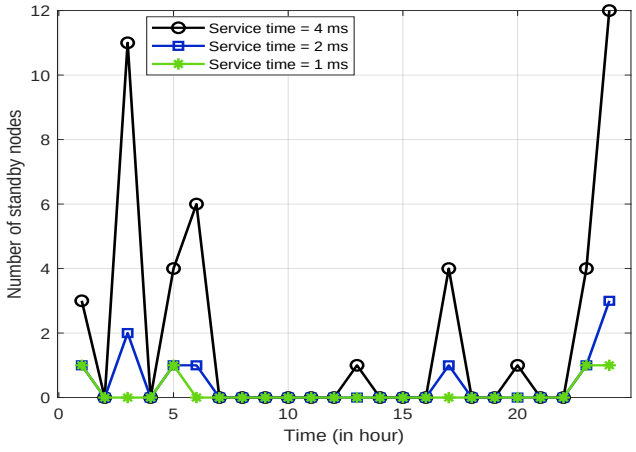
We observe here how the proposed solution is influenced by service time of a fog node. Figure 7a–c illustrate the number of WFNs, SFNs and IFNs, respectively, in different parts of a day in the proposed GDAFC under service times (μ) of 4 ms, 2 ms and 1 ms. At low service times, such as 1 ms, fewer working nodes are deployed compared to those in high

service times, such as 2 ms. We can observe from Figure 7a that as the service time increases the number of WFNs increases. Thus, the number of IFNs declines with the increment of service time (see Figure 7c).

In the access network segment, the allocated bandwidth of the communication channel could be limited [39], resulting in the imposition of a high forwarding delay for sending a task-processing request and obtaining a reply from a fog node. In this section, we also study how the task-forwarding delay to a user ($D_{forward}$) influences the number of WFNs, SFNs and IFNs in our GDAFC. Additionally, we aim at investigating this under different task completion delay requirements in order to understand further when $D_{forward}$ significantly influences the decision to determine the number of WFNs, SFNs and IFNs in GDAFC. Figure 8a,b demonstrate that as the $D_{forward}$ increases, the number of allocated WFNs and SFNs increases, in cases when delay requirements are stringent (e.g., 20 ms). The reason for this is that our proposed GDAFC in this scenario tries to offset the impact of a long forwarding delay by reducing computation delays (i.e., increasing the number of deployed WFNs and SFNs) so as to keep the overall delay within the delay requirement boundary. It is worth noting that in such case, the utilization of the WFNs and SFNs may be reduced in GDAFC. In these figures, we note that as the delay requirement becomes relaxed, the number of required WFNs and SFNs declines. Furthermore, looking at Figure 8a–c, we can infer that as the $D_{forward}$ value becomes relatively low compared to the delay requirement value (e.g., 100 ms), the $D_{forward}$ has a negligible impact on determining the number of WFNs, SFNs and IFNs for a given task arrival rate.

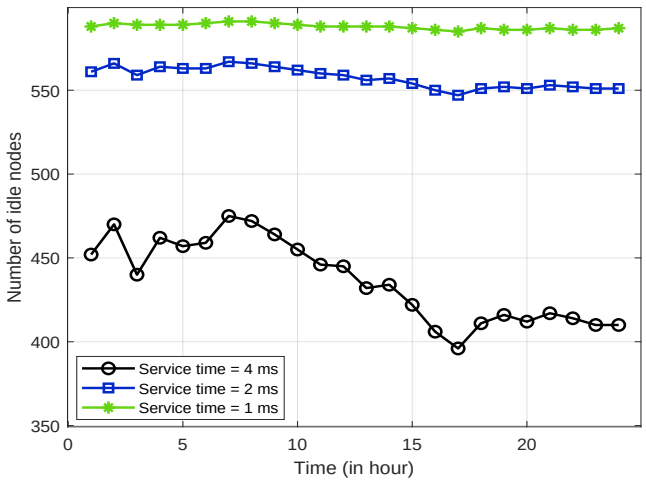


(**a**)



(**b**)

Figure 7. *Cont.*

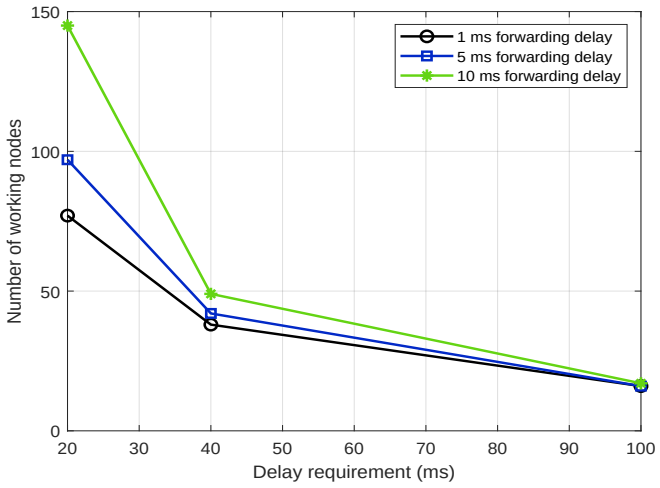


(**c**)

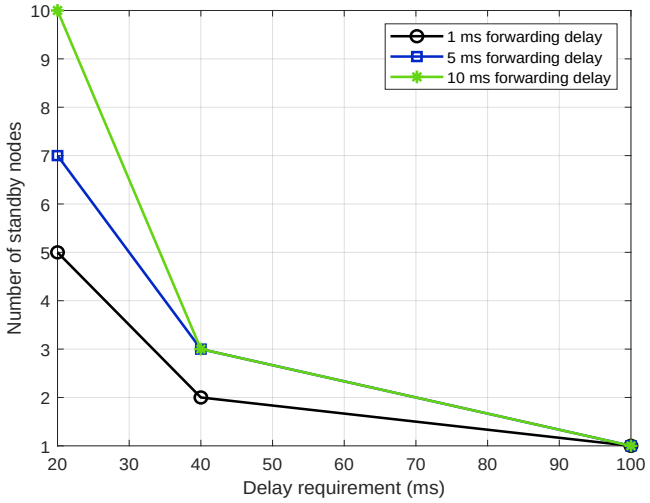
Figure 7. The influence of service time of a fog node on the number of selected WFNs, SFNs and IFNs in GDAFC. (**a**) The number of WFNs; (**b**) the number of SFNs; (**c**) the number of IFNs.

We compared the energy consumption performance of our proposed solution with a solution which always keeps the fog nodes active, regardless of request arrivals. Figure 9 shows the energy consumption of an FCI under 20 ms and 40 ms delay requirements. As we can observe from this figure, the amount of energy consumption in GDAFC rises strongly, influenced by the amount of task arrivals. The lowest energy is consumed between the 7th and 8th hours under both delay requirements. We note in this figure that the stringent DR value tends to impose more energy consumption compared to the case when the DR value is relatively relaxed (i.e., 40 ms in this performance evaluation). There are two reasons why the energy consumption in DR = 20 ms is more than DR = 40 ms one. First, in a stringent delay requirement case, the broker needs to deploy more WFNs and SFNs than when the DR = 20 ms. Second, in a relaxed delay requirement scenario, the sleep interval length is longer for the fog nodes. This in turn reduces the chance of idle listening of the communication interface of a fog node, thereby allowing the fog node to conserve more energy compared to the case when the sleep interval is short when DR value is stringent (e.g., 20 ms).

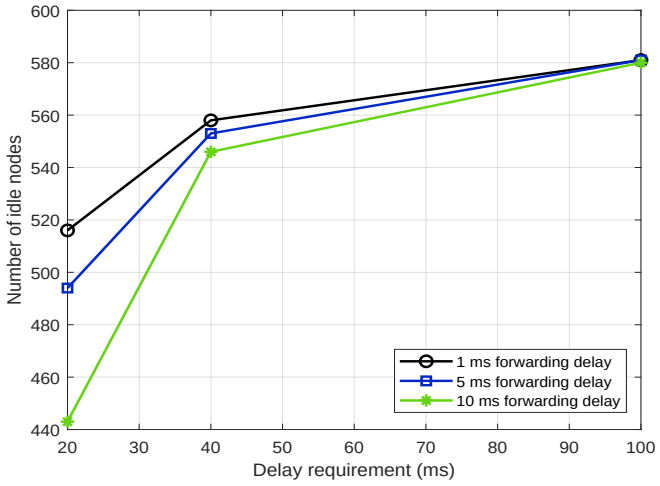
The service time (μ) of a fog node and the forwarding delay ($D_{forward}$) also has impact on the overall energy consumption of an FCI in GDAFC. Figure 10 shows that as the service time increases, the energy consumption in an FCI increases. This is because with the increment of the service time in GDAFC the number of deployed WFNs and SFNs increases (see Figure 7a,b). Furthermore, we note in Figure 11 that as $D_{forward}$ increases, the overall energy consumption increases in an FCI. The cause behind this is that when $D_{forward}$ is high, GDAFC tends to deploy more WFNs and SFNs compared to the case when $D_{forward}$ is lower (see Figure 8a,b). It can be noted in this figure that the amount of energy consumption noticeably increases as the $D_{forward}$ increases when the delay requirements are stringent.



(**a**)



(**b**)



(**c**)

Figure 8. The influence of $D_{forward}$ on the number of selected WFNs, SFNs and IFNs under different delay requirements in GDAFC. (**a**) The number of WFNs under different imposed delay requirements; (**b**) the number of SFNs under different imposed delay requirements; (**c**) the number of IFNs under different imposed delay requirements.

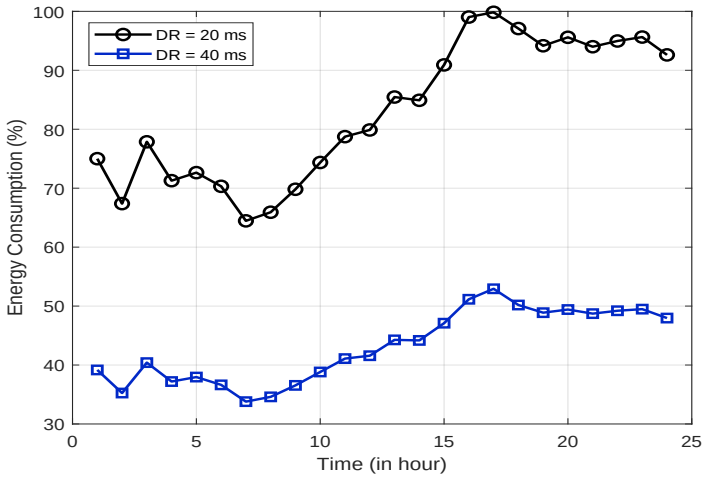


Figure 9. Energy consumption of a fog computing infrastructure under 20 ms and 40 ms delay requirements in the proposed GDAFC.

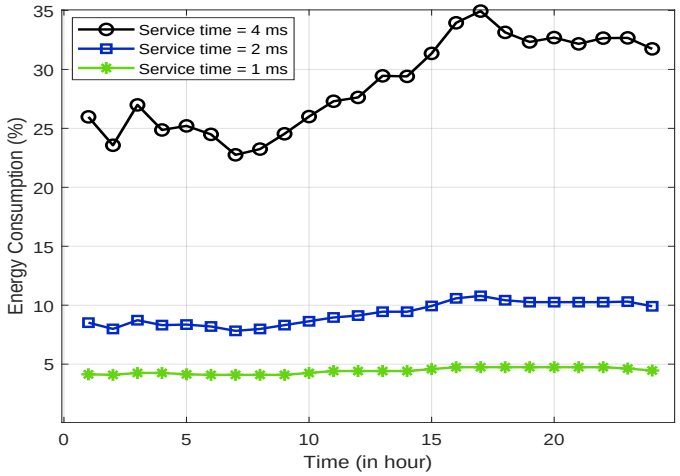


Figure 10. Energy consumption of a fog computing infrastructure under different service times of a fog node in the proposed GDAFC.

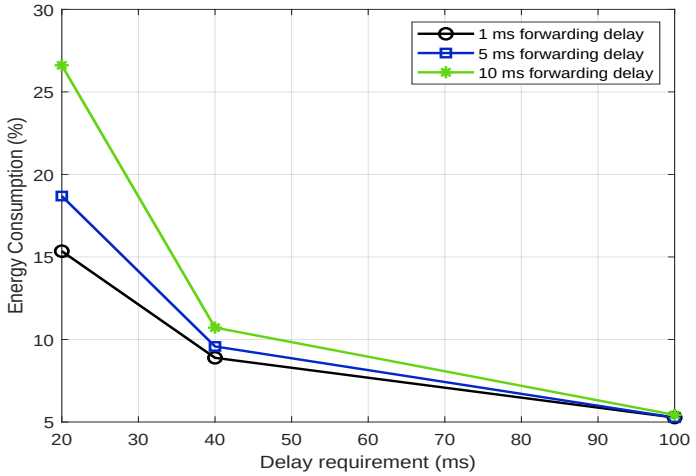


Figure 11. Energy consumption of a fog computing infrastructure under different forwarding delays in the proposed GDAFC.

5. Discussion and Open Research Issues

It is apparent that fog computing will pave the way for realizing the latency requirements of different latency stringent applications, including augmented reality, virtual reality and smart traffic control. To keep pace with the ever-increasing demand for such latency-stringent applications, we will witness an unprecedented amount of demand for fog computing in the coming years. This will undoubtedly lead to an increase in energy consumption in the access and network edge segments, where fog nodes are deployed. The proposed GDAFC solution aims at reducing the energy consumption by the fog-computing infrastructure, while not compromising with the latency requirements of the application. This solution relies solely on the prediction of task load arrivals. Poor prediction accuracy would lead to to under-allocations or over-allocations of computational resources. This may, in turn, result in unnecessarily increases in energy consumption or failure to meet QoS requirements. In the GDAFC approach, the sleep interval length of the communication interface of a fog node (both WFN and IFN) and the number of total deployed WFNs and SFNs are determined by taking into consideration the delay requirements of an application. Therefore, this solution should be able to meet the application's latency requirements, while reducing energy consumption as much as possible. However, we believe that the prediction accuracy can also play a significantly important role in terms of the task-processing delay performance and energy-saving performance of a fog-computing infrastructure in our solution.

In particular, in cases when the computational resources are under-allocated, the incoming task may experience task delays, which may exceed the delay requirements of the tasks. Note that once the IFNs are identified and they are invoked to move into a sleep state, the broker can communicate with them only after I_{re}. Therefore, in cases of a sudden rise in computational demand, the broker needs to rely solely on WFNs and SFNs. The questions that could be asked include the following: What would be the possible approach a broker can adopt under such a situation in order to meet the delay requirements? What are the possible ways to improve the prediction accuracy so as to avoid under- or over-allocation?

In cases when the numbers of deployed WFNs and SFNs are not sufficient with a given service rate, the broker may invoke those fog nodes (i) to reduce the sleep interval of the communication interfaces (this will allow the broker to communicate with the fog node mode more frequently than before) and (ii) adjust the service rate (processing performance) of the fog node to meet the deadline of task completion. The former approach is a commonly applied approach for saving energy in any processor (CPU frequency is adjusted depending on performance requirements [40]).

The prediction accuracy improvement should be another important area that future research should investigate. We believe that aside from improving different supervised learning methods, future research should investigate how different context information regarding customers' premises can be integrated in order to understand the task-processing arrival demands or the types of applications that are popular at a given time. Understanding the context from various sources including end-users, fog nodes, network infrastructure and clients, and then transforming them into cognitive levels, can contribute a great deal towards filfilling the SLAs of different services for clients, while minimizing the carbon footprint of fog-computing infrastructures.

Furthermore, in cases when a WFN or SFN fails while processing task(s), there is a need for a mechanism to resume the incomplete task in other WFNs or SFNs seamlessly. Therefore, future research should be conducted to investigate on how the progress of task-processing in the fog nodes can be tracked, as well as how incomplete tasks can be efficiently reassigned to other fog nodes, while not violating the imposed delay requirement for the task completion. Note that one possible approach to track the progress of the task processing in the fog node would be the periodic polling of each fog node. A short inter-polling interval will allow faster failure identification. However, this would lead to an increase in the amount of network traffic and would force the fog node interface to leave its sleep state, resulting in an increase in energy consumption in the fog node. A longer

inter-polling interval will result in delaying the failure identification; however, this will not force the fog nodes to leave their sleep state frequently. Therefore, future research should study how an optimal inter-polling interval can be set, taking into consideration different constraints, including the assigned task-completion delay requirement.

6. Conclusions

In this paper, we propose an energy-efficient and demand-aware fog computing solution. This paper has demonstrated that it is unnecessary to keep all the fog nodes in an FCI active in order to provide a requested service. Therefore, we have proposed to dynamically decide the number of working nodes in an FCI based on the predicted workload at a given time. In order to avoid task dropping or delay-requirement violations when the prediction accuracy is low, we have proposed to designate some of the fog nodes in an FCI as standby nodes. In particular, our dynamic allocation approach for fog nodes divides the fog nodes into three separate groups: working nodes, standby nodes and idle nodes. Based on our numerical analysis, we have successfully demonstrated that the proposed solution reduces the energy consumption of a fog-computing infrastructure successfully, while meeting the desired delay requirements. Our future avenues of research in this area include developing a solution for efficiently tracking task-processing progress in the fog nodes and reassigning incomplete tasks to other fog nodes in the event of a fog node failure.

Author Contributions: Conceptualization, D.S.N.K.P.A.K. and S.H.S.N.; methodology, D.S.N.K.P.A.K. and S.H.S.N.; validation, D.S.N.K.P.A.K. and S.H.S.N.; formal analysis, D.S.N.K.P.A.K. and S.H.S.N.; investigation, D.S.N.K.P.A.K.; data curation, D.S.N.K.P.A.K.; writing—original draft preparation, D.S.N.K.P.A.K., S.H.S.N., F.H.R., G.K.; writing—review and editing, S.H.S.N., F.H.R., G.K., G.M.L. and T.-W.A.; Visualization, D.S.N.K.P.A.K. and S.H.S.N., supervision, S.H.S.N. and T.-W.A.; funding acquisition, G.M.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Universiti Teknologi Brunei (UTB), Brunei Darussalam.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

WFN	Working Fog Node
SFN	Standby Fog Node
IFN	Idle Fog Node
QoS	Quality-of-Service
RECON	Reconfiguration period
SLA	Service Level Agreement
UE	User Equipment
FCI	Fog Computing Infrastructure
CI	Communication Interface
PU	Processing Unit
ARIMA	Autoregressive Integrated Moving Average

References

1. Beloglazov, A.; Buyya, R.; Lee, Y.C.; Zomaya, A. A taxonomy and survey of energy-efficient data centers and cloud computing systems. In *Advances in Computers*; Elsevier: Amsterdam, The Netherlands, 2011; Volume 82, pp. 47–111.
2. Boru, D.; Kliazovich, D.; Granelli, F.; Bouvry, P.; Zomaya, A.Y. Energy-efficient data replication in cloud computing datacenters. *Clust. Comput.* **2015**, *18*, 385–402. [CrossRef]
3. Consumerlab, E. hot consumer trends 2016. *Retrieved Sept.* **2016**, *1*, 10.

4. Weiner, M.; Jorgovanovic, M.; Sahai, A.; Nikolié, B. Design of a low-latency, high-reliability wireless communication system for control applications. In Proceedings of the 2014 IEEE International Conference on Communications (ICC), Sydney, NSW, Australia, 10–14 June 2014; pp. 3829–3835.
5. Byers, C.C. Architectural imperatives for fog computing: Use cases, requirements, and architectural techniques for FOG-enabled IoT networks. *IEEE Commun. Mag.* **2017**, *55*, 14–20. [CrossRef]
6. Leverich, J.; Kozyrakis, C. On the energy (in) efficiency of hadoop clusters. *ACM SIGOPS Oper. Syst. Rev.* **2010**, *44*, 61–65. [CrossRef]
7. Oma, R.; Nakamura, S.; Duolikun, D.; Enokido, T.; Takizawa, M. An energy-efficient model for fog computing in the internet of things (IoT). *Internet Things* **2018**, *1*, 14–26. [CrossRef]
8. Hou, S.; Ni, W.; Zhao, S.; Cheng, B.; Chen, S.; Chen, J. Frequency-reconfigurable cloud versus fog computing: An energy-efficiency aspect. *IEEE Trans. Green Commun. Netw.* **2019**, *4*, 221–235. [CrossRef]
9. Rahman, F.H.; Newaz, S.S.; Au, T.W.; Suhaili, W.S.; Lee, G.M. Off-street vehicular fog for catering applications in 5G/B5G: A trust-based task mapping solution and open research issues. *IEEE Access* **2020**, *8*, 117218–117235. [CrossRef]
10. Amudha, S.; Murali, M. DESD-CAT inspired algorithm for establishing trusted connection in energy efficient FoG-BAN networks. *Mater. Today Proc.* **2021**, *in press*. [CrossRef]
11. Fang, W.; Zhang, W.; Chen, W.; Liu, Y.; Tang, C. TME 2 R: Trust Management-Based Energy Efficient Routing Scheme in Fog-Assisted Industrial Wireless Sensor Network. In *International Conference on 5G for Future Wireless Networks*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 155–173.
12. Wang, T.; Qiu, L.; Sangaiah, A.K.; Xu, G.; Liu, A. Energy-efficient and trustworthy data collection protocol based on mobile fog computing in Internet of Things. *IEEE Trans. Ind. Inform.* **2019**, *16*, 3531–3539. [CrossRef]
13. Malik, U.M.; Javed, M.A.; Zeadally, S.; ul Islam, S. Energy efficient fog computing for 6G enabled massive IoT: Recent trends and future opportunities. *IEEE Internet Things J.* **2021** . [CrossRef]
14. Wan, J.; Chen, B.; Wang, S.; Xia, M.; Li, D.; Liu, C. Fog computing for energy-aware load balancing and scheduling in smart factory. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4548–4556. [CrossRef]
15. Shahid, M.H.; Hameed, A.R.; ul Islam, S.; Khattak, H.A.; Din, I.U.; Rodrigues, J.J. Energy and delay efficient fog computing using caching mechanism. *Comput. Commun.* **2020**, *154*, 534–541. [CrossRef]
16. Jiang, Y.L.; Chen, Y.S.; Yang, S.W.; Wu, C.H. Energy-efficient task offloading for time-sensitive applications in fog computing. *IEEE Syst. J.* **2018**, *13*, 2930–2941. [CrossRef]
17. Xiao, Y.; Krunz, M. Distributed optimization for energy-efficient fog computing in the tactile Internet. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 2390–2400. [CrossRef]
18. Rahman, F.H.; Newaz, S.S.; Au, T.W.; Suhaili, W.S.; Mahmud, M.P.; Lee, G.M. EnTruVe: ENergy and TRUst-aware Virtual Machine allocation in VEhicle fog computing for catering applications in 5G. *Future Gener. Comput. Syst.* **2022**, *126*, 196–210. [CrossRef]
19. Reddy, K.H.K.; Luhach, A.K.; Pradhan, B.; Dash, J.K.; Roy, D.S. A genetic algorithm for energy efficient fog layer resource management in context-aware smart cities. *Sustain. Cities Soc.* **2020**, *63*, 102428. [CrossRef]
20. Meisner, D.; Gold, B.T.; Wenisch, T.F. PowerNap: Eliminating server idle power. In *ACM Sigplan Notices*; ACM: New York, NY, USA, 2009; Volume 44, pp. 205–216.
21. Mathew, V.; Sitaraman, R.K.; Shenoy, P. Energy-aware load balancing in content delivery networks. In Proceedings of the INFOCOM, 2012 Proceedings IEEE, Orlando, FL, USA, 25–30 March 2012; pp. 954–962.
22. Kamitsos, I.; Andrew, L.; Kim, H.; Chiang, M. Optimal sleep patterns for serving delay-tolerant jobs. In Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking, Passau, Germany, 13–15 April 2010; ACM: New York, NY, USA, 2010; pp. 31–40.
23. Sarji, I.; Ghali, C.; Chehab, A.; Kayssi, A. Cloudese: Energy efficiency model for cloud computing environments. In Proceedings of the 2011 International Conference on Energy Aware Computing, Istanbul, Turkey, 30 November–2 December 2011; pp. 1–6.
24. Duan, L.; Zhan, D.; Hohnerlein, J. Optimizing cloud data center energy efficiency via dynamic prediction of cpu idle intervals. In Proceedings of the Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on Cloud Computing IEEE, New York, NY, USA, 27 June–2 July 2015; pp. 985–988.
25. Dias, M.P.I.; Karunaratne, B.S.; Wong, E. Bayesian estimation and prediction-based dynamic bandwidth allocation algorithm for sleep/doze-mode passive optical networks. *J. Light. Technol.* **2014**, *32*, 2560–2568. [CrossRef]
26. Farahnakian, F.; Pahikkala, T.; Liljeberg, P.; Plosila, J.; Tenhunen, H. Utilization prediction aware VM consolidation approach for green cloud computing. In Proceedings of the Cloud Computing (CLOUD), 2015 IEEE 8th International Conference on Cloud Computing, New York, NY, USA, 27 June–2 July 2015; pp. 381–388.
27. Zhang, S.; Zhao, S.; Yuan, M.; Zeng, J.; Yao, J.; Lyu, M.R.; King, I. Traffic Prediction Based Power Saving in Cellular Networks: A Machine Learning Method. In Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Redondo Beach, CA, USA, 7–10 November 2017; ACM: New York, NY, USA, 2017; p. 29.
28. Saker, L.; Elayoubi, S.E.; Chahed, T. Minimizing energy consumption via sleep mode in green base station. In Proceedings of the Wireless Communications and Networking Conference (WCNC), Sydney, NSW, Australia, 18–21 April 2010; pp. 1–6.
29. Khan, A.M.; Freitag, F. On Participatory Service Provision at the Network Edge with Community Home Gateways. *Procedia Comput. Sci.* **2017**, *109*, 311–318. [CrossRef]

30. Chou, C.H.; Wong, D.; Bhuyan, L.N. Dynsleep: Fine-grained power management for a latency-critical data center application. In Proceedings of the 2016 International Symposium on Low Power Electronics and Design, San Francisco, CA, USA, 8–10 August 2016; pp. 212–217.
31. Rahman, F.H.; Au, T.W.; Newaz, S.S.; Suhaili, W.S.; Lee, G.M. Find my trustworthy fogs: A fuzzy-based trust evaluation framework. *Future Gener. Comput. Syst.* **2020**, *109*, 562–572. [CrossRef]
32. Ge, C.; Sun, Z.; Wang, N. A survey of power-saving techniques on data centers and content delivery networks. *IEEE Commun. Surv. Tutor.* **2013**, *15*, 1334–1354.
33. Luan, T.H.; Gao, L.; Li, Z.; Xiang, Y.; Wei, G.; Sun, L. Fog computing: Focusing on mobile users at the edge. *arXiv* **2015**, arXiv:1502.01815.
34. Ross, T.J. *Fuzzy Logic with Engineering Applications*; John Wiley & Sons: Hoboken, NJ, USA, 2009.
35. Izquierdo, L.R.; Olaru, D.; Izquierdo, S.S.; Purchase, S.; Soutar, G.N. Fuzzy logic for social simulation using NetLogo. *J. Artif. Soc. Soc. Simul.* **2015**, *18*, 1. [CrossRef]
36. Li, S.; Da Xu, L.; Wang, X. Compressed sensing signal and data acquisition in wireless sensor networks and internet of things. *IEEE Trans. Ind. Inform.* **2013**, *9*, 2177–2186. [CrossRef]
37. Kayacan, E.; Ulutas, B.; Kaynak, O. Grey system theory-based models in time series prediction. *Expert Syst. Appl.* **2010**, *37*, 1784–1789. [CrossRef]
38. Dabbagh, M.; Hamdaoui, B.; Guizani, M.; Rayes, A. Toward energy-efficient cloud computing: Prediction, consolidation, and overcommitment. *IEEE Netw.* **2015**, *29*, 56–61. [CrossRef]
39. Yan, S.; Gu, Z.; Nguang, S.K. Memory-event-triggered H_∞ output control of neural networks with mixed delays. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**. [CrossRef]
40. Kliazovich, D.; Arzo, S.T.; Granelli, F.; Bouvry, P.; Khan, S.U. Accounting for load variation in energy-efficient data centers. In Proceedings of the Communications (ICC), 2013 IEEE International Conference on Communications (ICC), Budapest, Hungary, 9–13 June 2013; pp. 2561–2566.