Load-Balancing Algorithm for Multiple Gateways in Fog-Based Internet of Things

Fatemeh Banaie, *Student Member, IEEE*, Mohammad Hossein Yaghmaee, *Senior Member, IEEE*, Seyed Amin Hosseini, and Farzad Tashtarian

***Abstract*—This article investigates the performance of multicriteria-based load-balancing scheme among gateways in fog-assisted Internet of Things (IoT). We employ a queueing model of the IoT system to calculate the latency of data streams from the IoT devices to the applications. However, a gateway node may easily become congested since all the traffic to IP networks are directed toward the gateway. A congested point can affect system performance and may cause reliability problems in the system. Thus, we employ multiple gateways to alleviate the performance degradation of the network, along with a multicriteria decision-making (MCDM)-based load-balancing policy among the gateways to achieve a global load fairness. The proposed model is evaluated in single-hop IPv6 over low-power wireless personal area networks (6LoWPANs). The evaluation results show the effectiveness of the proposed load-balancing model in providing fast and reliable responses to user queries.**

***Index Terms*—IPv6 over low-power wireless personal area network (6LoWPAN), fog-based Internet of Things (IoT), hybrid proxy, load balancing, multicriteria decision making (MCDM).**

I. Introduction

THE Internet of Things (IoT) has been developed as an intriguing paradigm that describes the connections of smart objects and devices over IP networks with the support of modern communication techniques [1], [2]. The IoT enables things to observe and sense the environment states, to make coordinated decisions in various sensing service applications, such as smart transportation, smart healthcare, home automation, and smart city. In order to realize the full benefits of the IoT, it is necessary to provide the new innovative networking technologies and communication schemes, which allows IPv6 communications over low-power and lossy networks (LLNs) such as wireless sensor networks (WSNs) [3]–[5].

To this end, the Internet Engineering Task Force (IETF) has developed some standardized communications protocols, such as the IPv6 over low-power wireless personal area networks (6LoWPANs) [6], the routing protocol for low power and lossy network (RPL) [7], and the constrained application protocol (CoAP) [8] to tackle the technical challenges of the existing protocol's overhead against the limitations of the sensing devices in the terms of constrained energy, memory, and computational resources [9]. However, providing sufficient computing resources to process and store the huge amount of generated data by the distributed IoT devices is necessary to alleviate the traffic load in the network and minimize the latency of IoT devices [11].

Owing to the large volumes of high-velocity streams of data, a specialized platform is employed to process and store the big data generated from the distributed IoT devices. The platform can provide flexible and efficient computing resources by connecting to a remote cloud backend [12], [13]. Such an architecture can offer localized services by leveraging the processing and storage resources of the cloud on the edge of the network, known as *fog computing*. Specifically, in-network computation is provided through edge devices (e.g., gateways located on the fog nodes) by initially processing the collected data from the distributed sensors and transmitting the refined data toward the requested host. This is necessary to facilitate the service management and real-time processing of data from a wide variety of IoT devices [14]. However, transmitting the generated data from the IoT devices to the gateways may consume high bandwidth and energy of the IoT devices. Besides, it increases the latency of data processing in the network, which makes it intolerable for many delay-sensitive applications [15].

To accommodate performance issues, content caching in the application layer is proposed to speed up the delivery rate of the processing data and solve the energy efficiency problem. Therefore, the sensed data will be sent to the gateway by the IoT devices and clients can access the sensed data from the gateway rather than the IoT device. However, all of this generated traffic from low-power wireless personal area networks (LoWPANs) should pass through the gateway, which may easily become congested bottleneck and cause reliability problems [16]. Multiple gateways can be employed to relieve this issue and provide efficient connectivity within the IoT network.

Two adjacent gateways may be located close to each other, thus, having overlapped coverage areas. Consequently, each IoT device may be associated with one or more gateway(s) of the edge nodes. In this case, the traffic loads among the gateways may be unbalanced due to the absence of the proper load-balancing scheme. The overloaded gateways will become the bottleneck of the system, thus causing longer communications latency of data streams in the network. Over several research works in literature, applications workload is assigned

Manuscript received September 26, 2019; revised January 3, 2020 and February 12, 2020; accepted March 9, 2020. *(Corresponding author: Seyed Amin Hosseini.)*

Fatemeh Banaie, Mohammad Hossein Yaghmaee, and Seyed Amin Hosseini are with the Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad 9177948974, Iran

Farzad Tashtarian is with the Mashhad Branch, Islamic Azad University, Mashhad, Iran.

Digital Object Identifier 10.1109/JIOT.2020.2982305

to each gateway according to the required computing resource in a hierarchical cloudlet network in [17]. Specifically, an application-aware load-balancing scheme is applied in [18] to select the destination gateways for different types of user requests and the amount of computing resources allocated for each application in each cloudlet. However, assigning all workloads of the application to a specific gateway would incur the recaching problem among gateways in a dynamic IoT environment. To solve the problem, a multicriteria decision-making (MCDM)-based load-balancing technique can be employed to alleviate the network latency by efficiently associating the task requests among the overlapping gateways.

In this article, we present an analytical model to evaluate the performance of traffic load balancing in a multiple gateway-enabled IoT system. The proposed load-balancing model in this article is twofold. First, a queuing network problem is solved to evaluate the latency of task executing in the network. After modeling, an MCDM-based load-balancing scheme is used to balance the traffic loads among the gateways to achieve a global load fairness and reduced latency of service processing in the IoT system. The main contributions and key results of this article are summarized as follows.

1) We present a model-driven and analytical approach to evaluate the performance of a load-balanced multigateway-enabled IoT system. To the best of our knowledge, this is the first study that considers the $M/G/1$ system with a multiple vacation model for evaluating the network latency in load balancing the traffic loads among gateways. In this model, the busy period of the controller for processing the task requests of other gateways induce vacations for each of the other gateways. The works of literature consider a simple $M/M/1$ model for evaluating the network delay [16]–[19]. However, despite the Poisson process provide a good approximation of the arrival rate in real systems [20], exponential distribution may not provide a general estimation of the real-world scenarios. Considering more general queuing models can increase the effectiveness of the evaluation model for real-world applications.
2) We formulate the problem of workload allocation into the gateways as a multiobjective optimization problem in the form of a mixed-integer linear programming (MILP). The proposed MILP model tries to balance the network load by minimizing the maximum loads of the gateways. Since this problem is NP-complete, we apply an MCMD method for gateway selection by taking into account their load changes over time in order to deal with the high time complexity issue of the MILP model, particularly in large-scale scenarios. The analytical hierarchy process (AHP) is one of the most popular MCDM method that is being used in complex decision-making processes due to its simplicity and consistency [21]–[23]. It is an effective method in the decision-making problem (e.g., load management problem) that selects the most suitable gateway by evaluating the candidate gateways.
3) We deploy the IPv6 protocol stack over sensor devices and integrate the IoT devices with multiple gateways load-balancing method to make the evaluation closer to the real-world models.

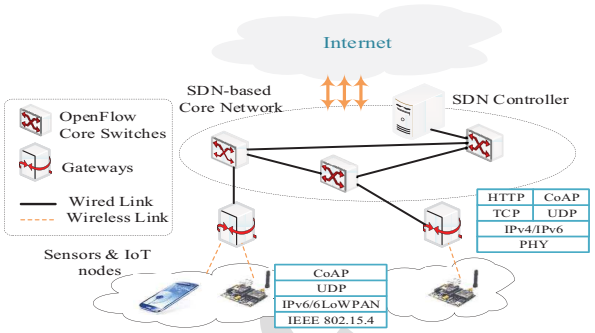


Fig. 1. Example scenario of IoT sensing system.

The remainder of this article is organized as follows. In Section II, we briefly describe the system model used in this article. In Section III, the proposed analytical load-balancing scheme is described in detail. In Section IV, we demonstrate and discuss the evaluation results, followed by the conclusion in Section V.

II. System Model

This article considers a multigateway-based 6LoWPAN architecture that supports the MCDM-based load-balancing method to balance the traffic loads among the gateways. In this architecture, which is schematically shown in Fig. 1, all sensor nodes access the Internet through the gateways. The gateway deployment leverages the edge IoT architecture in which gateways are border entities in the networks and handle ingress and egress data traffic. In addition, a software-defined networking (SDN)-based core is used to connect the gateways to the Internet. Gateway nodes are connected to the OpenFlow-enabled switches that separate out the control functions from data forwarding functions. The SDN controller manages the gateway nodes via the OpenFlow-based switches that monitor the traffic at the data plane [24]. It also has a load-balancing function to associate the user load to the gateways in a fairly manner.

The sensor nodes keep measuring the environmental parameters, e.g., noise-level measurement, and send the measured data to the gateways using CoAP POST methods in a confirmable (CON) mode. Since the CoAP protocol utilizes UDP in the transport layer, it supports optional reliability by defining two modes of messages: 1) CON and 2) nonconfirmable (NON). To ensure reliable communication, the messages are retransmitted with exponential timeouts until the receiver acknowledges the reception of data or it reaches to the maximum number of retransmissions.

A Web application accesses the measured data via a gateway/proxy with a cache deployed on it to allow some of the user queries to be served from the cache. However, the IoT data are transient [25], i.e., it has a lifetime and expires after this time period. Therefore, a caching policy is required to keep the cached data up-to-date in the proxy. We employed a CoAP proxy to leverage the functionalities offered by the proxy in decoupling the applications from physical sensors and

devices. A CoAP proxy usually operates in different operational modes of *forward proxy* and *reverse proxy*. In the former case, proxy obtains the sensed data by sending CoAP GET methods, while in the latter case, a periodical transmission of the sensed data is used by the IoT nodes. *Hybrid proxy* can leverage the benefits of both forward and reverse proxies [8]. In this model, a reverse proxy with occasional forward queries serves as a gateway between applications and sensor nodes.

Upon receiving a request, the cache decides whether to retrieve the fresh sensing data from the sensors using the CoAP GET method (in piggybacked CON mode) or to use the cached sensing data. Let $\mathcal{G} = \{1, \ldots, m\}$ denotes a set of gateways in the system and λ be the aggregated task request arrival rate to the system. Thus

$$\lambda = \sum_{g=1}^{m} \lambda_g, \quad g \in \mathcal{G} = \{1, \ldots, m\} \tag{1}$$

where λ_g is the arrival rate to each gateway $g \in \mathcal{G}$ and m is the number of gateways. If we assume that IoT nodes send the data in exponentially distributed time periods with mean $\overline{T_s} = 1/\lambda_s = 60$ s, the probability that the user query finds the outdated records (ORs) in different gateways is denoted by [29]

$$\begin{aligned} P_{\mathrm{OR},g} &= \int_{x=\overline{T_s}}^{\infty} P\{A_g < A_s | A_g = x\} dB(x) \\ &= \frac{\lambda_g}{\lambda_s + \lambda_g} e^{-(\lambda_g+\lambda_s)\overline{T_s}}, \quad g \in \{1..m\}. \end{aligned} \tag{2}$$

The above expression calculates the probability that interarrival times between the user requests is smaller than the cache update intervals (i.e., Prob$[A_g < A_s]$), consequently it shows the probability of data requests that should be forwarded to the IoT devices. A_g and A_s are the Laplace–Stieltjes transform (LST) of the interarrival time of the user requests to the gateway $g \in \mathcal{G}$ and the refresh messages of the sensor nodes, respectively. As Poisson interarrival times are exponential random variables, they are typically exponentially distributed and the LST of number of arriving messages can be defined as

$$A_j^*(s) = \int_0^{\infty} e^{-sx} \lambda_j e^{-\lambda_j x} dx = \frac{\lambda_j}{\lambda_j + s}, \quad j \in \{g, s\}. \tag{3}$$

Thus, the required refreshment messages are sent with the rate $\lambda_{get,g} = \lambda_g \times P_{\mathrm{OR},g}$.

III. Traffic Load Balancing Among Gateways

A. Problem Statement

There are often several overlapping gateways in the IoT domain running over IEEE802.15.4/6LoWPAN, which play a significant role in packet delivery as a relay point and end-devices. The generated traffic in 6LoWPAN is directed to the Internet via the gateways. To this end, a multigateway-based architecture can provide significant benefits in terms of network capacity and reliability. Normally, an SDN server would select the nearest local gateway with more computational resources for task association. However, this may lead to the imbalanced traffic load among gateways, so that some gateways suffer from heavy loads while the others are lightly loaded.

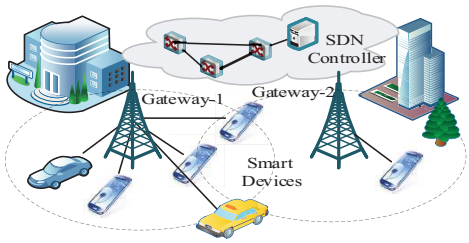


Fig. 2. Illustration of unbalanced traffic loads among gateways.

The heavily loaded gateways may confront the increased cache miss rate that will rise the need for retrieving the fresh data from the IoT devices. Apart from the energy exhaustion of the nodes that are close to these gateways, it may incur a higher delay beside the longer processing queues, which necessitates applying a workload-balancing scheme among the overlapping gateways.

Fig. 2 depicts a scenario of unbalanced traffic among gateways in a smart city application. Let us consider a smart city system in which the streets and buildings are equipped with smart devices and sensors. These smart devices sense the environmental parameters and send them to the nearest gateway that is available for them. Suppose in a special holiday, a large number of retrieval requests for finding the available parking spot, location of special places, traffic monitoring, weather conditions, etc., are sent toward these devices. Consequently, each device may need to transmit a large amount of data to these requests, which may result in increased delay of network and exhaust the energy supplies of the resources. In order to efficiently deliver the sensed data, the gateways cache their data periodically and respond to the requests on behalf of the devices. Normally, the gateways at the main streets or crowded places receive the most client retrieval requests that may lead to the improper load balancing and congestion on links nearby them. The unbalanced traffic loads may significantly increase the average delay of the network and disable the efficient data transmitting in the smart city system.

To this end, we leverage a load-balancing strategy of multiple gateways based on the SDN-based core network. In this method, the controller obtains an evenly balanced server load by considering the changing features of the network, such as resource loads and available resources. The SDN controller is a logical entity that implements the network policy, such as traffic scheduling and load management [26]. This solution can achieve a global load fairness and reduced latency of service processing in the IoT system. Furthermore, combining the load-balancing technology with the resource caching method provides reduced energy consumption for the IoT devices.

B. Problem Formulation

To formulate the problem, we have defined L as the total load of a set of overlapping gateways, $\mathcal{G} = \{g_1, g_2, \ldots, g_m\}$, where m is the number of gateways in each overlapping area. γ_t represents the load factor of each individual task, and the residual load capacity of each gateway is denoted by $L_{\mathrm{res},g}$,

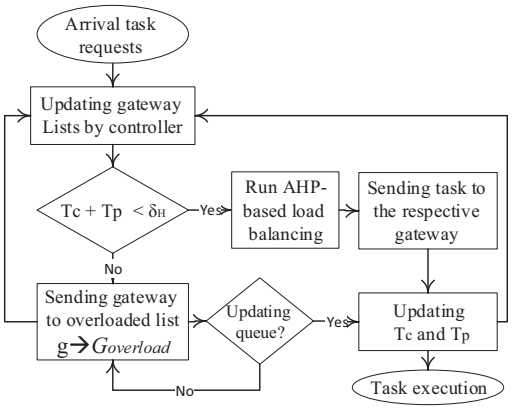


Fig. 3. MCDM-based load-balancing scheme.

respectively. The problem can be formulated as

$$\begin{aligned} \min\ & \alpha L + \beta \sum_{t} \sum_{g} B_{t,g} \\ \text{s.t. C1:}\ & \sum_{t \in tasks} B_{t,g}\gamma_t < L \quad \forall g \in \text{gateways} \\ \text{C2:}\ & \sum_{t \in \text{tasks}} B_{t,g}\gamma_t < L_{\text{res},g} \quad \forall g \in \text{gateways} \\ \text{C3:}\ & \sum_{g \in \text{gateways}} B_{t,g} \leq 1 \quad \forall t \in \text{tasks} \\ \text{Var.}\ & L_g \geq 0, \quad B_{t,g} \in \{0, 1\}. \end{aligned} \tag{4}$$

$B_{t,g}$ is a binary variable indicating whether the task t is assigned to the gateway g (i.e., $B_{t,g} = 1$) or not (i.e., $B_{t,g} = 0$). α and β represent the importance of the load factor of gateways and the number of assigned tasks in the objective function, respectively. Constraints C1 and C2 ensure that the assigned load to each gateway must be less than the residual load capacity of the gateways. Constraint C3 imposes that each task can be allocated to exactly one gateway.

Theorem 1: The problem of fairly assigning tasks to a set of overlapping gateways is NP-complete.

Proof: Let us consider m gateways where each of which can process at most $L_{\text{res},g}, g = 1 \ldots m$, amount of workloads. Moreover, without loss of generality, we define a set of requesting tasks with different weights w_g as the load factor of the task and the value v_g as the perceived gain by the users. The goal is to assign the maximum number of tasks (e.g., maximum value) to the gateways given the maximal load capacity of the gateways. Thus, the problem is reduced in polynomial time to the known 0-1 multiple Knapsack problem [27]. Since the 0-1 multiple Knapsack problem is known as NP-complete, we can conclude that the addressed problem is NP-complete. ■

C. Heuristic Algorithm

The proposed MILP model is in the form of an NP-complete problem, thus we use the AHP algorithm to tackle the time complexity of the problem. To investigate the performance of multigateway systems, it is essential to calculate the overall latency of the network. The overall latency of the data streams (T_{tot}) consists of the *transmission delay* (T_t) toward gateways

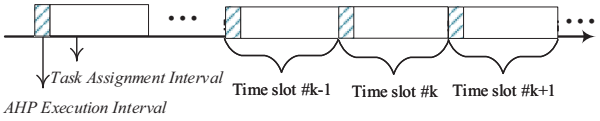


Fig. 4. Time slots in AHP method.

and the *processing delay* (T_p) by the respective gateway. Thus, we have

$$T_{\text{tot},g} = T_{t,g} + T_{p,g}. \tag{5}$$

The former delay can be neglected if the cached data are used instead of the retrieving data from sensor nodes.

The local controller consists of two components: 1) traffic monitoring module (TMM) and 2) load optimization module (LOM). The TMM is responsible for the collection, analysis, and reporting of the traffic load on each gateway. It performs the operation by separating the gateway's traffic loads and analyzing the latency of gateways. Then, it assigns them into three different lists named *over-loaded*, *under-loaded*, and *normal-load* gateways

$$L_c = \begin{cases} g \rightarrow \mathcal{G}_{\text{Overload}} & T_{\text{tot},g} > \delta_H \\ g \rightarrow \mathcal{G}_{\text{Normalload}} & \delta_M \leq T_{\text{tot},g} \leq \delta_H \\ g \rightarrow \mathcal{G}_{\text{Underload}} & T_{\text{tot},g} < \delta_M \end{cases} \tag{6}$$

where δ_H and δ_M are the upper and lower thresholds of the queuing latency, respectively. These parameters can be adjusted according to the network specifications. TMM will also collaborate with LOM in the enhancing gateway's traffic loads, with the intent to ensure fairness and reduce the network latency and the energy consumption of sensor nodes. After classifying the gateway lists regarding queue latency, LOM is responsible for assigning tasks at each time slot τ. In fact, each time slot is divided into two intervals. The first interval is allocated for running the AHP algorithm and in the second interval, the arriving tasks are allocated to the selected gateway until the end of τ (Fig. 4). This will prevent the unnecessary service delay, which is essential in delay-sensitive IoT applications.

Thus, the load-balancing mechanism consists of three iterative process.

- Step 1: TMM monitors the traffic load of the associated gateways and accordingly assign them into three different lists named as over-loaded, under-loaded, and normal-load.
- Step 2: LOM applies the AHP decision-making method for the overlapping gateways in *normal*-loaded and *under*-loaded lists to find the best suitable server for incoming tasks.
- Step 3: After assigning the tasks to each gateway, the load of the gateway is updated by the monitoring module of the controller. The server load may also change in task departure time.

Thus, in order to calculate the task processing latency of the gateways, at first, we will focus on the transmission and processing delay of each gateway in the network. The notation for the most important parameters is shown in Table I.

TABLE I
IMPORTANT PARAMETER NOTATION

Parameter	Description
m	Number of gateways
λ	Incoming task rate
λ_s	The rate of update messages sent by IoT nodes
λ_g	The arrival rate of gateway g
$P_{OR,g}$	The probability of cache miss rate in g
T_{tot}	overall latency of data streams
T_t	transmission delay
T_p	Processing delay
B_g	Service time of gateway g
V_g	Vacation duration of gateway g
τ_g	Probability of taking phase $K+1$

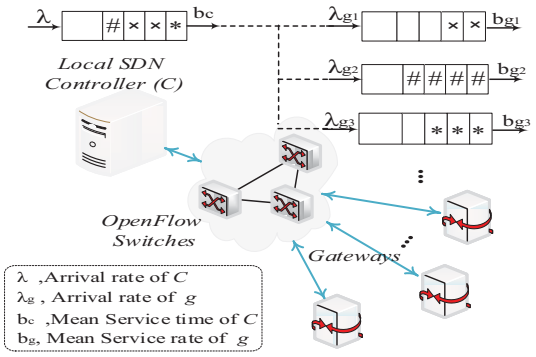


Fig. 5. Queueing model of gateway deployment.

D. Delay Analysis of Network

In this section, we suppose that the system consists of m homogeneous gateways, which independently render service to the jobs in a nonpreemptive fashion according to the FCFS discipline. The SDN controller acts as an $M/D/1$ queueing system which indicates that the interarrival time of requests exponentially distributed with a rate of $1/(\lambda = \sum_{g=1}^{m} \lambda_g)$, while task service times are deterministic and it depends on the resource capacity of the controller. When the controller receives a new job request, it processes the requests and forwards it to the respective gateway, as shown in Fig. 5.

Each gateway is modeled as an $M/G/1$ queuing system with multiple vacations model [30]. The vacation queueing models are an extension of the classical queueing theory in which the server becomes unavailable (performing nonqueueing jobs) for some potentially random period of time. The vacation process could be characterized by three aspects: 1) vacation start-up rule; 2) vacation termination rule; and 3) vacation duration distribution [31]. These models make queueing systems more realistic and flexible for real-world scenarios by allowing the server to perform nonqueueing jobs. In this case, the vacation model is applied for enabling to calculate the request processing latency induced by the controller. In other words, the latency of task processing by the controller for each stream is obtained in the gateway side. Because the controller is not able to classify the streams due to the dynamic distributions of tasks and resources. Thus, the busy period of the controller for processing the task requests of other gateways induces a vacation for each gateway. In vacation systems, the server goes on vacation for some potentially random period of time. When the server returns from a vacation and its buffer is not empty, the

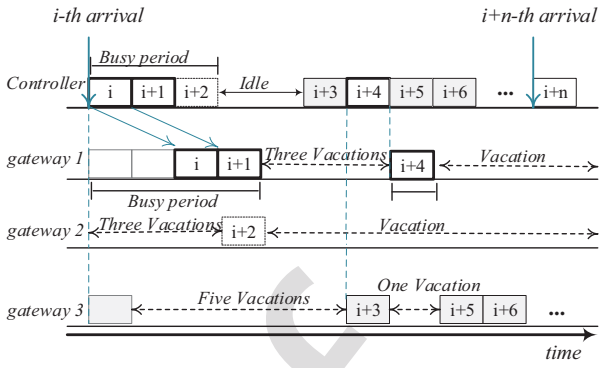


Fig. 6. System behavior with three gateways.

service period starts. As shown in Fig. 6, when the controller serves a request belonging to a gateway g, other gateways start a vacation if their queue is empty.

The task request arrival to each gateway, $g \in \mathcal{G} = \{1 \dots m\}$, follows the Poisson process with an arrival rate of λ_g and has independent and identically distributed service time with a mean value of b_g. Without loss of generality, we supposed that the service time of all gateways are the same and is $B_G(s) = B_g(s), g \in \{1, \dots, m\}$. Upon completion of a service if there are no tasks in the queue, the server takes vacation consisting of K-phases with each phase has time duration $V_{g,1}, V_{g,2}, \dots, V_{g,k}$ with the distribution functions $V_{g,k}(x) = B_c(x), g \in \{1 \cdots m\}$ and $k \in \{1 \dots K\}$. Let $\tau_g = (1 - [\lambda_g/\lambda])$ be the probability that the server takes the phase $k+1$ after completion of the phase k. Therefore, the server will take a vacation with the probability τ_g, and it will take a service period with the probability $(1-\tau_g)$. This means that taking a 1-phase vacation period could be happened with the probability of $\tau \times (1-\tau)$, and a 2-phase vacation duration has the probability of $\tau \times \tau \times (1-\tau)$. Hence, the vacation period in the system is defined as

$$\begin{aligned} V_g &= V_{g,1} \quad \text{with probability} \quad \tau_g(1-\tau_g) \\ &= V_{g,1} + V_{g,2} \quad \text{with probability} \quad \tau_g^2(1-\tau_g) \\ &\cdots \\ &= V_{g,1} + \cdots + V_{g,K-1} \quad \text{with probability} \quad \tau_g^{K-1}(1-\tau_g) \\ &= V_{g,1} + \cdots + V_{g,K} \quad \text{with probability} \quad \tau_g^{K}(1-\tau_g). \end{aligned} \tag{7}$$

Since all vacation phases have the same distribution function, i.e., $V_{g,j}(x) = B_c(x)$ for all $j \in \{1, K\}$, thus, the LST of V_g is

$$V_g^*(x) = \sum_{j=1}^{K} \left(V_{g,j}^*(x)\tau_g^j\right)(1-\tau_g) = \sum_{j=1}^{K} \left(B_c^*(x)\tau_g^j\right)(1-\tau_g) \tag{8}$$

where $V_{g,j}^*(x) = \sum_{i=1}^{j} V_{g,i}^*$. Assume that all $V_{g,j}, j \in \{1 \dots K\}$ are mutually independent. In this case, the conditional LSTs of $V_{g,j}$ can be expressed as

$$\sum_{j=1}^{K} V_{g,j}^*(x) = \left(V_g^*(x)\right)^K. \tag{9}$$

Thus, we have

$$V_g^*(x) = \sum_{k=1}^{K} \left(B_c^*(x)\tau_g\right)^k \left(1-\tau_g\right). \tag{10}$$

To model the system, a set of Markov point $\{L_n, n = 1, 2, \ldots\}$ is considered at which either a vacation is ended or a service is ended, where L_n is the number of task requests in the system immediately after the nth Markov point. Let Θ_n be the state of the gateway at the nth Markov point denoted by

$$\Theta_n \triangleq \begin{cases} 0, & \text{end of a vacation} \\ 1, & \text{end of a service.} \end{cases} \tag{11}$$

The steady-state joint distributions of the number of tasks in g are defined by

$$q_{g,k} \triangleq \lim_{n\to\infty} \text{Prob}[\Theta_n = 0, L_n = k], \quad k = 0, 1, 2, \ldots \tag{12}$$

$$\pi_{g,k} \triangleq \lim_{n\to\infty} \text{Prob}[\Theta_n = 1, L_n = k], \quad k = 0, 1, 2, \ldots \tag{13}$$

and the corresponding generating functions are defined by

$$Q_g(z) \triangleq \sum_{k=0}^{\infty} q_{g,k} z^k \tag{14}$$

$$\Pi_g(z) \triangleq \sum_{k=0}^{\infty} \pi_{g,k} z^k. \tag{15}$$

The model is governed by the following balance equations:

$$q_{g,k} = \left(q_{g,0} + \pi_{g,0}\right) f_{g,k} \qquad k \geq 0 \tag{16}$$

$$\pi_{g,k} = \sum_{j=1}^{k+1} \left(q_{g,j} + \pi_{g,j}\right) a_{g,k-j+1} \quad k \geq 0 \tag{17}$$

$$\sum_{k=0}^{\infty} q_{g,k} + \sum_{k=0}^{\infty} \pi_{g,k} = 1 \tag{18}$$

where $f_{g,k}$ is the probability that there are k arrivals for g during a vacation time that its PGF is defined as

$$F_g(z) = \sum_{k=0}^{\infty} f_{g,k} z^k = \int_0^{\infty} e^{-\lambda_g(1-z)x} dV_g(x) = V_g^*\left(\lambda_g - \lambda_g z\right). \tag{19}$$

Similarly, $a_{g,k}$ is the probability that k tasks arrive for g during a service time with the PGF of

$$A_g(z) = \sum_{k=0}^{\infty} a_{g,k} z^k = \int_0^{\infty} e^{-\lambda_g(1-z)x} dB_G(x) = B_G^*\left(\lambda_g - \lambda_g z\right). \tag{20}$$

Thus, we can write these equations as

$$Q_g(z) = \left(q_{g,0} + \pi_{g,0}\right) V_g^*\left(\lambda_g - \lambda_g z\right) \tag{21}$$

$$\Pi_g(z) = \left[Q_g(z) + \Pi_g(z) - \left(q_{g,0} + \pi_{g,0}\right)\right] \frac{B_G^*\left(\lambda_g - \lambda_g z\right)}{z} \tag{22}$$

$$Q_g(1) + \Pi_g(1) = 1. \tag{23}$$

From (22) and (23), we get

$$\Pi_g(z) = \frac{\left(q_{g,0} + \pi_{g,0}\right)\left[1 - V_g^*\left(\lambda_g - \lambda_g z\right)\right] B_G^*\left(\lambda_g - \lambda_g z\right)}{B_G^*\left(\lambda_g - \lambda_g z\right) - z}. \tag{24}$$

Substituting (22) and (25) into (24), we get

$$q_{g,0} + \pi_{g,0} = \frac{1-\rho_g}{1 - \rho_g + \lambda_g E\left[V_g\right]} \tag{25}$$

where ρ_g is the traffic intensity of g and can be defined as $\rho_g = (\lambda_g/\mu_G)$. From (17) with $k = 0$ and (26), we get

$$q_{g,0} = \frac{\left(1-\rho_g\right) V_g^*\left(\lambda_g\right)}{1 - \rho_g + \lambda_g E\left[V_g\right]} \tag{26}$$

$$\pi_{g,0} = \frac{\left(1-\rho_g\right)\left[1 - V_g^*\left(\lambda_g\right)\right]}{1 - \rho_g + \lambda_g E\left[V_g\right]}. \tag{27}$$

Let us assume that the service time of the controller is deterministic ($\mu_c = 1/D$) and the service time of each gateway exponentially distributed with a mean rate of μ_G. As $V^*(x) = B^*(x)$ and $B^*(x) = e^{-Dx}$, we have $V^*(x) = \int_0^{\infty} e^{-\lambda_g x(1-z)} dV(x) = V^*(\lambda_g - \lambda_g z)$. By substituting $(\lambda_g - \lambda_g z)$ in $V^*(x)$, we can obtain $V^*(\lambda_g - \lambda_g z) = e^{-D(\lambda_g - \lambda_g z)}$. From the above expressions, the PGF of the number of tasks is denoted as

$$Q_g(z) = \left(q_{g,0} + \pi_{g,0}\right) \sum_{j=1}^{K} \left(\tau_g e^{-D(\lambda_g - \lambda_g z)}\right)^j \left(1 - \tau_g\right). \tag{28}$$

The PGF for the number of arrivals during the service time is also denoted as $A(z) = \int_0^{\infty} e^{-\lambda_g x(1-z)} dV(x) = B^*(\lambda_g - \lambda_g z) = [\mu_G/(\mu_G + \lambda_g - \lambda_g z)]$. Thus, we have

$$\Pi_g(z) = \frac{\mu_G\left(q_{g,0} + p_{g,0}\right)\left(1 - \sum_{j=1}^{K}\left(\tau_g e^{-D(\lambda_g - \lambda_g z)}\right)^j \left(1 - \tau_g\right)\right)}{\mu_G - z\left(\mu_G + \lambda_g - \lambda_g z\right)}. \tag{29}$$

1) Processing Delay of the Gateway: Once we obtain the queue length, we are able to calculate the distribution of waiting time and response time. Let W_g denote the waiting time in steady state, and similarly, let $W_g^*(x)$ be the LST of W. In FCFS systems, the number of jobs left after departing task is equal to the number of jobs which have arrived while this task was in the system, i.e., $\Pi_g(z) = W_g^*(\lambda_g - \lambda_g z) B^*(\lambda_g - \lambda_g z)$. Thus, we have

$$W_g^*(x) = \frac{x\left(1-\rho_g\right) V_g^*(x)}{x - \lambda_g + \lambda_g B_G^*(x)\big)}. \tag{30}$$

The LST of the response time of each gateway is obtained by $T_g^*(x) = W_g^*(x) B_g^*(x)$ and the mean value of the response time is obtained as $-T_g^{*'}(0)$. Thus, we have

$$T_{p,g} = \overline{T_g} = \frac{v_g^{(2)}(1)}{2\lambda_g v_g} + \frac{\lambda_g b_G^{(2)}}{2\left(1-\rho_g\right)} + b_G \tag{31}$$

where $v_g = -(d/ds) V_i^*(0)$, $b_G = -(d/ds) B_G^*(0)$, and $b_G^{(2)} = -(d^2/ds^2) B_G^*(0)$.

2) Transmission Delay of the Gateway: We assume IoT devices are periodically sending the sensed data in exponential distribution with mean $T_s = 1/\lambda_s$. Since the data in cache have a limited validity time, a gateway checks data validity upon receiving a new request from the controller. If the data are valid, it will serve the request using data in the cache, otherwise, it will fetch the fresh data from IoT devices through sending a validation GET message. As we calculate in (2), $P_{\mathrm{OR},g}$ denote the probability of sending refresh messages toward IoT devices. Thus, the transmission time can be obtained as

$$T_{t,g} = P_{\mathrm{OR},g} \times T_{\mathrm{rtt}} = \frac{T_{\mathrm{rtt}}\lambda_g e^{-(\lambda_g+\lambda_s)\overline{T_s}}}{\lambda_g + \lambda_s} \tag{32}$$

where T_{rtt} is the round trip time at the medium access control (MAC) layer that we calculate it using the method proposed in [32]. Note that we neglect the average network delay ofthe SDN-based core switches in delivering the traffic to the clients.

E. AHP-Based Load Association Scheme

AHP is an effective multicriteria decision-making method that is used for finding the best case among other feasible alternatives in decision-making problems such as load management.

Decision making in this method is done based on a set of evaluation metrics, and through nominating convenient weights for each metrics that are derived from experts. We apply a modified AHP that is able to integrate network statistics in the ranking procedure.

1) Ranking the Gateways: Although it seems preferable to send task requests to the nearest gateways with more computational resources, this may lead to an unbalanced processing load among gateways. This can affect the task's completion time, which is not suitable in delay-sensitive applications. Thus, network parameters, such as *available resources*, *link quality*, and *resource load* can be considered as comparison metrics in task associations to the gateways.

In this method, a priority is assigned to each gateway with respect to the weights of different metrics, in which these weights are defined in $1 \leq i \leq 9, i \in Z$ for each metric according to the AHP algorithm. The resource load can be applied as the main metric for balancing the loads among gateways since it reflects the current load status of each gateway in the network. Resource load index (LI) is defined as the remaining service time of the current task, in addition to the service time of the tasks that are waiting in the queue. The LST of remaining service time can be defined as

$$B_g^+(x) = \frac{1 - B_g^*(x)}{x\overline{b_{p,x}}}. \tag{33}$$

Thus, LI can be obtained by $L_g^*(x) = B_g^+(x)W_g^*(x)$, and $\overline{L_g} = -L_g^{*'}(0)$.

Let us assume that the gateway g_1 is the overloaded gateway, and the gateways g_2 and g_3 are the candidate overlapping gateways with g_1. First, we calculate the preference of selecting each gateway in a comparison matrix as in Table II, where $p_{ij} = (\overline{L_j}/\overline{L_i})$. From the last relation, we deduce $p_{ij} = (1/p_{ji})$.

Algorithm 1 AHP-Based Load Association

1: Initialization:
a) Pk ← arrival task
b) $\mathcal{G}_g$ ← overlapping gateways of g
c) $q_{over} = \emptyset$, List of overloaded gateways
d) $q_{normal} = \emptyset$, List of normalloaded gateways
e) $q_{under} = \emptyset$, List of underloaded gateways
2: For g from 1 to m do
3: If Pk is for gateway g
4: calculating $T_{tot,g} = T_{t,g} + T_{p,g}$
5: If $T_{tot,g} > \delta_H$
6: sending g into q_{over}
7: for all i in $\mathcal{G}_g$ do
8: calculating $\overline{L}_i$ according to (31)
9: calculating the priorities according to Table II-V
10: selecting the most suitable $i \in \mathcal{G}_g$
11: sending Pk to gateway i
12: end for
13: elseif $\delta_L < T_{tot,g}$
14: sending g into q_{under}
15: sending Pk to g
16: else
17: sending g into q_{normal}
18: sending Pk to g
19: end if
20: end if
21: end for

TABLE II
RESOURCE LOAD COMPARISON

Resource Load	g_1	g_2	g_3
g_1	1	p_{12}	p_{13}
g_2	p_{21}	1	p_{23}
g_3	p_{31}	p_{32}	1

TABLE III
PREFERENCE MATRIX

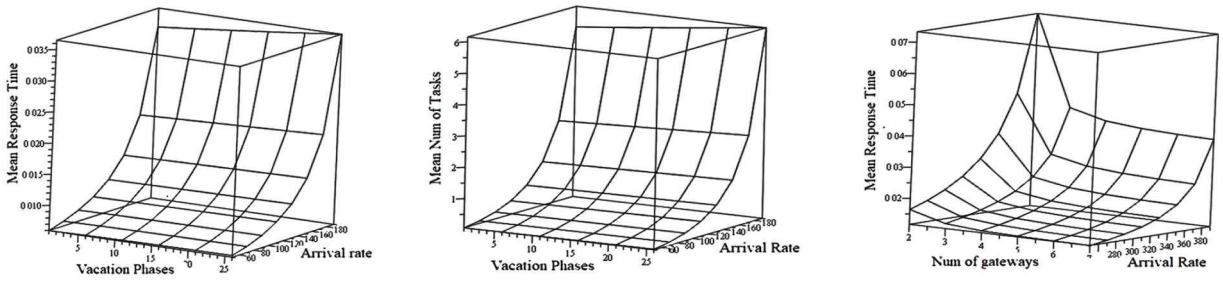
RL	g_1	g_2	g_3
g_1	$a_{11}=\frac{1}{1+p_{21}+p_{31}}$	$a_{12}=\frac{p_{12}}{p_{12}+1+p_{32}}$	$a_{13}=\frac{p_{13}}{p_{13}+p_{23}+1}$
g_2	$a_{21}=\frac{p_{21}}{1+p_{21}+p_{31}}$	$a_{22}=\frac{1}{p_{12}+1+p_{32}}$	$a_{23}=\frac{p_{23}}{p_{13}+p_{23}+1}$
g_3	$a_{31}=\frac{p_{31}}{1+p_{21}+p_{31}}$	$a_{32}=\frac{p_{32}}{p_{12}+1+p_{32}}$	$a_{33}=\frac{1}{p_{13}+p_{23}+1}$

TABLE IV
FINAL PRIORITIES

Resource Load	g_1	g_2	g_3	priority
g_1	1	p_{12}	p_{13}	$w_1=\frac{a_{11}+a_{12}+a_{13}}{3}$
g_2	p_{21}	1	p_{23}	$w_2=\frac{a_{21}+a_{22}+a_{23}}{3}$
g_3	p_{31}	p_{32}	1	$w_3=\frac{a_{31}+a_{32}+a_{33}}{3}$

To obtain the preference of the gateways, we calculate the priority of each gateway with respect to the resource load according to the AHP algorithm. The preference matrix, Table III, is obtained by dividing each element with the sum value of each column, i.e., $a_{ij} = [p_{ij}/(p_{ij} + p_{jj} + pkj)]$.

Finally, the priorities can be obtained by normalizing the preference matrix as $w_i = [(a_{ii} + a_{ij} + a_{ik})/3]$, shown in Table IV.



(a) (b) (c)

Fig. 7. Main descriptors of the IoT system with different parameters. (a) Mean response time of each gateway. (b) Mean number of tasks in each gateway. (c) Mean response time of the network.

TABLE V
RESOURCE LOAD COMPARISON & PRIORITY

Resource Load	g_1	g_2	g_3	priority
g_1	1	0.6	0.5	**0.22**
g_2	1.48	1	0.7	**0.33**
g_3	1.87	1.25	1	**0.44**

Let us assume that LI of the three gateways are calculated as $L_1 = 2.61$ ms, $L_2 = 1.75$ ms, and $L_3 = 1.39$ ms, respectively. Thus, the comparison matrix for these three gateways is obtained as Table V.

After calculating the priorities of each gateway with respect to the preferred metrics, the convenient gateway can be selected based on these priorities. To ensure the consistency of prioritization in metric weights, the consistency ratio (CR) can be calculated to check the degree of consistency, which should be CR ≤ 0.1 [21]. The details of the MCDM-based load-balancing algorithm is shown in the following pseudocode. As seen, the calculations are done for each of the overlapping gateways in the network, and the number of overlapping gateways is limited ($2 < m < 4$). The overall computational complexity of computing the exact stationary queue length distribution is $O(K^2)$ [33], where K is the buffer size. Note that in our algorithm, the buffer size cannot be infinite due to the upper threshold (δ_H) for the server load. On the other hand, time complexity of AHP is in $O(\min\{mn^2, m^2n\})$ time, where m is candidates and n is metrics [34]. Thus, the total complexity is $O(K^2) + O(\min\{mn^2, m^2n\})$.

IV. PERFORMANCE EVALUATION

To evaluate the performance of the load-balancing model, we consider a single-hop 6LoWPAN-based WSN with multiple overlapping gateways. The analytical model has been solved using Maple 16 from Maplesoft, Inc., [35] and MATLAB is used to run different simulation scenarios. The performance of the network is analyzed in a number of different scenarios.

In the first step, we evaluate the analytical model with different parameters, such as task arrival rate and different vacation phases. Fig. 7 illustrates the effect of queuing parameters on the performance measures of the model. In the first scenario, we have considered multivacation-based queueing model with different arrival ratio, varying from $\lambda_g = 0.2\lambda$ to $\lambda_g = 0.8\lambda$ task arrival per second to an arbitrary gateway ($\lambda = 250$ arrivals per second). To investigate the effect of the number of phases of vacation on the system performance, we assumed each gateway can take different vacation phases from $K = 1$ to $K = 30$ in each iteration. The execution time of each gateway is exponentially distributed and the average gateway execution time is set to $(1/\mu_G) = 45$ ms.

The mean response time for each gateway g is shown in Fig. 7(a). As it is observed from the figure, increasing the number of vacation phases induces a higher level of response time for the gateway. However, the delay imposed by the vacation period is more noticeable in the higher arrival rate. Because in higher rates of task arrival, the length of vacation period taken by the gateways is longer than lower rates.

Fig. 7(b) shows the number of tasks waiting in the queue for each gateway. As can be seen, the mean number of tasks is increasing function with respect to both request arrival rate and the number of phases of vacation. However, when $\lambda_g = 200$, the graph rises with a steep slope as compared to $\lambda_g = 50$. The reason for this is due to the fact that when the task arrival rate increases, the probability of taking longer vacation periods rises for each gateway. Thus, the graph rises with the steeper rate in higher arrival rates.

In the next step, we consider the end-to-end delay of the network in different scenarios. First, the average delay of the SDN-based IoT domain is considered with different arrivals from $\lambda = 250$ to $\lambda = 400$, and varying number of gateways (from 2 to 7) in the system. The maximum vacation phases is set to $K = 30$, and the average execution time of the gateways is considered $1/\mu_G = 45$ ms, respectively. Fig. 7(c) shows that the average delay of the tasks increases with the number of gateways whilst it shows a small rise with arrival rate. The reason for this is the contention of additional gateways on the controller and longer vacation periods that are induced for each gateway.

The next scenario evaluates the effectiveness of MCDM-based load balancing in comparison with the conventional round robin (RR) and response time-based load-balancing method [26]. RR load balancing is a way that widely used to distribute client requests among the servers due to its simplicity. As known in RR, the requests are forwarded to each server in turn and the algorithm instructs the load balancer to jump back to the beginning of the list once the end is reached

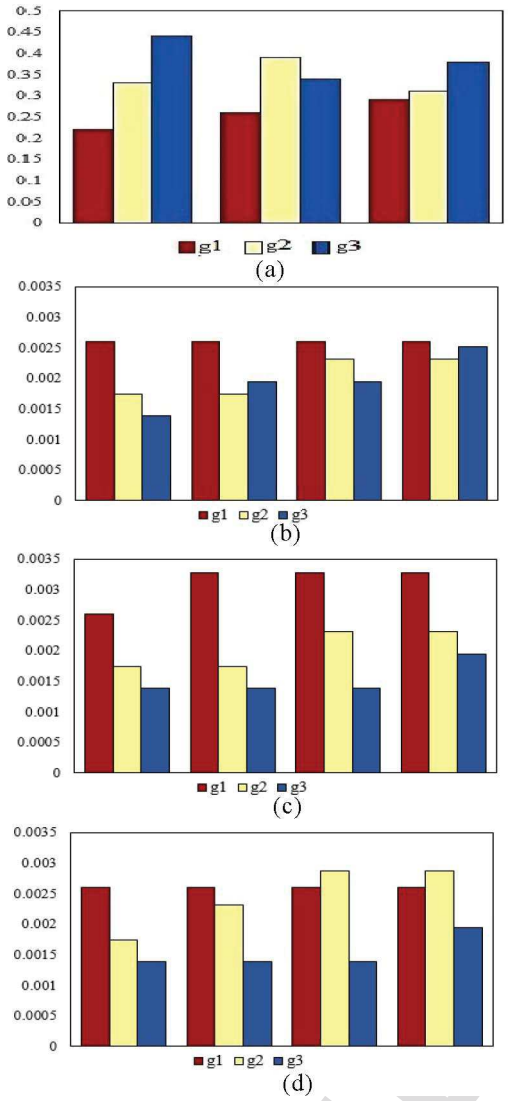


Fig. 8. (a) Priority of gateways over time. (b) Loads in MCDM-based method. (c) Loads in RR method. (d) Loads in LBBSRT.

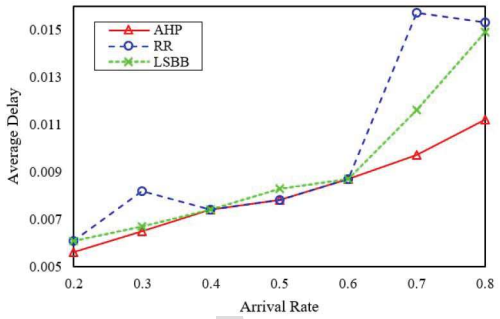


Fig. 9. Network latency in three methods.

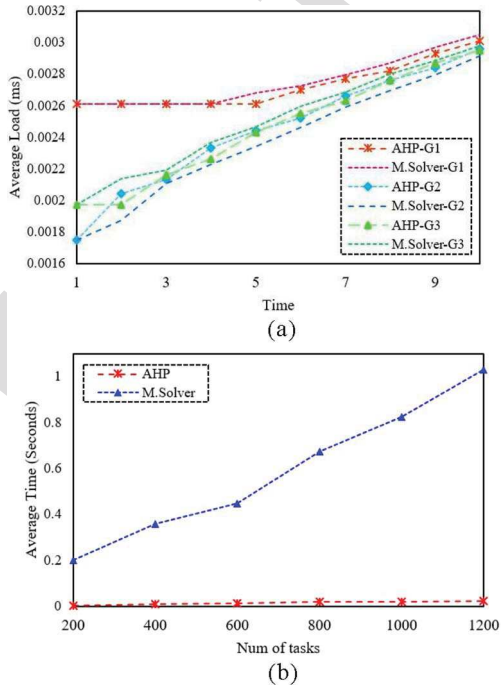


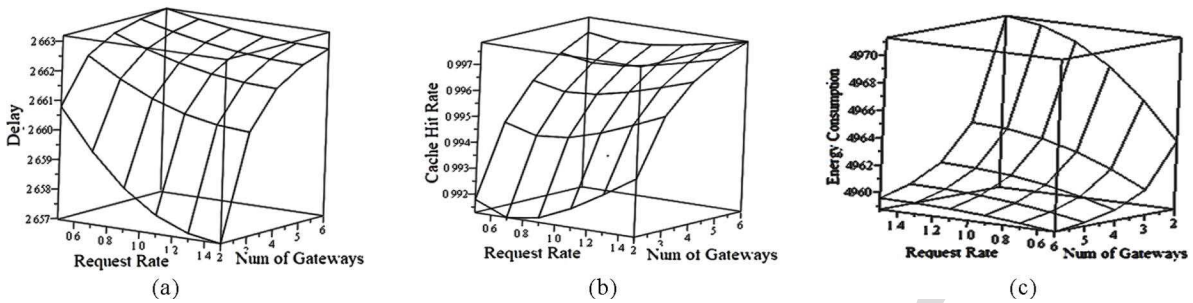
Fig. 10. (a) Average load of gateways. (b) Average execution time.

and repeats again. This solution can achieve a better response time than a random load-balancing strategy [28]. However, considering the server delay in traffic load balancing can help to obtain better response time. In addition, server load may impact on selecting the appropriate threshold for the response times. So, we take into account the weighted index of network features in balancing the traffic loads.

In order to evaluate it, a domain with three overlapping gateways are considered with LI of $L_1 = 2.61$ ms, $L_2 = 1.75$ ms, and $L_3 = 1.39$ ms for each gateway. Fig. 8(a) illustrates the variation of the gateways priority with respect to their Resource LI. As can be seen, the priority of the gateways almost converges to the same value over time as the load is assigned to them in a fairly manner. Fig. 8(b)–(d) demonstrates the comparative results of the LI of the gateways in three methods. The graphs clearly depict that the proposed method evenly dispatches the loads among the gateways, while other methods lead to the partly unfair load association among resources.

To investigate the performance of the network in three methods, the average delay of task execution for three gateways is calculated with the arrival task rate in Fig. 9. The figure shows the variable delay experienced by gateways in the RR method due to the unbalanced allocations of the requests by the arrival rate. As expected, the average delay increases with the arrival rate for all approaches. However, this rate rises with a steep slope for the RR method as compared to two other methods. This graph clearly shows the effectiveness of the MCDM-based load-balancing approach as compared to RR and LBBSRT in providing a fast response to the user queries.

In the next experiment, we evaluate the performance of the MILP version and the AHP algorithm in terms of the average load of gateways [Fig. 10(a)] and average execution time [Fig. 10(b)]. As the graphs illustrate, the average load of the gateways is almost the same in both approaches. However, by increasing the number of tasks, the average execution time of the MILP-based model shows a notable growth due to the time complexity issue, while the AHP algorithm determines a reasonable solution for all cases of task number.



(a) (b) (c)

Fig. 11. Main descriptors of the system when number of the gateways varies between 2 and 6, maximum allowed phases of vacation is 30, λ varies between 0.5 and 0.15 arrivals per minute, and service rate of the gateways $\mu_G = 220$, respectively. (a) End-to-end delay in network. (b) Probability of cache hit in a gateway. (c) Energy consumption of network.

A. Performance of the Combined Load-Balancing and Caching Strategy

In this section, we evaluate the performance of the proposed model over IEEE 802.15.4, the total number of IoT devices is set to 200 sensor nodes. We assumed the packet size of 127 B with a bit error rate of $BER = 10^{-5}$. The Poison arrival rate of requests is varied from 0.5 to 1.5 arrivals per minute, and the average service time of the gateways is 22 ms. Fig. 11 shows the performance measure of the network. We noted that the end-to-end delay of the network increases with the number of the gateways. The reason for this is the increased vacation periods induced to the proxies in a higher number of the gateways. However, this parameter falls with the request arrival rate as incoming tasks benefit from the updating work by validations triggered by queries from previous tasks. As a result, the cache hit rate increases with a higher rate of user queries.

Fig. 11(b) illustrates the combined effect of the gateways and the arrival rate in the proxy cache hit rate. As noted, the cache hit rate is increased by rising the number of gateways. However, this rate shows a mild decrease with the user request rate in the beginning due to the increased amount of outdated entries in the cache. Then, it begins to increase as the new arrival requests benefits from the updated data cached by the queries from previous tasks. As mentioned previously, the network latency consists of processing delay and transmission delay. So, the increased ratio of cache miss will lead to the increased latency of the network. This, in turn, can impact on the energy consumption of the sensor nodes.

The energy consumption per node, shown in Fig. 11(c), is decreased by increasing the number of gateways. Again, the impact of the user request rate is visible in increasing the energy consumption per node. Thus, an appropriate load association scheme can lead to save the energy of IoT devices. Finally, we note that energy consumption can decrease with increasing the number of gateways as it distributes the traffic loads among the gateways and accordingly reduces the cache miss rate and recaching operations in the gateways.

As seen from the results, the proposed multigateways architecture along with a resource caching policy can provide fast responses and reduced energy consumption for IoT devices. Furthermore, AHP-based load balancing can help to achieve a global load fairness among the network entities taking into account their load changes over time. Considering a general vacation queueing model in the evaluation and the IPv6 protocol stack over sensor devices in implementation makes the evaluation closer to the real-world models. However, a preferential treatment for different traffic flows with variable evaluation metrics in AHP can also be applied to improve the Quality of Experience (QoE) for different IoT applications.

V. Conclusion

In this article, we have proposed and analyzed the multiple vacation-based queuing system to model the performance of the IoT edge network. In order to speed up the user's access to the sensor data, we leveraged a multigateways architecture along with a resource caching policy in the IoT domain. An MCDM-based load-balancing technique is also employed to provide a global load fairness among the network entities. The proposed model facilitates the remote access from the Internet to IPv6 IoT devices by leveraging the multigateway architecture over IEEE 802.15.4 connectivity. The evaluation results show the effectiveness of the proposed solution in the fast and reliable acquisition of big data from the IoT domain, which is promising to drive IoT development. In future work, we plan to address the scenario in which the network has different traffic classes of task request which are associated to the gateways according to their QoS requirements.

References

[1] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on Internet of Things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1125–1142, Oct. 2017.

[2] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," *Comput. Netw.*, vol. 54, no. 15, pp. 2787–2805, 2010.

[3] G. Xu, W. Yu, D. Griffith, N. Golmie, and P. Moulema, "Toward Integrated distributed energy resources and storage devices in smart grid: Modeling, analysis, and evaluation," *IEEE Internet Things J.*, vol. 4, no. 1, pp. 192–204, Feb. 2017.

[4] S. Hong *et al.*, "SNAIL: An IP-based wireless sensor network approach toward the Internet of Things," *IEEE Wireless Commun.*, vol. 17, no. 6, pp. 34–42, Dec. 2010.

[5] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On reducing IoT service delay via fog offloading," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 998–1010, Jan. 2018.

[6] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 packets over IEEE 802.15.4 networks," Internet Eng. Task Force, RFC 4944, Sep. 2007.

[7] T. Winter *et al.*, "RPL: IPv6 routing protocol for low-power and lossy networks," IETF, RFC 6550, 2012.

[8] Z. Shelby, K. Hartke, and C. Bormann. (Jun. 2014). *Constrained Application Protocol (CoAP), Internet Draft*, [Online]. Available: http://datatracker.ietf.org/wg/core/charter

[9] Z. Sheng, H. Wang, C. Yin, X. Hu, S. Yang, and V. C. M. Leung, "Lightweight management of resource constrained sensor devices in Internet-of-Things," *IEEE Internet Things J.*, vol. 2, no. 5, pp. 402–411, Oct. 2015.

[10] M. Kovatsch, S. Duquenony, and A. Dunkels, "A low-PoWer CoAP for Contiki," in *Proc. IEEE 8th Int. Conf. Mobile Ad Hoc Sensor Syst.*, Oct. 2011, pp. 855–860.

[11] T. Suganuma, T. Oide, S. Kitagami, K. Sugawara, and N. Shiratori, "Multiagent-based flexible edge computing architecture for IoT," *IEEE Netw.*, vol. 32, no. 1, pp. 16–23, Jan./Feb. 2018.

[12] M. Armbrust *et al.*, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[13] L. Wang and R. Ranjan, "Processing distributed Internet of Things data in clouds," *IEEE Cloud Comput.*, vol. 2, no. 1, pp. 76–80, Jan./Feb. 2015.

[14] S. Arshad, M. A. Azam, M. H. Rehmani, and J. Loo, "Recent advances in information-centric networking based Internet of Things (ICN-IoT)," *IEEE Internet Things J.*, vol. 6 no. 2, pp. 2128–2158, Apr. 2019.

[15] X. Chen, Q. Shi, L. Yang, and J. Xu, "ThriftyEdge: Resource efficient edge computing for intelligent IoT applications," *IEEE Netw.*, vol. 32, no. 1, pp. 61–65, Jan./Feb. 2018.

[16] X. Sun and N. Ansari, "Traffic load balancing among brokers at the IoT application layer," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 1, pp. 489–502, Mar. 2018.

[17] Q. Fan and N. Ansari, "Workload allocation in hierarchical cloudlet networks," *IEEE Commun. Lett.*, vol. 22, no. 4, pp. 820–823, Feb. 2018.

[18] Q. Fan and N. Ansari, "Application aware workload allocation for edge computing-based IoT," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2146–2153, Jun. 2018.

[19] Q. Fan and N. Ansari, "Towards workload balancing in fog computing empowered IoT," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 1, pp. 253–262, Jun. 2018.

[20] G. Grimmett and D. Stirzaker, *Probability and Random Processes*, 3rd ed. Oxford, U.K.: Oxford Univ. Press, Jul. 2010.

[21] T. L. Satty, "Decision making with the analytic hierarchy process," *Service Sci.*, vol. 1, no. 1, pp. 83–98, 2008.

[22] Z. Gao, X. Li, H. Zhang, and V. C. M. Leung, "A load balance self-healing algorithm based on AHP for backhaul transmission in UDNs," in *Proc. Netw. Infrast. Digit. Content (IC-NIDC)*, 2018, pp. 315–319.

[23] A. M. Abdullah, H. A. Ali, and A. Y. Haikal, "A reliable TOPSIS-based multi-criteria and hierarchical load balancing method for computational grid," *Clust. Comput.*, vol. 22, pp. 1085–1106, Jan. 2019.

[24] A. Lara, A. Kolasani, and B. Ramamurthy, "Network innovation using OpenFlow: A survey," *IEEE Commun. Survey Tuts.*, vol. 16, no. 1, pp. 483–512, 1st Quart., 2014.

[25] W. Wang, S. De, R. Toenjes, E. Reetz, and K. Moessner, "A comprehensive ontology for knowledge representation in the Internet of Things," in *Proc. 11th Int. Conf. Trust Security Privacy Comput. Commun. (TrustCom)*, 2012, pp. 1793–1798.

[26] J. Cui, Q. Lu, H. Zhong, M. Tian, and L. Liu, "A load-balancing mechanism for distributed SDN control plane using response time," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 4, pp. 1197–1206, Dec. 2018.

[27] C. Cotta and J. M. Troya, "A hybrid genetic algorithm for the 0–1 multiple knapsack problem," in *Artificial Neural Nets and Genetic Algorithms*. Heidelberg, Germany: Springer. 1998, pp. 250–254.

[28] S. Kaur, K. Kumar, J. Singh, and N. S. Ghumman, "Round-robin based load balancing in software defined networking," in *Proc. Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, 2015, pp. 2136–2139.

[29] J. Misic, V. B. Misic, and F. Banaie, "Reliable and scalable data acquisition from IoT domains," in *Proc. IEEE GlobeCom*, Singapore, 2017, pp. 1–6.

[30] H. Takagi, *Queuing Analysis: Vacation and Priority Systems*, vol. 1. Amsterdam, The Netherlands: North-Holland, 1991.

[31] N. Tian and Z. Zhang, *Vacation Queueing Models: Theory and Applications*. Berlin, Germany: Springer, 2006.

[32] J. Misic and V. B. Misic, *Wireless Personal Area Networks: Performance. Interconnections and Security With IEEE 802.15.4*. Hoboken, NJ, USA: Wiley, 2008.

[33] H. Li and Y. Zhu, "M(n)/G/1/N queues with generalized vacations," *Comput. Oper. Res.*, vol. 24, no. 4, pp. 301–316, 1997.

[34] N. Z. Mamat and J. K. Daniel, "Statistical analyses on time complexity and rank consistency between singular value decomposition and the duality approach in AHP: A case study of faculty member selection," *Math. Comput. Model.*, vol. 46, no. 7, pp. 1099–1106, 2007.

[35] (2015). *Maplesoft, Maple 16*. [Online]. Available: http://www.maplesoft.com/products/maple

Fatemeh Banaie (Student Member, IEEE) received the M.Sc. degree in software computer engineering from the Ferdowsi University of Mashhad, Mashhad, Iran, in 2012, where she is currently pursuing the Ph.D. degree with the Computer Engineering Department.

From November 2016 to November 2017, she was a Visiting Research Scholar with the Department of Computer Science, Ryerson University, Toronto, ON, Canada. Her research interests are in the area of cloud networking, resource management, and Internet of Things.

Mohammad Hossein Yaghmaee (Senior Member, IEEE) is a Full Professor with the Computer Engineering Department, Ferdowsi University of Mashhad, Mashhad, Iran. From November 1998 to July 1999, he was a Visiting Research Scholar with the Network Technology Group, C&C Media Research Laboratories, NEC Corporation, Tokyo, Japan. From September 2007 to August 2008, he was a Visiting Associate Professor with the Lane Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV, USA. From July 2015 to September 2016, he was a Visiting Professor with the Electrical and Computer Engineering Department, University of Toronto, Toronto, ON, Canada. His research interests are in smart grid communication, software-defined networking, and network function virtualization.

Seyed Amin Hosseini received the B.Sc. and M.Sc. degrees in computer engineering from the Ferdowsi University of Mashhad, Mashhad, Iran, in 1990 and 1998, respectively, and the Ph.D. degree in computer network from the University of Science, Malaysia, in 2010.

He is the Dean of the Information and Communication Center, Ferdowsi University of Mashhad. His research interests include wireless networks, energy efficiency protocols, and network security.

Farzad Tashtarian is an Assistant Professor with the Department of Computer Engineering and the Head of the System and Computer Networking Research Center, Islamic Azad University of Mashhad, Mashhad, Iran. He is also a Member of the Young Researchers and Elite Club, Mashhad Branch, Islamic Azad University. From February 2013 to October 2013, he was a Visiting Scholar with the University of Missouri Kansas City, Kansas City, MO, USA. He has coauthored several research papers and published in well-known conferences and journals, including IEEE Global Communications Conference, IEEE/ACM International Symposium on Quality of Service, the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, and *Future Generation Computer Systems* (Elsevier). He is broadly interested in software-defined networking, network function virtualization, multimedia streaming, Internet of Vehicles, mathematical modeling, and distributed optimization.