Egyptian Informatics Journal 23 (2022) 215–224

Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

Full length article

ESRA: Energy soaring-based routing algorithm for IoT applications in software-defined wireless sensor networks

N. Samarji [*], M. Salamah

Department of Computer Engineering, Eastern Mediterranean University, 99628, Famagusta, North Cyprus via Mersin 10, Turkey

A R T I C L E   I N F O

Article history:
Received 8 June 2021
Revised 14 October 2021
Accepted 1 December 2021
Available online 30 December 2021

Keywords:
Controller Placement Problem
Energy Efficiency
Genetic Algorithm
Software-Defined Networking

A B S T R A C T

Software-defined wireless sensor networking is an emerging networking architecture envisioned to play a critical role in the looming internet of things paradigm. Since energy is a scarce resource in wireless sensor networks, many energy-efficient routing algorithms were proposed to enhance the network lifetime. However, most of these algorithms lack network stability and reliability in the presence of dead nodes. This paper presents ESRA: Energy Soaring-based Routing Algorithm for IoT Applications in Software-Defined Wireless Sensor Networks, specifically for monitoring environment to address this shortcoming. The proposed ESRA algorithm efficiently selects the network cluster heads to be considered for solving the controller placement problem, intending to achieve network reliability and stability and enhance the network lifetime. The selection of controllers among the cluster heads is formulated as an NP-hard problem, considering the residual energy of the cluster heads, their spatial distance to the sink, and their load or density. To tackle this NP-hard problem, genetic algorithm is adopted to optimize the network lifetime, throughput, latency, and network reliability in the presence of different percentages of dead nodes. Simulation results showed that ESRA outperforms other three state-of-the art algorithms in terms of network lifetime and throughput by 15%, 20%, and 25%, in terms of energy savings by 10%, 20%, and 25%, and in terms of delay by 10%, 15%, and 20%. We also applied the proposed scheme on real networks adopted from the internet topology zoo, which showed promising results compared to other existing works.

1. Introduction

Software-Defined Networking (SDN) is a new hot topic for many researchers and has been implemented in different environments, especially in Wireless Sensor Networks (WSNs). In WSNs, the promising goal is to obtain energy efficiency and maximize the network lifetime. Hence, achieving this goal is usually done by clustering the network that saves the nodes' energy by minimizing the transmission distance between the nodes. However, an important challenge related to clustering is the cluster head (CH) selection technique that directly affects the network performance. As it is well known that CHs spend more energy than the cluster members, after a certain period, the CHs' energy becomes exhausted. This will lead to a disconnection in the network connectivity. Hence, CHs should be changed periodically and efficiently selected to balance the overall network energy [1]. When the energy consumption is balanced among the sensor nodes, network stability, as well as the network lifetime will be maximized [2,3].

The fundamental structure of the SDN lies on the separation of the control plane from the data plane. The control plane consists of one or more intelligent devices [4,5] called the controller(s) which handle the decision policies and forwarding rules of the network. On the other hand, the data plane consists of the forwarding devices, such as the routers and switches which forward the packets according to the forwarding policies assigned by the controller(s).

* Corresponding author.
E-mail addresses: nivine.samarji@emu.edu.tr (N. Samarji), muhammed.salamah@emu.edu.tr (M. Salamah).

Peer review under responsibility of Faculty of Computers and Information, Cairo University.

Production and hosting by Elsevier

Despite the flexibility that SDN offers to network management due to detaching the control and data planes, any disconnection between these planes will have a fatal impact on QoS and the network performance [6]. Hence, to prevent a single point of failure, a multi-SDN controller-based network becomes a promising solution [6,7]. The implementation of a multi-SDN controller in a network has showed noticeable improvements in overall network performance [8,9]. For this reason, many researchers have focused mainly on using a multi-controller based network. However, a challenging matter that directly impacts the network performance when using multi-controller is the number and the position of the SDN controllers in the network, which is known as the Controller Placement Problem (CPP) [10,11]. Hence, in SDN-based WSNs, CHs should be carefully chosen since the required SDN controllers are usually placed at the CHs. Therefore, the CPP may not be efficiently solved in case a wrongly chosen CH locates a controller. Consequently, the network performance may experience a noticeable degradation.

Besides the cluster head selection challenge, attaining the network stability or steady-state in the presence of dead nodes is another challenge to be addressed. The network steady-state is maintained when no overloaded or overwhelmed controller(s) exists. The density of a node, or in other words its load, causes a quick depletion of its energy. As a result, the network experiences early dead node occurrence. Hence, the network steady-state cannot be achieved due to the disconnection in its connectivity. When a controller becomes overloaded, its response time exceeds a certain threshold, causing an increase in network delay. Also, more packet-loss occurs as the load of the controller reaches its upper bound. Therefore, in the presence of overloaded controller(s), an efficient migration strategy is needed to balance the load among the controllers. However, this migration should not cause an overwhelmed receipt controller. Moreover, the network steady-state cannot be attained in the presence of dead controller(s) since the network policies, flow tables, and the network topology are obtained on the behalf of the controller.

Motivated by these challenges mentioned earlier, we propose the ESRA algorithm by leveraging the SDN concept in WSNs. In our proposed method, each SDN controller applies the energy-soaring routing algorithm adopted from the albatross bird to efficiently select the CHs in its domain. The albatross flying technique allows the bird to travel long distances using the windshields with minimum effort and few flaps of its wings [12]. First, the root controller executes the k-way spatial clustering algorithm to partition the network into disjoint clusters. The disjoint clusters play a significant role in maintaining the network reliability, especially in the presence of dead nodes and links. Packet transmission can still be performed via alternative paths of the disjoint clusters in the presence of dead nodes or links. The network CHs are then included in the SDN controller selection by applying the Genetic Algorithm (GA). After finding the optimal position of the SDN controllers, each controller distributes the nodes among the CHs by considering the nodes' position. Each controller assists the load balancing of its domain by distributing the load of its overwhelmed CH(s). In the presence of dead CH(s), the controller runs the GA to select new CH(s). The root controller maintains the network steady-state that ensures the functionality of controllers by executing the Network Stability Algorithm (NSA). A node is considered dead in the proposed ESRA algorithm only if its energy falls below a threshold. However, it is considered overloaded or overwhelmed if its response time is above a specific threshold.

So far, in literature, there is no work that addresses the CPP in WSNs by considering two important factors. The first is the presence of dead nodes, and the second is the adoption of the albatross flying technique for the CHs selection.

The paper contributions can be listed as follows:

a) The CPP is solved by applying the GA, where the fitness function optimizes various network performance metrics such as the network latency, throughput, reliability, and energy saving in the presence of different percentages of dead nodes.

b) A novel clustering algorithm is presented to efficiently find the network CHs by adopting the natural flying soaring technique of the albatross bird. This clustering algorithm proves to be an energy-efficient where the network lifetime is enhanced.

c) The concept of SDN is used in WSN where the SDN controller is responsible for clustering the network, maintaining the network steady-state in the presence of different percentages of dead nodes by applying the NSA algorithm.

d) Network performance evaluation of the ESRA algorithm is compared with different state-of-the-art algorithms where the proposed algorithm shows its superiority over other algorithms. In addition, the effectiveness of ESRA algorithm is analyzed by applying it on some real datasets topologies.

We simulated the proposed ESRA algorithm and analyzed various important network performance metrics such as network lifetime, throughput, network latency, and total energy consumption under different percentages of dead nodes. For comparison purposes, we have chosen recent existing energy-aware algorithms found in literature such as Energy-Efficient Fault-Tolerant Clustering Algorithm for Wireless Sensor Networks (EEFCA) [13], Gateway Clustering Energy-efficient Centroid (GCEEC) for Wireless Sensor Networks in Agriculture Precision [14], and Energy-Efficient Clustering Routing Protocol for Wireless Sensor Networks Based on Yellow Saddle Goatfish Algorithm (YSGA) [15]. Simulation results exhibit that the proposed ESRA algorithm shows significant network performance improvement over the above chosen state-of-the-art algorithms.

The remaining of the paper is organized as follows: Section II describes the related work in literature. Section III describes the motivation. Section IV describes system overview. Section V covers the performance evaluation issues. Finally, Section VI concludes the work.

2. Related work

Traditional network management is a complex task that can't cope with the growth of the network. Hence, implementing SDN in a network solves the traditional network restrictions by flexibly managing the network and coping with the current network demands. However, the CPP is a challenging issue that arises with the presence of multi-controller in a network. Heller *et al.* [6] were the first to present the controller placement problem considering the network latency. They proved that having one controller meets the network requirements; however, fault resiliency is not considered. By adopting the work of Heller *et al.* [6], many authors have focused on improving network performance.

In the context of enhancing the network energy, Hu *et al.* [16] solved the CPP by focusing on enhancing the SDN-based network energy. Authors in [17] focus on maintaining energy efficiency in SDWSNs by reducing the number of generated data packets with two main concepts: implementing the content awareness at each sensor node to decide whether to send the data or not to the controller; and adaptive data broadcast, which replaces the packet transmission from the data plane with packet transmission in the control plane. Although the presented algorithm enhances the network lifetime, the idea of transmitting the data in the control plane instead of the data plane contradicts the fundamental concept of the SDN. Also, in the presented work, each sensor node sends a hello message with a frequency higher than the packet transmission frequency to let the controller knows if the node is still alive or not. Hence, in the presence of overwhelmed nodes, this method does not work well as the controller will assume that the node is dead. This false assumption degrades the network throughput and QoS.

Killi and Rao [18] solved the CPP by maintaining the network reliability and load balancing among controllers. A heuristic algorithm (namely, simulated annealing) is used for large-scale networks. Their proposed scheme assumes a node is dead if its response time exceeds a given threshold, which is not always the case. A node with response time above a threshold might be overloaded and not dead.

Samarji and Salamah [19] proposed a Fault Tolerance Metaheuristic-Based Scheme for Controller Placement Problem in Wireless Software Defined Networks (FTMBS) to optimize the network fault resiliency in solving the CPP. In their proposed scheme, CHs are chosen with the highest energy in each round without considering the node's location from the sink or the base station.

Qureshi *et al.* [14] proposed a load management scheme named Gateway Clustering Energy-Efficient Centroid (GCEEC) to address the load burden issues caused by sensor nodes that relay their transmission data to those that are close to the base station. In their scheme, the CHs nodes are chosen from the mean position, and the gateway nodes transmit the load of the overwhelmed CHs to the base station. The experimental results showed that the proposed GCEEC scheme is an energy-efficient algorithm in comparison with other schemes.

Luo *et al.* [16] solved the CPP by maximizing the network energy saving. They modeled their problem as Binary Integer Problem (BIP) which is good for small scale networks and used GA for large scale networks. In their scheme, they assumed each active link consumes the same energy; thus, they aim is to have the least number of active links. However, the drawback lies in their assumption, as nodes have different data to send on the links. So, this false assumption cannot hold true unless the same number of nodes use each link with the same transmission flow.

Cui *et al.* [20] presented a load balancing scheme for solving the CPP. In their scheme, they assume a controller is dead if its response time is above a certain threshold. Accordingly, new controller is to be chosen from CHs instead of the dead one. However, the controller could have been overloaded and if so, the load migration strategy should be applied to redistribute the loads among the controllers. Hence, the condition of having dead controllers holds true whenever the residual energy falls below a certain threshold.

Different metaheuristic-based clustering techniques [21] have been introduced to minimize the network energy consumption and enhance network lifetime. For instance, the Yellow Saddle Goatfish Algorithm (YSGA) [15] is a metaheuristic-based algorithm that optimally selects the CHs, manages to intensify the network lifetime by considering an unfixed number of clusters, and clustering the nodes to nearest CHs. However, if the distance between the sensor node and its CH is larger than that between the sink and the node, then, the sensor node is not clustered. In this case, the node sends its data directly to the sink. This contradicts the concept of clustering on one hand, and causes degradation in network throughput if a failure path or link between the node and the sink exists on another hand.

Nitesh *et al.* [13] proposed a fault tolerance and energy utilization-based scheme for the large-scale network. The proposed scheme named Energy-Efficient Fault-Tolerant Clustering Algorithm for Wireless Sensor Networks (EEFCA) considers the node position from the base station, residual energy, and the number of sensor nodes in each cluster. In their scheme, each sensor node calculates the cost of joining a relay node close to the base station whenever its associated CH is dead. Accordingly, it transmits the load to the relay node that sends it to the base station. In their proposed scheme, the selection of CHs depends on sensor nodes' initial energy level, which should be based on the remaining energy and nodes' positions. In addition, their proposed scheme requires various calculations done by relay nodes to choose the best CH for communication, which leads to more energy consumption.

In our proposed scheme, each node's density or load is taken into consideration, where the root controller executes the NSA algorithm to ensure the network steady-state is achieved in the presence of percentages of dead nodes. In our proposed NSA, the load migration is done only if the added load to the current load of the receipt controller doesn't exceed a certain threshold.

3. Motivation

The albatross is one of the smart creatures in the animal world that uses the windshields to avoid exhausting its energy when flapping its wings, and as a result, will be able to travel long distances [22]. The albatross bird has inspired researchers at the Massachusetts Institute of Technology to develop a new wind and energy harvesting model by adopting the albatross dynamic soaring flying technique. The model focuses on designing energy-efficient-based wind-propelled drones and gliders to monitor remote regions for long-duration, long-range under various wind conditions [12]. However, MIT researchers have revealed that the birds tended to turn by an average angle of 60 degrees, far shallower an arc than the 180-degree half-circle that most scientists have assumed. The dynamic soaring flying technique allows the albatross bird to travel far distances in a single day, with few flaps of its wings, saving lots of effort and energy [12].

In [22], the author stated that albatross birds, when heading north, fly in approximately 60° anticlockwise loops and change to clockwise loops when heading south. From this perspective, we have adopted this flying scheme to select the CHs based on soaring between nodes with high energy levels and nodes with low energy levels of a shifting angle of 60° in the same energy level set. We claim that adopting this dynamic soaring technique ensures a balanced network energy level revealed in the network lifetime enhancement. Although the k-way spatial clustering method is used by authors [19], [33], [23]; however, selecting the cluster heads is based on randomly selecting high nodes' residual energies. The cluster head selection is based on both the nodes' residual energies and positions in our scheme.

4. System overview

4.1. System description

In this study, the system model is considered a multi-domain-based WSN. The network model can be viewed as an undirected graph G(S,E) having S sensor nodes and E links. The network model, shown in Fig. 1, consists of implementing three controllers, *i*, i = root, 1,2; one placed at the sink called root, and the two controllers are efficiently placed at specific cluster heads, selected from the cluster heads found in network that satisfy the network constraints by applying the GA. Each SDN controller manages and controls its domain and shares the network state with other controllers. In the presence of dead controller(s), the root controller optimally selects new controller(s) by applying the GA. Each controller selects CHs of its associated domain by applying the soaring scheme among the high-energy level nodes. The root controller keeps checking the overall network's steady-state to avoid the presence of any dead controller(s) or overloaded controller (s). If the root controller detects any overloaded controller(s), it executes the NSA algorithm.

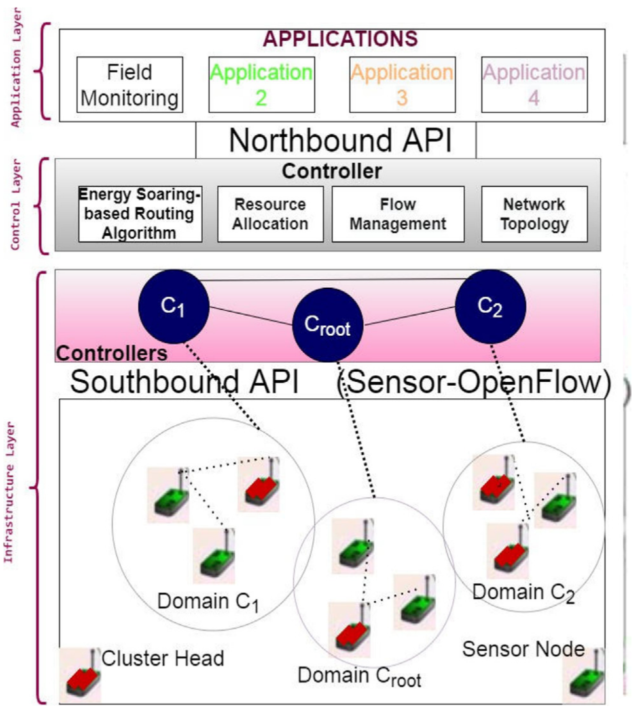

Fig. 1. The proposed network architecture

The following assumptions are considered:

• The root controller is implemented at the sink, where the base station is assumed to be, is failure-free, knows the network topology.

• Sensor nodes are randomly distributed.

• The forwarding tables are updated by the controllers; consist of alternative paths to avoid network breakage in case of failure.

• Sensor OpenFlow channel hosted out-of-band

• All sensor nodes initially have an equal energy level.

• High Energy Level Set (HES) is assumed to be in the anticlockwise direction, and the Low Energy Level Set (LES) is assumed to be in the clockwise direction.

Once the root controller finds the controllers' locations, it clusters the rest of the nodes among the network controllers using the k-mean method [24].

4.2. Problem description

In a multi-SDN controller –based WSNs, the CPP is a critical issue to be addressed for achieving the required network performance. In WSNs, sensors are equipped with limited battery that can't be recharged. Hence, the status of a node is dead whenever it experiences quick energy depletion which falls below a threshold. This early dead node occurrence degrades the overall network performance. Therefore, saving the network energy is one of the critical network performance factors to be achieved. Clustering algorithm can achieve this purpose; yet, efficiently saving the network energy is questionable. Most clustering algorithms take into account the nodes' residual energy only, where nodes' location and status are also critical factors to be considered. Besides, the network energy is directly affected by the status of the network; i.e., when a network is in steady-state, the load among the controllers is balanced and hence, no delay occurs and the network functions at its highest performance. Hence, we have focused in our proposed algorithm to maximize network energy saving and achieve the network steady-state in the presence of percentage of dead nodes.

4.3. The proposed ESRA algorithm

The root controller applies the k-way spatial clustering technique [25] that considers the nodes' locations to find the disjoint clusters, then executes the energy soaring scheme to select the network cluster heads. The proposed ESRA algorithm adopts the natural flying skill of the albatross bird that depends on the windshields to travel long distances with few flips of its wings and consequently avoids early exhaustion of its energy. Researchers have declared that the birds turn by an average angle of 60 degrees, contradicting the claim that some scientists have assumed turning half-circle [12]. Hence, we have followed the researchers' declaration and applied 60 degrees shifting angle.

The proposed scheme ensures network resiliency by avoiding dead or overloaded cluster heads to solve the CPP. It also achieves Pareto-optimal solutions by applying the non-dominated sorting genetic algorithm (NSGA-II) [26] on the solutions obtained from the GA. Figure 2 shows the flowchart of the ESRA algorithm.

The following steps describe the ESRA algorithm.

Step 1: The nodes are sorted based on their energy levels. Nodes with energy levels higher than a threshold (denoted as Ermin) are added to a set called high set and denoted by setH{}.

Step 2: After finding setH{}, nodes again are placed into two sets: the LES that consists of nodes having energy level above Ermin and less than half the initial energy level, and the HES that consists of the nodes with energy level greater than half the initial energy level.

Step 3: The mean for both sets HES and LES denoted by (X_{mean}, Y_{mean}) is calculated by looping over every node in both sets to find the corresponding image nodes denoted by (X_{image}, Y_{image}), which is achieved by shifting 60 degrees anticlockwise for HES and clockwise for LES. Equations (1), (2), (3) and (4) illustrate the coordinates of the image node after the shifting [27] mechanism for HES and LES, respectively.

Step 4: After finding the mean of both sets, the distance value between each image node and the mean node is calculated, the minimum one is chosen. Then, the closest node to the image node becomes the cluster head.

Step 5: If the cluster head node belongs to the HES set, the algorithm will start again from LES for the next cluster head selection and vice versa.

Step 6: Apply the network stability algorithm (NSA).

By adopting this dynamic soaring technique among different levels of nodes' energies eligible to be cluster heads, we claim that the overall network lifetime can be improved.

$$X_{Himage} = \cos(60)*(X-X_{Hmean})-\sin(60)*(Y-Y_{Hmean}) + X_{Hmean} \quad (1)$$

$$Y_{Himage} = \sin(60)*(X-X_{Hmean}) + \cos(60)*(Y-Y_{Hmean}) + Y_{Hmean} \quad (2)$$

$$X_{Limage} = \cos(-60)*(X-X_{Lmean})-\sin(-60)*(Y-Y_{Lmean}) + X_{Lmean} \quad (3)$$

$$Y_{Limage} = \sin(-60)*(X-X_{Lmean}) + \cos(-60)*(Y-Y_{Lmean}) + Y_{Lmean} \quad (4)$$

4.4. The proposed genetic algorithm

GA is a heuristic search and a multi-objective optimization approach inspired by Charles Darwin's theory of natural evolution. The fittest individuals are selected for reproduction to produce offspring of the next generation based on natural selection. Usually, GA consists of having two main operators, crossover and mutation; it can generate near-optimal solutions. Since finding an optimal solution to the controller placement problem is computationally NP-hard [6], we have used the GA to give near-optimal solutions for solving the CPP efficiently. In our case, the GA approach is

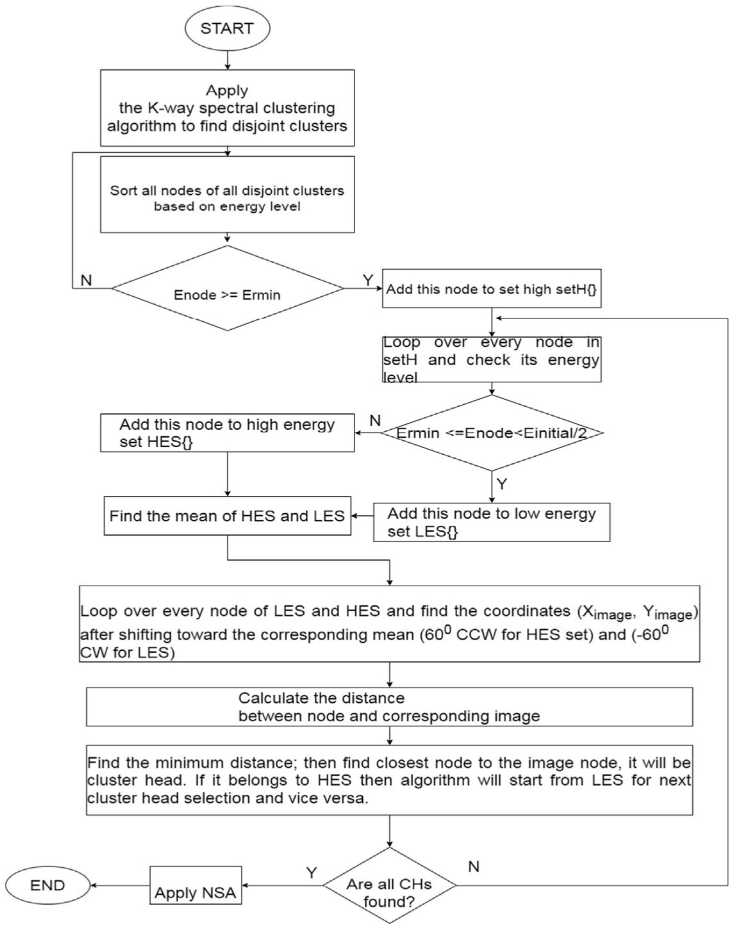
START
Apply the K-way spectral clustering algorithm to find disjoint clusters
Sort all nodes of all disjoint clusters based on energy level
Enode >= Ermin
N
Y
Add this node to set high setH{}
Loop over every node in setH and check its energy level
Ermin <=Enode<Einitial/2
N
Y
Add this node to high energy set HES{}
Add this node to low energy set LES{}
Find the mean of HES and LES
Loop over every node of LES and HES and find the coordinates (Ximage, Yimage) after shifting toward the corresponding mean (60⁰ CCW for HES set) and (-60⁰ CW for LES)
Calculate the distance between node and corresponding image
Find the minimum distance; then find closest node to the image node, it will be cluster head. If it belongs to HES then algorithm will start from LES for next cluster head selection and vice versa.
Are all CHs found?
Y
N
Apply NSA
END

Fig. 2. ESRA flowchart

adopted to simultaneously optimize the following: the network connectivity, the network load balance, the network latency, and the network energy saving. The algorithm stopping condition is achieved by either a required number of populations being reached or an optimal fitness level for the population. An illustration of the GA is shown in Table 1. Each controller *i* is associated with several cluster heads n_i, where n_i is a subset of N CHs in the network. A chromosome consists of a number of genes where each gene represents a cluster head. A population is a collection of K chromosomes. For each chromosome in K, *k* random cluster heads are selected, where $k \subset n_i$, and the fitness function is calculated based on the following four objectives:

- Optimizing the network connectivity: The number of flow messages implicitly reflects the strength of network connectivity. The number of flow messages arriving at controller *i* from one of its cluster heads $j \in n_i$, is denoted by f_{ji}. An important note to be mentioned is even in the presence of faulty paths or nodes, the transmission of messages is done via alternative paths to reach the destination. For this, we have considered the use of k-way spectral clustering which partitions the network into disjoint clusters. Eq. (5) shows the maximum average flow among all the K chromosomes of the randomly selected k controllers, where n_k is the number of cluster heads associated with the related controller.

$$f1 = \max_{K} \sum_{i=1}^{k} \sum_{j=1}^{n_i} \frac{f_{ji}}{n_i} \tag{5}$$

- Optimizing the network load balance: Balancing the load among controllers avoids the presence of overwhelming controller(s) which negatively affects the network delay due to the increase in the controller's response time. When using the in-band scheme, as in our case, balancing the load among controllers is an important aim to be achieved to avoid unnecessary delays. The load of a controller *i*, denoted by $Load_i$, is the sum of all the successfully flow messages issued by the controller's associated cluster heads $j \in ni$, (i.e. f_{ji}), and neighboring cluster heads $j' \in nb$ of the neighboring controller b, denoted by f_{ji}. Therefore, a message is successfully received if a node j issuing the flow is nonfaulty, denoted by ftj, as shown by Eq. (6), and the path between cluster head j and controller i exists, denoted by p_{ji}, as shown by Eq. (7). Hence, the load of controller i can be shown as given in Eq. (8).

$$p_{ji} = \begin{Bmatrix} 1000 \,|\, a\ path\ exists\ j,\ and\ a\ controller,\ i \\ 1 \,|\, otherwise \end{Bmatrix} \tag{6}$$

$$ft_j = \begin{Bmatrix} -1 \,|\, if\ cluster\ head\ j\ is\ faulty \\ 1 \,|\, otherwise \end{Bmatrix} \tag{7}$$

$$Load_i = \sum_{j=1}^{n_i} f_{ji\times} ft_j \times p_{ji+} \sum_{j'=1}^{n_b} f_{j'i\times} ft_{j'} \times p_{j'i} \tag{8}$$

Equation (9) shows the minimum load among all the K chromosomes of the randomly selected *k* controllers.

Table 1
The proposed GA pseudo-code.

```
Input: Parameter popsize, crossover probability $p_c$, mutation probability $p_m$, maximum iteration iter_max, k is number of controllers
Output: The controllers' locations
1: initialize popsize individuals;
2: check feasibility of each individual;
3: WHILE number of generations <= iter_max do
4: For i = 1 to popsize do
5: F (i) $\leftarrow \varphi$
6: Select k controllers from cluster heads, S={1,...,i} is set of cluster head ids
7: For j = 1 to k do
8: Apply the k-mean method to assign every cluster head to controller
9: end for
10: calculate the fitness value F of each chromosome given in equation 16
11: Order the population based on evaluation value;
12: Perform the Tournament selection process;
13: Apply the partially matched crossover operator
14: Apply the mutation operator
16: Update the population for the next generation;
17: end for
18: END WHILE
19: return S
```

$$f2 = \min_{K}(\sum_{i=1}^{k} \mathrm{Load_i}) \tag{9}$$

• Minimizing the network delay: The network delay, denoted by D_{Total}, is the sum of transmission delay, propagation delay, and the queuing delay of a controller in each chromosome. The transmission delay is the time taken to push all the packet's bits into the link, and is given by L/B, where L is the packet size, and B is the bandwidth. We neglected the transmission delay in our case as it is very small compared to the other delays. The propagation delay is the time needed for a packet to reach the destination, and the queuing delay represents the waiting time of a packet in the controller's buffer where each controller's buffer is modeled as an M/M/1 queuing system [28]. The service rate is denoted as μ, and $\lambda_{f_{ji}}$ is the arrival rate of requests from cluster head j to controller i. The distance between a cluster head j and a controller i is denoted by d_{ji}, and the speed of light is denoted by c. Therefore, the aim is to minimize the maximum worst-case latency, given by Eq. (10), which must be bounded to a given threshold, denoted by $T_{threshold}$ to avoid unnecessary delays.

$$f3 = D_{Total} = \max_{K} \sum_{i=1}^{k} \sum_{j=1}^{n_i} (\frac{d_{ji}}{c} + \sum_{l=1}^{f_{ji}} \frac{1}{\mu - \lambda_l})$$

$$\text{s.t. } D_{Total} <= T_{threshold} \tag{10}$$

■ Maximizing the network lifetime: The network lifetime represents the total number of alive nodes existing at the end of the simulation. A node j belonging to controller i dies when the current energy, denoted by E_j, falls below an energy threshold, denoted by $E_{threshold}$. The sensor current energy, E_j, is the difference between the initial energy, denoted by $E_{(ini)j}$ and the total consumption energy, denoted by $E_{jconsumption}$. Equations (11), (12) and (13) provide the calculation details for energy consumption for transmitting L bits, receiving L bits, and total energy consumption. The energy required for a node j to transmit L bits at a distance d_j is denoted by ET_j and given by Eq. (11), where E_{elec} is the energy consumption of node transceiver circuit for receiving or transmitting one-bit data, E_{fs} and E_{amp} are power consumption coefficients needed for power amplification in the free channel and multi-path fading channel respectively, and d_0 denotes the distance threshold to decide which radio model is used. The energy required for a node j to receive L bits is denoted by ER_j and given by Eq. (12). The total node's energy consumption is the sum of the transmission and reception energies as shown by Eq. (13).

$$ET_j = \begin{cases} E_{elec} * L + L * d_j^{\,2} * E_{fs}, d \le d_0 \\ E_{elec} * L + L * d_j^{\,4} * E_{amp}, d > d_0 \end{cases} \tag{11}$$

$$ER_j = E_{elec} * L \tag{12}$$

$$E_{jconsumption} = ET_j + ER_j \tag{13}$$

Therefore, the current energy of a node j is given by Eq. (14).

$$E_j = E_{(ini)j-} E_{jconsumption} \tag{14}$$

Equation (15) shows the maximum network lifetime among all the K chromosomes of the randomly selected k controllers by avoiding the consideration of dead cluster heads in the system.

$$f4 = \max_{K} \sum_{i=1}^{k} \sum_{j=1}^{n_i} \sum_{l=1}^{j} E_l \times p_{ji} \times ft_j \tag{15}$$

The weights values are as follows: $\omega1 = 25/48$, $\omega2 = 13/48$, $\omega3 = 7/48$, and $\omega4 = 3/48$ after applying the scalarization method [29]. The GA objective function denoted by F is given by Eq. (16):

$$F = \max(\omega_1 f1 + \omega_2 f2 + \omega_3 f3 + \omega_4 f4) \tag{16}$$

5. Network stability algorithm (NSA)

The network stability algorithm ensures smooth network functionality by avoiding overloaded nodes after load migration of any dead or overwhelmed controller(s). The network stability algorithm is shown in Table 2. The description of the NSA algorithm is given by the below three steps that ensure the network stability and reliability after the load migration.

Step 1: The root controller checks whenever the response time, denoted by $T_{i\_response}$ of the controller i is above a threshold, 2 ms [19], then the root controller executes the NSA algorithm. In the proposed scheme, we considered a percentage of dead nodes to be present in the system. Hence, a controller is overloaded if it is not dead and its response time is above a given threshold. This condition doesn't exist in [30,17]. The algorithm depends on the controllers' average response time as an input to detect the two sets, denoted as H_C for overloaded controllers and L_C for low-loaded controllers.

Step 2: If the controller(s) is alive and its average response time is above the given threshold, then it is added to the H_C set, else it is added to L_C set.

Table 2
NSA algorithm.

```
initialize controller set H_C = {} & L_C = {}, A={$T_{i\_response}$ of all controllers}
let j be the serial number of the controller, i is number of controllers
 1. For j = 1 to i do
 2. select $T_{j\_response}$ from A , s. t. defj = false
 3. if $T_{j\_response}$ > 2 ms then
 4. add j to H_C
 5. Else
 6. add j to L_C
 7. end if
 8. if defj== True then continue with 20
 9. end if
 10. while (H_C ∩ L_C is NotEmpty) do
 11. CO = $\underset{i\in H\_C}{Max}$ {f2}
 12. CH_O = $\underset{n\in CO}{Max}$ {f2}s.t $def_n$==false and flag(n)==false
  // to ensure receipt controller does not get overwhelmed
 13. CL = $\underset{i\in L\_C}{Min}$ {f2ofcontroller + f2ofCH_O <= 2600}
 14. If CL==NULL then Flag(CH_O) = true
 15. Go to step 12
 16. else
 17. add < CO;CH_O;CL > to P
 18. Remove CO from H_C
 19. Remove CL from L_C
  // setting the load denoted by Load in eq.(8).
  //distributing load of dead controller
 20. set Ld={$Load_d$, d is the dead controller},
     Flag(j) = false, j∈{1,. . .,$n_d$}
 21. o = Max{Ld};
 22. l = Min{CL}
 23. While Ld Is NotEmpty do
 24. For j = 1 to $n_d$ do
 25. If (Load $_l$+Load (flagged(j) = false)<=2600 then
 26. Add < j,l > to P
 27. Update $Load_l$
 28. Remove j from Ld
 29. else
 30. Flag(j)==True
 31. Continue with 23
 32. end if
 33. end for
 35. end while
 36. return P
```

Step 3: If the controller(s) is dead, the root controller distributes the load of the dead ones among the alive and not overloaded controller. The distribution of the load is done only if the total load of the receipt controller doesn't exceed a load threshold.

6. Performance evaluation

Simulations were carried out using MATLAB 2019b to evaluate the performance of the proposed ESRA algorithm. Different percentages of dead nodes were considered during simulations. Simulation parameters are shown in Table 3. We have considered the simulation time of 300 s, ran our scheme 50 times for each percentage of dead nodes, and took the average to achieve a 95% confidence interval. Three SDN controllers are used for 500 randomly deployed sensor nodes in a 200 m × 200 m field area. We considered that the packet arrival rate starts with 200 packets/s and increases 100 packets every 10 sec to reach 3000 packets/s. The following performance metrics were considered: network lifetime, throughput, energy consumption, and network latency. Comparisons were carried out with the following three energy-aware algorithms found in the literature. The first, the EEFCA protocol [13], is a distributed cluster-based algorithm aiming at providing fault tolerance in the presence of CHs failures and whose cost function is based on nodes' energy and locations. Each sensor node does the cost function calculation to choose its cluster head among relay nodes having the highest cost value. The second algorithm, GCEEC [14], is an energy-based routing protocol where the centroid position is considered for CHs election and gateway nodes are selected from CHs to release the load from the overwhelming CHs then forward the data to the base station. The CH calculates the average cluster's energy and the weight for the gateway nodes adjacent to the neighboring CH in each cluster. The one with the highest weight is selected as a gateway node for the respective cluster. The third algorithm is YSGA [15], a metaheuristic-based algorithm aiming to enhance network lifetime by reducing network energy consumption.

Table 3
Simulation Parameters.

Parameter	Value
Field area	200×200 m^2
Sensor nodes	500
Partially crossover probability, pc	0.8
Mutation probability, pm	0.2
Population size, popsize	50
Stopping iteration, $S_{iteration}$	5000
Residual Energy	2 Joules
E_{rmin}	0.05 Joules
Packet arrival rate	200packets/s-3000packets/s
Defective percentage	[0,1,5,10,20,50]%
E_{DA}	5 nJ/bit
ε_{fs}	10 pJ/bit/m2
ε_{mp}	0.0013 pJ/bit/m4

Performance analyses were carried out for the following metrics:

6.1. Worst-case latency

We have recorded the worst-case latency under the ESRA algorithm with various dead nodes' percentages and compare it with YSGA [15], GCEEC [14], and EEFCA [13]. Fig. 3 presents the latency results for the above four schemes. As it is seen, the ESRA algorithm outperforms YSGA, GCEEC, and EEFCA algorithms by 10%, 15%, and 20%, respectively. In EEFCA and GCEEC algorithms, calculations are done by sensor nodes and cluster heads, respectively, that add more delay to the network. In the YSGA algorithm, the number of CHs is not fixed. Instead, the number of CHs is dynamically changed to build the best network configuration in every round. Once the optimal cluster heads are selected, each sensor node is incorporated into the nearest cluster head. Nevertheless, if the distance from the sensor node to the BS is shorter than the distance to the CH, then the sensor node is not clustered, so the information of this node is transmitted directly to the base station. As a whole, this adds more delays, especially in the presence of more than one faulty node. However, in our proposed algorithm, each controller runs the GA to select a CH instead of the dead one and every node is clustered. This ensures the packet transmission via alternative paths in the presence of dead nodes, and hence the overall latency

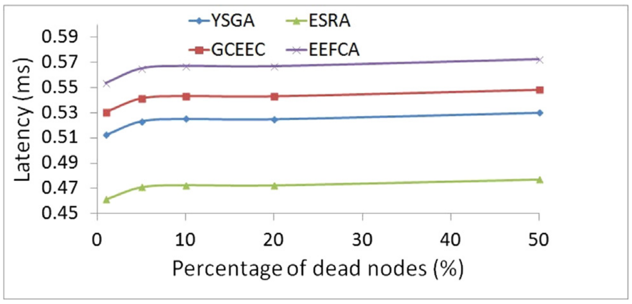

Fig. 3. Latency analysis

is decreased, as seen in Fig. 3. Another thing is the load is well distributed in our proposed algorithm, where the system ensures the steady-state of the network in the presence of overwhelmed nodes which has a positive impact on overall network latency. Load balance is missing in YSGA, EEFCA, and GCEEC algorithms, which increased network latency.

6.2. Network lifetime

The network lifetime is a critical network performance metric, especially for wireless sensor networks. Having different dead nodes' percentages, the percentage of alive nodes at the end of simulation reflects the network lifetime. Balancing the energy consumption among the sensor nodes in the system ensures extended the network functionality. Hence, more alive nodes exist in the system. Fig. 4 shows the percentage of alive nodes for the proposed ESRA algorithm against YSGA, GCEEC, and EEFCA algorithms. Energy depletion occurs faster in EEFCA and GCEEC algorithms due to excessive calculations done on behalf of sensor nodes that resulted in an early occurrence of dead nodes. In YSGA, the number of clusters is dynamic that causes a less consumption of energy than having a fixed number of cluster heads. On the contrary, the algorithm takes into account the nodes' residual energies when selecting the cluster heads without considering the nodes' location to the sink and the nodes' load. Hence, in the presence of overloaded nodes, it is preferable to consider the nodes' load and location along with the residual energy in the cluster head selection algorithm. This explains why the YSGA algorithm fails to achieve the network steady-state in the presence of dead nodes. In contrast, the proposed ESRA algorithm considers the nodes' residual energies, distance to the sink, and CHs' load which balances the network energy consumption and enhances network lifetime. Hence, the ESRA algorithm outperforms YSGA, GCEEC, and EEFCA algorithms by 15%, 20%, and 25% respectively, as shown in Fig. 4.

6.3. Percentages of successfully received packets

Another critical network performance metric is the network throughput, referred to as the percentage of successfully received packets. Fig. 5 shows the percentage of successful packets received under various dead nodes' percentages. The percentage of successful packets received is inversely affected by the increase in the percentage of dead nodes in the system. Since the load of each CH is considered in the proposed ESRA algorithm, the network energy is well balanced and more alive nodes exist. Also, ensuring alternative paths for packet transmission in the presence of dead nodes has positively impacted the overall network throughput. That explains why the proposed ESRA algorithm is superior to YSGA, GCEEC, and EEFCA algorithms. In YSGA, GCEEC, and EEFCA algorithms, the load distribution of dead nodes is not considered. Thus, the percentage of received packets decreases quickly as the percentage of dead nodes increases. Hence, the ESRA algorithm outperforms YSGA, GCEEC, and EEFCA algorithms by 15%, 20%, and 25%, respectively.

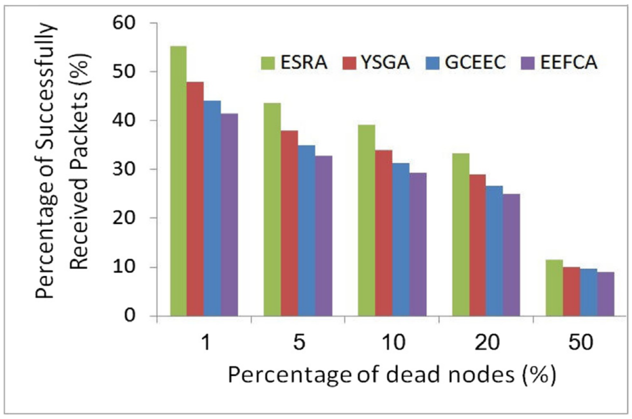

Fig. 5. Percentage of successfully received packets analysis

6.4. Energy consumption

Energy is a critical factor in wireless sensor networks where sensors are equipped with limited power in which packet transmission consumes most of the sensors' energy. The cluster-based network proves to save overall network energy; however, an efficient selection of cluster heads directly affects the network's energy consumption. The ESRA algorithm is a cluster-based algorithm where the network is clustered into several disjoint clusters. In YSGA, the number of clusters is not fixed, and the CHs are selected based on nodes' residual energies, consumption energies, and distance to the sink in each round. However, the nodes' densities are not considered; thus, balancing the energy consumption among sensor nodes is not achieved when the load increases and the percentage of dead nodes increases. A cluster head is selected in our proposed ESRA algorithm by applying the soaring technique described in the ESRA flowchart, which balances the energy consumption among the sensor nodes. Fig. 6 shows the energy consumption of the ESRA algorithm against YSGA, GCEEC, and EEFCA algorithms when various percentages of dead nodes are present.

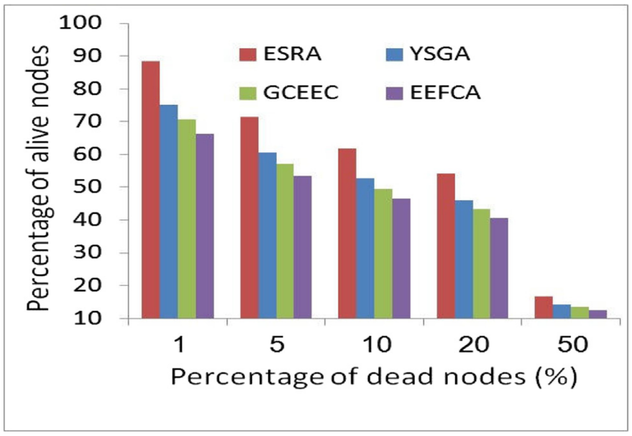

Fig. 4. Network lifetime analysis

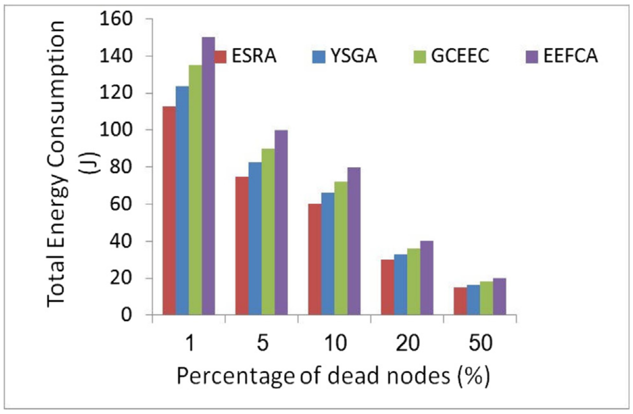

Fig. 6. Total energy consumption analysis

Table 4
Latency and Execution Time Comparison.

N_CO	Topology	Latency (ms)			Execution Time (s)		
		ESRA	Ref. [30]	Ref. [18]	ESRA	Ref. [30]	Ref. [18]
3	JANET	12	14	–	11	11	–
	GEANT	15.8	–	16.8	10	–	10
4	JANET	10	14.1		12	12	–
	GEANT	14	–	16.5	13	–	13
5	JANET	10.2	15	–	22	21	–
	GEANT	14.3	–	16.6	19	–	18
6	JANET	11	17		35	34	–
	GEANT	14.4	–	17	34	–	33
7	JANET	13	18	–	46	45	–
	GEANT	14.5	–	17.5	45	–	43
8	JANET	13.8	18.5		57	55	–
	GEANT	14.7	–	18	56	–	54
9	JANET	14	19	–	66	63	–
	GEANT	15.1	–	18.5	67	–	64

Note that when the percentage of dead nodes increases, more dead nodes are present, and as a consequence, the total network energy consumption decreases. On the other hand, due to excessive calculation done by sensor nodes in both GCEEC and EEFCA algorithms, the network energy consumption is higher than that of the ESRA algorithm. Results show that the ESRA algorithm outperforms YSGA, GCEECA, and EEFCA algorithms by 10%, 20%, and 25%, respectively. Simulation results prove that the ESRA algorithm manifests its superiority in terms of energy efficiency.

6.5. Applying ESRA algorithm on real internet topologies

We have applied our proposed ESRA algorithm on real datasets [31]. We run our scheme to choose the best number of controllers starting at one controller to seven controllers for the JANET network and GEANT network. First, we explain some of the graph metrics [23]. For instance, in an unweighted graph, the distance between two connected nodes represents the number of edges counted in the shortest path. Then the diameter of the network is defined as the maximum distance, i.e., the maximum number of edges in the shortest path of any two connected nodes. We then specify the latency constraint of a network as half the diameter [18]. For Janet and GEANT, we specified the latency constraint as 14 ms and 10 ms, respectively. Table 4 illustrates the latency and execution time comparison results when various controllers (denoted as N_CO) are used. It is clear that the proposed ESRA algorithm outperforms the two schemes [30,18] in terms of latency by 26% and 15%, respectively. It is worth mentioning that since the ESRA algorithm is a cluster-based routing algorithm, the overall latency decreases with the decrease in distance between nodes. Another fact is that the ESRA algorithm dynamically chooses cluster heads by soaring among the nodes preventing quick energy depletion, and hence enhancing the network lifetime. Conversely, the average execution time of the ESRA algorithm exceeds that of [30,18] by almost 5% and 7%, respectively, as shown in the last column of Table 4.

7. Conclusion

In this paper, we address the controller placement problem in a multi-controller SDN-based WSNs to optimize the network energy saving and enhance the network lifetime. Our proposed ESRA algorithm is energy-efficient that selects the CHs by adopting the flying technique of the albatross bird. Moreover, an efficient network performance evaluation is carried out by considering a percentage of dead nodes to be present in the network that usually exists in a real WSN. Hence, ESRA algorithm achieved the network stability and enhanced the network lifetime in the presence of dead nodes. The algorithm effectively balances the network energy consumption by soaring among the high and low energy level nodes to select the network cluster heads. The cluster head selection is important since the controllers are chosen among these cluster heads by applying the GA that considers the nodes' residual energies, distance to the sink, and the load of CHs. In YSGA [15], although the number of clusters is not fixed; however, choosing CHs is only based on nodes' residual energies and distance to the sink and does not consider the load of the CHs. In our proposed ESRA algorithm, the root controller only chooses the controllers instead of the dead ones, and the CH selection is done on behalf of the domain's controller. The proposed ESRA algorithm outperforms the YSGA [15] algorithm, GCEEC [14], and EEFCA [13] in terms of network lifetime and percentage of successfully received packets by 15%, 20%, and 25% respectively, in terms of latency by 10%, 15%, and 20%, respectively, and in terms of energy consumption by 10%, 20%, and 25%, respectively. The proposed ESRA algorithm also showed latency improvement over [30,18] when applied over Janet and GEANT networks, respectively. The obtained network performance improvements added a marginal increase in the proposed scheme's execution time due to involving more calculations for choosing the network CHs. For future work, we are planning to include the energy harvesting concept in our model where the sensor nodes are designed to harvest energy from the environment during the daytime and only utilize their battery during the night.

Conflict of Interest

The authors declare that there is no conflict of interest regarding the publication of this article.

References

[1] Essa A, Al-Dubai AY, Romdhani I, Eshaftri MA, "A new dynamic weight-based energy efficient algorithm for sensor networks," in Smart Grid Inspired Future Technologies: First International Conference, SmartGIFT, Liverpool, UK, 2016.
[2] Kong L, Xiang Q, Liu X, Liu X-Y, Gao X, Chen G, et al. ICP: instantaneous clustering protocol for wireless sensor networks. Comput Netw 2016;101:144–57.
[3] Yuea J, Zhang W, Xiao W, Tang D, Tang J, "Energy efficient and balanced cluster-based data aggregation algorithm efficient and balanced cluster-based data aggregation algorithm," Procedia Engineering, vol. 29, p. Procedia Engineering, 2012.
[4] Yoon S-K, Khalib ZIA, Yaakob N, Amir A, Soh PJ, Abdul Khalib ZI, et al. Controller placement algorithms in software defined network – a review of trends and challenges. MATEC Web Conf 2017;140:01014. doi: https://doi.org/10.1051/matecconf/201714001014.
[5] Hu Y, Wang W, Gong X, Que X, Cheng S. On reliability-optimized controller placement for Software-Defined Networks. China Commun. 2014;11(2):38–54.

[6] Heller B, Sherwood R, McKeown N, "The Controller Placement Problem," in Proceedings of the first workshop on Hot topics in software defined networks, Finland, 2012.
[7] Singh AK, Srivastava S. A survey and classification of controller placement problem in SDN: a survey and classification of controller placement problem in SDN. Int J Network Mgmt 2018;28(3):e2018. doi: https://doi.org/10.1002/nem.v28.310.1002/nem.2018.
[8] Yao H, Qiu C, Zhao C, Shi L. A multicontroller load balancing approach in software-defined wireless networks. Int J Distrib Sens Netw 2015;2015:1–8.
[9] Kobo HI, Abu-Mahfouz AM, Hancke GP. A survey on software-defined wireless sensor networks: challenges and design requirements. IEEE Access 2017;5:1872–99.
[10] Hu T, Guo Z, Yi P, Baker T, Lan J. Multi-controller based software-defined networking: a survey. IEEE Access 2018;6:15980–96.
[11] Jalili A, Keshtgari M, Akbari R. A new set covering controller placement problem model for large scale SDNs. J Information Systems Telecommun 2018;6:25–32.
[12] Chu J, "Engineers identify key to albatross' marathon flight," Massachusetts Institute of Technology MIT News Office, 10 October 2017. [Online]. Available: http://news.mit.edu/2017/engineers-identify-key-albatross-marathon-flight-1011.
[13] Nitesh K, Azharuddin M, Jana PK, "Energy efficient fault-tolerant clustering algorithm for wireless sensor networks," in 2015 International Conference on Green Computing and Internet of Things (ICGCIoT), Noida, India, 2015.
[14] Qureshi K, Bashir MU, Lloretq J, Leon A, Optimized cluster-based dynamic energy-aware routing protocol for wireless sensor networks in agriculture precision, J Sensors, vol. 2020, p. 19 pages, 2020.
[15] Rodríguez A, Del-Valle-Soto C, Velázquez R. Energy-efficient clustering routing protocol for wireless sensor networks based on yellow saddle goatfish algorithm. Mathematics 2020;1515(8):1–17.
[16] Hu Y, Luo T, Beaulieu NC, Deng C. The energy-aware controller placement problem in software defined networks. IEEE Commun Lett 2017;21(4):741–4.
[17] Buzura S, Iancu B, Dadarlat V, Peculea A, Cebuc E. Optimizations for energy efficiency in software-defined wireless sensor networks. Sensors 2020;4779 (20):1–23.
[18] Killi BPR, Rao SV. Capacitated next controller placement in software defined networks. IEEE Trans Netw Serv Manage 2017;14(3):514–27.
[19] Samarji N, Salamah M, A Fault Tolerance Metaheuristic-Based Scheme for Controller Placement Problem in Wireless Sensor Networks, Int J Commun Systems, vol. 34, no. 4, 2021.
[20] Yu J, Wang Y, Pei K, Zhang S, Li J, "A Load Balancing Mechanism for Multiple SDN Controllers based on Load Informing Strategy," in 18th Asia-Pacific Network Operations and Management Symposium (APNOMS), Japan, 2016.
[21] Singh SP, Sharma SC. Genetic-Algorithm-Based Energy-Efficient Clustering (GAEEC) for homogenous wireless sensor networks. IETE J Res 2018;64 (5):648–59.
[22] Bradshaw D, in Vertebrate Ecophysiology: An Introduction to its Principles and Applications, Cambridge University Press, 2003.
[23] Schutz G, A k-cover model for reliability-aware controller placement in software-defined networks, in Computational Science-ICCS 2019, Springer, 2019, pp. 604-613.
[24] Sasikumar P, Khara S, K-Means Clustering in Wireless Sensor Networks, in International Conference on Computational Intelligence and Communication Networks, 2012.
[25] Jorio A, El Fkihi S, Elbhiri B, Aboutajdine D, "A New Clustering Algorithm in WSN Based on Spectral Clustering and Residual Energy," in Seventh International Conference on Sensor Technologies and Applications, SENSORCOMM 2013, 2013.
[26] Lin S, "NGPM – A NSGA-II Program in Matlab v1.4," MATLAB Central File Exchange, 16 July 2011. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/31166-ngpm-a-nsga-ii-program-in-matlab-v1-4.
[27] Vince J, "Matrix Transforms," in Foundation Mathematics for Computer Science, Springer, 2015, pp. 193-194.
[28] Xiong B, Peng X, Zhao J. A concise queuing model for controller performance in software-defined networks. J Comput 2016;11(3):232–7.
[29] Roughan M, Willinger W, "Internet Topology Research Redux," 2013. [Online]. Available: http://sigcomm.org/education/ebook/SIGCOMMeBook2013v1_chapter1.pdf.
[30] Mohanty S, Priyadarshini P, Sahoo S, Sahoo B, Sethi S, "Metaheuristic Techniques for Controller Placement in Software-Defined Networks," in TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON), 2019.
[31] A. University, "The Internet Topology Zoo," University of Adelaide, 2012. [Online]. Available: http://www.topology-zoo.org/dataset.html. [Accessed 2012].

N. Samarji was born in North Lebanon in 1982. She received the B.S. degree in Computer Science from Beirut Arab University, Beirut, in 2004, the M.S. and Ph.D. degrees in Computer Engineering from Eastern Mediterranean University, North Cyprus, in 2015 and 2021. Dr. Samarji is the author of 5 journal papers, and 2 conference papers. Her research interests include Wireless Sensor Networks, Software-Defined Networking, Optimization Algorithms, Visible Light Communication (VLC), Simultaneous Wireless Information and Power Transfer (SWIPT), Hybrid channel, Queuing Theory, Performance Evaluation, and Wireless Personal Communications Systems.

M. Salamah was born in Jordan in 1965. He received the B.S., M.S., and Ph.D. degrees in electrical and electronics engineering from the Middle East Technical University, Ankara, Turkey, in 1996. From 1988 to 1994, he was a Research Assistant in Electrical and Electronics Engineering Department, METU, Ankara, Turkey. From 2010 to 2015, he was Chairman of Computer Engineering Department and Rector's Advisor for Middle East Countries. Since 1996, has been an Associate Professor with the Computer Engineering Department, Eastern Mediterranean University, North Cyprus. He is the author of more than 25 journal papers, 10 reports, 60 conference papers, and two books. His research interests include Computer networks, Wireless Communication Systems, Software-Defined Networking, Queuing Theory and Performance Evaluation, Simulation, Parallel Processing, Computer Architecture, Microprocessor Systems, Hardware-oriented Algorithms, Visible Light Communication, and Wireless Personal Communications Systems. Prof. Salamah was assigned as the Chairman of the 6th Symposium on Computer Networks and Co-Chair of the 10th Turkish Symposium on Artificial Intelligence and Neural Networks that were jointly held on June 2001 in Gazimagosa – KKTC – Turkey.