Engineering Science and Technology, an International Journal xxx (xxxx) xxx

HOSTED BY

Contents lists available at ScienceDirect

Engineering Science and Technology, an International Journal

journal homepage: www.elsevier.com/locate/jestch

Full Length Article

Chaotic salp swarm algorithm for SDN multi-controller networks

Abdelhamied A. Ateya [a,b,*], Ammar Muthanna [b,c], Anastasia Vybornova [b], Abeer D. Algarni [d], Abdelrahman Abuarqoub [e], Y. Koucheryavy [f], Andrey Koucheryavy [b]

[a] Electronics and Communications Engineering, Zagazig University, Zagazig, Egypt
[b] St. Petersburg State University of Telecommunication, 22 Prospekt Bolshevikov, St. Petersburg, Russia
[c] Peoples' Friendship University of Russia (RUDN University), 6 Miklukho-Maklaya St, Moscow, Russia
[d] College of Computer and Information Sciences, Princess Nourah Bint Abdulrahman University, Riyadh, Saudi Arabia
[e] Faculty of Information Technology, Middle East University, Amman, Jordan
[f] Tampere University of Technology, Tampere, Finland

ARTICLE INFO

Article history:
Received 12 November 2018
Revised 24 December 2018
Accepted 24 December 2018
Available online xxxx

Keywords:
Controller placement
Latency
Optimization algorithm
SDN
Swarm
Utilization

ABSTRACT

Software-defined networking (SDN) is a novel network paradigm that enables flexible management for networks. However, with the increase in network capacity, a single controller of SDN has many limitations on both performance and scalability. Distributed multi-controller deployment is a promising method to satisfy fault tolerant and scalability. There are still open research issues related to controllers placement, and the optimal number of deployed controllers. In this paper, a dynamic optimization algorithm that is based on the Salp Swarm Optimization Algorithm (SSOA) is developed with the introduction of chaotic maps for enhancing the optimizer's performance. The algorithm dynamically evaluates the optimum number of controllers and the optimal connections between switches and controllers in large scale SDN networks. In order to evaluate the proposed algorithm, several experiments were conducted and implemented in various scenarios. Moreover, the algorithm was compared to the linear and meta-heuristic algorithms. Simulation results show that the proposed algorithm outperforms meta-heuristic algorithms and a game theory based algorithm in terms of execution time and reliability.

1. Introduction

Software Defined Networking (SDN) is a recent communication paradigm introduced for cost effective dynamic networks. The main idea behind SDN is the physical separation of the control plane and the forwarding plane through programmable software controllers, which enables dynamic configuration and control of the whole network [1]. This process differs completely from the traditional networks, in which the data plane is responsible for the whole process of data forwarding and thus both planes are completely integrated in the same device [2]. Data plane includes all forwarding devices that are responsible for traffic forwarding through the network, while the control tire contains all devices used for making traffic handling decisions. SDN approach covers the interaction and collaboration between the control and data planes, as the control plane controls and manages all forwarding devices in the data plane [3]. Control plane is responsible for setting configuration parameters and deciding the forwarding roles of all forwarding devices that perform traffic forwarding based on the received instructions. A set of intelligent controllers can be deployed in the control plane to act as the brain of the SDN network [4]. The command signals between the control and data planes are accomplished by means of an appropriate standard interface protocol (e.g. ForCES and OpenFlow) [5,6].

With the recent developments and advances in the industry of electronics and sensory manufacturing, the number of wireless devices is dramatically increasing. This puts high constraints on the design and development of the future systems (e.g. 5G, IoT and Tactile Internet) that cover the connectivity among these devices [7]. These design challenges include traffic volume, connectivity and reliability [8]. By 2020, it is expected that the data traffic will be 200 times higher than that in 2010 and by 2030 it will be 20,000 times higher than the traffic in 2010 [9]. Another important issue is the high requirements and demands required from the future systems [10]. The fifth generation of cellular system (5G) is expected to achieve an increased data rate and user demands up to 1000 fold (i.e. data rate of up to ten Gbps) with ultra

* Corresponding author at: 22 Prospekt Bolshevikov, St. Petersburg, Russia.
E-mail address: a_ashraf@zu.edu.eg (A.A. Ateya).
☆ The work has been prepared with the support of the "RUDN University Program 5-100".
Peer review under responsibility of Karabuk University.

https://doi.org/10.1016/j.jestch.2018.12.015
2215-0986/

low latency, high reliability, high mobility, high throughput and high connectivity [11,12].

This ultra high amount of traffic can't be handled by current wireless solutions and thus, new technologies and paradigms should be involved. In order to achieve the requirements and user demands and handle the dramatic number of data traffic, 5G and next generation systems will deploy SDN and other technologies such as Mobile Edge Computing (MEC) and Network Function Virtualization (NFV) [13,14].

SDN network achieves various benefits through the decoupling of control plane and data plane. These benefits include [15,16]:

1. Simplicity of network management,
2. Hardware simplicity,
3. Higher network flexibility,
4. Reduction of round trip latency,
5. High reliability,
6. The ability of network innovations, and
7. Increasing over all system efficiency in terms of utilization [17].

Physical SDN controller can practically handle limited number of flow requests at a given time. Thus, the controller can only serve for limited number of switches and when the traffic increases above the upper limit that a controller can handle, there should be another controller to support. This indicates that the single centralized SDN controller may be prober, only for small scale networks [18].

The most common existing SDN interface is the OpenFlow, which is practically developed for single centralized controller [19]. OpenFlow protocol achieves poor efficiency, in terms of scalability, for large scale networks [20]. However, this problem can be improved by separating OpenFlow network into several fields, with one controller for each field. This separation and involvement of multiple controllers achieve the load balancing and scalability benefits to the SDN network [21].

Consequently, for large scale networks, employing single controller to manage all network switches isn't efficient and thus, the efficient solution is to employ multiple controllers. Fig. 1 illustrates the basic architecture of the SDN multi-controller networks [22]. SDN networks, either single or multiple controllers, consist of three main layers; the data plane layer, the control plane layer and the application layer [1,23].

In SDN multi-controller networks, the number of deployed controllers depends on the network scale and the amount of traffic. The main issues associated with SDN multi-controller networks are the number of deployed controllers and the allocation of deployed controllers to distributed switches, which is known as the controller placement problem [24]. The controller placement problem seeks to find the best places of the SDN controllers, in a way that achieves various objectives that include latency reduction, energy efficiency, load balancing and reliability enhancement [25]. The capital and operational expenditures (CAPEX and OPEX) of the network are mainly affected by the number of deployed controllers [26]. Thus, deploying optimal number of controllers and estimating their best allocations are a great demand for SDN multi-controller networks. This number and allocations should be dynamic processes that follow the dynamic changes of network size and load.

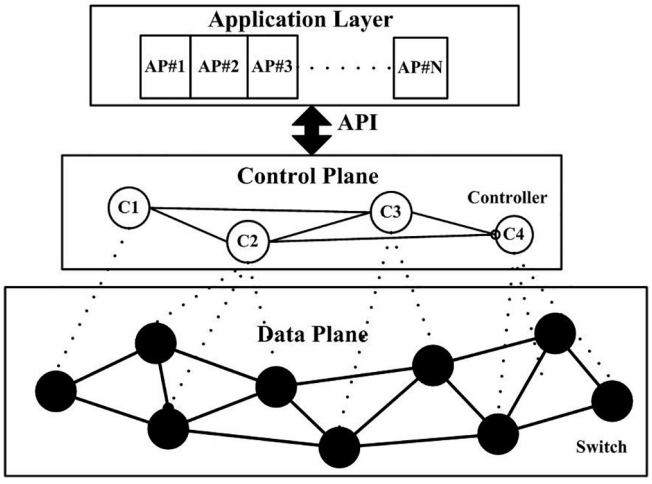

Fig. 1. General structure of SDN multi-controller networks.

For static networks, the controller placement problem can be easily solved at the start of the network operation; since the network load is static and the optimum number of controllers can be calculated readily. Dynamic changing in network traffics results in dynamic changing in the number of deployed controllers. When the network load increases, new controllers should be activated; while the decrease of network load should leads to deactivating a number of deployed controllers. Frequent changing in network load results in frequent switching of deployed controllers between on and off modes. Sleep mode may be deployed as an alternative mode to switching off [27]. The decision of mode of operation, for each controller to achieve the best system performance, may be introduced by a prober optimizer.

In this work, a dynamic optimization algorithm is developed to get the optimal number of controllers and the best allocations of switches with the available controllers for large scale SDN enabled networks. The algorithm is based on the Salp Swarm Optimization Algorithm (SSOA) with the introduction of chaotic maps for enhancing optimizer performance. In (Section II), a literature of the related works is discussed. Moreover, the novelty of the proposed work, compared to existing methods, is illustrated. In (Sections III and IV), the proposed work is introduced; at first the mathematical model of the system is presented, then the problem is defined and modeled and finally, the optimization algorithm for solving the controller placement and allocation problem is presented. In (Section V), the proposed algorithm is implemented for different networks with various scales and the performance is measured. Different simulation cases are considered to measure the effect of parameters variation on the network performance. Finally, the proposed algorithm is compared with other metaheuristic algorithms and deterministic algorithms.

2. Background and relatedwork

The controller placement and allocation problem has a great impact on the performance of the SDN enabled networks. This attracts a lot of researchers to this field and many studies have been conducted, to solve the problem and get the optimum solution. Many literature works have been conducted to review and categorize the developed methods considering the problem of controller placement and estimating the optimal number of controllers; including the main objective of the work and the considered mathematical methods. The recent review articles consider this work and survey the recent developed algorithms can be founded in [24,25].

In SDN multi-controller networks, the number of controllers and controller allocation problems are considered to be a non-deterministic polynomial (NP) - hard problem [28]. For large scale networks, with the increase of network load, the problem become harder and the solution become more difficult. The developed methods and existing solutions, for the problem of controller allocation and the problem of optimal number of controllers, can be classified based on the objectives as illustrated in Fig. 2 [24,25].

Please cite this article as: A. A. Ateya, A. Muthanna, A. Vybornova et al., Chaotic salp swarm algorithm for SDN multi-controller networks, Engineering Science and Technology, an International Journal, https://doi.org/10.1016/j.jestch.2018.12.015

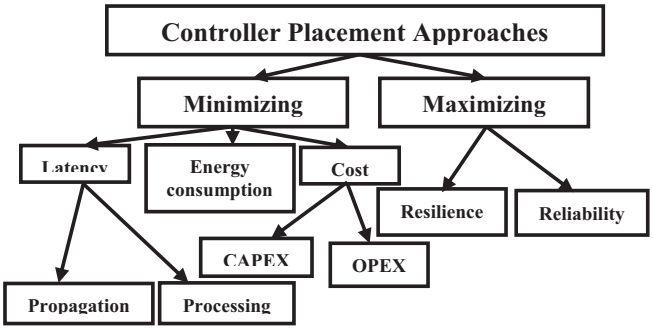

Fig. 2. Classifications of controller placement approaches.

The existing methods consider either minimizing or maximizing one or more objectives.

The proposed work considers minimizing the network latency and the deployment cost and thus, it can be located in two main categories, from the introduced categories in Fig. 2; Minimizing Latency and Minimizing Cost. The proposed work merges both categories in a single approach by considering both objectives; latency and cost, and weighting each considered objective. To this end, we consider the recent related works that consider minimizing the latency and the cost. Literatures and related approaches about the other categories can be found in [24,25].

2.1. Minimizing deployment cost

The deployment cost of the network elements includes both CAPEX and OPEX. Controller placement and number of deployed controllers mainly affect the deployment cost, thus methods have been conducted to solve the problem of controller placement and get the best number of deployed controllers that minimize the deployed cost of the SDN enabled networks [29].

Main methods, in this context, can be classified into two main approaches; static methods and dynamic methods. In both approaches, the optimal number of controllers, optimal types of controllers and optimum connections of switches and controllers are investigated for minimum deployment cost [30]. This can be happened statically (i.e. static approaches) or dynamically at real time (i.e. dynamic approaches).

In [31], a non-zero-sum game algorithm is introduced to get the optimal number of controllers and optimal connections of switches and controllers, for minimizing the deployed cost of SDN multi-controller networks. The algorithm is a distributed dynamic technique that can be implemented on each controller as an optimization engine and tracks the network load changes in real time. The algorithm is a low complexity and topology independent solution, which represent the main advantages of the work. The algorithm is simulated and tested for a random network and results indicate that, the algorithm reduces the deployment cost.

In [32], a cost aware controller placement algorithm is developed to get the optimal number of controllers and optimal placement. Moreover, the optimal types of deployed controllers are derived by the algorithm to minimize the deployment cost. A mathematical model is introduced to define the problem with two main considered constraints; latency of the path set up and the controller capacity. The algorithm runs with an input data includes controller capabilities, latency of flow set up and connection mediums. Simulation results validate the algorithm for small scale SDN networks; however it fails for large scale networks because the optimizer takes much time, due to large number of network elements and connections. An extension to this work is introduced in [33], to expand the problem and solve both planning and expansion problems.

2.2. Minimizing latency

The total latency between forwarding devices and SDN controllers is an important and critical issue for SDN networks. This time delay should be kept at minimum to enable fasts interaction between controllers and switches, and also to support the quality of service (QoS). There are two main times defined by the SDN controllers to complete its role; flow setup time and role installation time [34]. The time required to analyze the received packet and calculate the rule in the context of network policies is the flow setup time. This time mainly affected by the switch rate and number of connected switches to the controller [35]. The second time is the rule installation time, which represents the time required by a controller to install a new rule on a connected switch. Both times are considered to be the controller processing latency, which mainly depends on the amount of load on the controller. Another important delay that may takes place is the queuing delay which represents the time taken in the queue to the controller and this time mainly depends on the amount of connected switches and their rates [36]. Both queuing and processing time are referred as the response time of the controller. One more important delay is the propagation delay, which mainly depends on the distance between the controller and the switch [37].

In [38], authors introduce a framework for the global latency controller placement problem (GLCPP). An algorithm based on Particle Swarm Optimization (PSO) is developed to solve the problem, taking into account the latency between controllers and the capacity of each controller. In PSO algorithm, each swarm is considered to be a set of placements for different controllers and the goal of the optimizer is to get the best swarm based on the fitness function. The fitness mainly considers minimizing the distances between switches and controllers, and between neighboring controllers. The algorithm is implemented for random network, and compared with three linear programming algorithms. Simulation results indicate that the GLCPP algorithm introduces better results in terms of delay and execution time of the algorithm. The main disadvantage with the work is that it assumes the total number of deployed controllers, which may not be the optimal number especially with the dynamic change of network load.

In [34], a latency aware controller placement problem is defined and a chaotic based gray wolf optimization algorithm is introduced to solve the problem for optimum solution. The optimizer is mainly used to get the optimum number of deployed controllers and the optimal connections of switches and controllers that minimize the network latency. The algorithm is implemented for small scale random network with ten switches, and results are compared with the results of another meta-heuristic algorithm. Simulation results indicate that the algorithm achieves better performance in terms of latency. Moreover, the algorithm is compared with static allocation technique. The main conflict with this work is that the work is introduced for 5G cellular systems with SDN enabled, however the algorithm is implemented for small scale network and didn't tested for large scale networks and for real topologies.

Our proposed work, considers minimizing the network latency and the deployment cost. Both objectives are weighted to evaluate the optimal number of controllers and also the optimum allocations of switches to controllers. The novelty of our work comes from the consideration of both objectives and mainly, from the deployment of SSA that is considered to be one of the most efficient PSO based algorithm; especially in feature selection [39]. Moreover, the introduction of chaotic maps achieves higher efficiency for the optimizer. All developed related approaches consider only one objective and the recent works deploys versions of PSO based algorithms that has efficiency less than the considered SSA as introduced in Section V. The proposed algorithm is applicable and can be implemented to any SDN-multi controller network.

Please cite this article as: A. A. Ateya, A. Muthanna, A. Vybornova et al., Chaotic salp swarm algorithm for SDN multi-controller networks, Engineering Science and Technology, an International Journal, https://doi.org/10.1016/j.jestch.2018.12.015

3. Mathematical model for SDN multi-controller

In order to formulate the problem and identify the fitness function for the optimization problem, we will model the system at first. SDN multiple controllers network employs M switches that support an appropriate SDN interface, such as OpenFlow. OpenFlow switches are connected to N controllers that are distributed over the network. The set of controllers is C_i (i = 1,2,. . .,N), and the switches set is S_j (j = 1,2,. . .,M).

In order to perform the network operation, each OpenFlow switch requests the service from the corresponding controller. The controller responds with the switching rules and forwarding table. Since the controller's resources including storage, bandwidth and processing are limited; thus, a controller C_i is able to handle and manage a limited number of OpenFlow switches K_i. K_i is the number of switches associated with a certain controller C_i at a time, and can be calculated from the controller-switch matrix by summing up the number of ones in the controller's row. An example of a controller-switch matrix with N = 5 and M = 7 is represented in (1), with the K matrix that includes the number of connected switches with each controller. Rows of the matrix represent different controllers, while columns maps for switches. Controllers shouldn't be overloaded; this is because the failure probability is highly increased with the increase of the controller's load, once it reaches a threshold level.

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \gg\gg [K] = \begin{bmatrix} 2 \\ 1 \\ 2 \\ 1 \\ 1 \end{bmatrix} \tag{1}$$

One way to check the performance of the controller is by evaluating the time response of the controller that mainly affected by the queuing delay [40]. Controllers can be modeled using multi-server queuing model M/M/s [41]. The average response time T_i of the controller C_i is the sum of the queuing time and the processing time, and can be calculated using Erlang's C formula as a function of the arrival rate λ_i and the service rate μ.

$$T_i(\lambda) = \frac{C\left(s, \frac{\lambda_i}{\mu}\right)}{s\mu_i - \lambda_i} + \frac{1}{\mu} \tag{2}$$

Where, C (s, λ/μ) is the probability that all system servers are in use and any arriving packet will be queued, and can be calculated as in (3).

$$C\left(s, \frac{\lambda}{\mu}\right) = \frac{\left(\frac{(s\rho)^c}{s!}\right)\left(\frac{1}{1-\rho}\right)}{\sum_{k=0}^{s-1}\frac{(s\rho)^k}{k!} + \left(\frac{(s\rho)^c}{s!}\right)\left(\frac{1}{1-\rho}\right)} = \frac{1}{1 + \left(\frac{1}{1-\rho}\right)\left(\frac{s!}{(s\rho)^c}\right)\sum_{k=0}^{s-1}\frac{(s\rho)^k}{k!}} \tag{3}$$

$$\rho = \frac{\lambda_i}{s \cdot \mu} \tag{4}$$

where, ρ represents the server utilization, which is an indication of the system stability. The system has a stable distribution, only if the server utilization ρ is less than one. This can be interpreted using the state transition diagram of M/M/s model; when the received requests in the queue is greater than the controller's servers, the transition will still with $s\mu$, no more, and the controller is at maximum capacity.

The arrival rate λ_i of a controller C_i can be calculated as the sum of the average arrival rate of switches connected to the controller.

$$\lambda_i = \sum_{k_i} \lambda_s \tag{5}$$

The average load on a controller C_i can be calculated as the average requests queued and processed. Using Erlang's C formula the average controller load L_i can be calculated using (6).

$$L_i(\lambda) = s\rho + \frac{\rho}{1-\rho} C\left(s, \frac{\lambda_i}{\mu}\right) \tag{6}$$

4. Latency-aware, cost-aware controller placement problem

4.1. Problem formulation

The dynamic change of the network load makes the optimal number and the optimal locations of different controllers change dynamically in accordance. Consequently, we seek to solve an optimization problem to evaluate the dynamic value of optimal number of controllers and moreover, the dynamic optimum places of different controllers deployed in the network.

The network load represents the graphic controller for both, the number of controllers deployed and their locations. Once the network load increases, new controller or more should be deployed to handle the added load and compensate for the controller overloading problems. In the other side, if the network load is decreased, some of the deployed controllers should go in a sleep mode or separated, based on the amount of load reduction.

Allocating new controllers or cutting existing ones can be processed by the introduced optimizer. Moreover, their optimal locations are also introduced. The main objective of our framework is to find the minimum number of SDN controllers with their optimum locations and the optimum assignments of controllers to distributed switches. These optimal solutions should satisfy the main restrictions associated with the time response and utilization of each controller. The problem can be modeled as following:

$$Min\ f(N, C, D) \tag{7}$$

Subjected to:

$$T_i \leqslant \tau\ ,\ \forall\ i\ \in A \tag{8}$$

$$U_{lb} \leqslant\ U_i \leqslant U_{ub},\ \forall\ i\ \in A \tag{9}$$

where, f is a non-linear function of the number of deployed controllers N, D is the average delay between controller and connected switches (i.e. propagation, queuing and processing) and C is the cost of deployed controllers which represents both CAPEX and OPEX. This problem represents a multi-objective with multi-constraint optimization problem.

The first constraint indicates that the average response time T_i of the controller C_i should be less than a threshold value τ, which is predefined; this takes place for all controllers in the set of available controllers [A]. τ is predefined in a way to present a certain quality of service (QoS). The second restriction is concerned with the utilization index of each controller, which should be located between the lower bound of a controller utilization index U_{lb} and the upper bound of a controller utilization index U_{ub}. Both values of U_{ub} and U_{lb} are predefined in a way that supports the QoS of the system. The utilization index of a controller is used for mapping to power, storage and processing utilization.

4.2. System utilization

In this section, we define a system utilization function, which represents the fitness that is used to compare between different solutions and point to the best solution. The utilization function

Please cite this article as: A. A. Ateya, A. Muthanna, A. Vybornova et al., Chaotic salp swarm algorithm for SDN multi-controller networks, Engineering Science and Technology, an International Journal, https://doi.org/10.1016/j.jestch.2018.12.015

of a system is generally used to map values of variables or events into real numbers. It can be generally defined in (10).

$$u : X \rightarrow \mathbb{R} \tag{10}$$

The first utility that should be considered is the time utilization function, which maps to the time response of each controller; any form of common loss functions can be used. We introduce the quadratic function as it is mathematically tractable due to symmetry [42]. The time utility of a controller C_i is $U_T{}^{Ci}$ is defined as introduced in (11).

$$U_T^{C_i} = \begin{cases} \alpha + \beta(\tau - T_i(\lambda))^2 & , \quad T_i(\lambda) \leqslant \tau \\ 0 & , \quad T_i(\lambda) > \tau \end{cases} \quad \forall\ i\ \in A \tag{11}$$

where α and β are constants that their values have no effect on the decision. The first constant α can be assigned a certain value that represents the minimum non zero value of the time utilization $U_{T\text{-}thr}$ occurred when the response time equal to the threshold value. Eqs. (12) and (13) define both constants.

$$\alpha = \ U_{T-thr} \qquad \forall\ U_T \in \ [0,1] \tag{12}$$

$$\beta = \frac{(1 - U_{T-thr})}{\tau^2} \qquad \forall\ U_T \in \ [0,1] \tag{13}$$

For a threshold time utilization of each controller of 70%, the time utilization can be recalculated as in (14).

$$U_T^{C_i} = \begin{cases} 0.7 + \frac{0.3}{\tau^2}(\tau - T_i(\lambda))^2 & , \quad T_i(\lambda) \leqslant \tau \\ 0 & , \quad T_i(\lambda) > \tau \end{cases} \quad \forall\ i\ \in A \tag{14}$$

Another important utility that should be considered is the cost utility function, which maps to the cost of used controllers. The cost mainly refers to both terms, CAPEX and OPEX, dedicated with deployed controllers. The quadratic loss function is also represents a proper function for expressing the cost utilization. The cost utilization of each controller, in the set of available controllers, can be defined as following:

$$U_C^{Ci} = \begin{cases} \varphi(U_{ub} - U_{ci})^2 \ \forall\ U_{ci}\ \in\ [U_{lb}\ , U_{ub}] \\ 0 \qquad\qquad\quad \forall\ U_{ci}\ \notin\ [U_{lb}\ , U_{ub}] \end{cases} \quad \forall\ i\ \in A \tag{15}$$

where Φ is a constant that has no effect on the decision; whatever, what value it is assigned. A proper value for Φ can be defined in (16) for a U_C between 0 and 1.

$$\varnothing = \frac{1}{(U_{ub} - U_{lb})^2} \qquad \forall\ U_C \in \ [0,1] \tag{16}$$

Both utility functions represent the total utility of each controller in the network. In order to add both utilities, each of them must be weighted. Equation (17) identifies the total utility of each controller U_{Ci}.

$$U_{ci} = \delta_C U_C^{Ci} + \delta_T U_T^{Ci} \quad \forall\ i\ \in A \tag{17}$$

where δ_C is the cost weighting factor and δ_T is the time weighting factor. The total system utilization function represents the average value of the utilization of each controller. Thus, the total utilization function U can be calculated as following:

$$U = \sum_A U_{ci}/|A| \tag{18}$$

Also the total cost utility U_{CT} and the total time utility U_{TT} of all available controllers can be calculated as the following:

$$U_{CT} = \sum_A U_C^{Ci}/|A| \tag{19}$$

$$U_{TT} = \sum_A U_T^{Ci}/|A| \tag{20}$$

4.3. Salp swarm algorithm (SSA)

SSA is a meta-heuristic population based algorithm, which simulates the behavior of salps in oceans [43]. Salps are a species of Salpidae that look like jelly fishes with transparent shape. Salps navigate and forage in a swarm referred to as salp chain. SSA is a recent type of PSO that models the salp chain [43]. The salp population consists of the leader salp and the follower salps. The leader salp is the first salp in the salp chain, which leads the swarm. Follower salps follow the leader salp for reaching the food source.

The position of each salp is a d-dimensional search space with d represents the number of variables in a certain problem, in the same way as other swarm based algorithms. The current position vector of n salps in the search space is $X^j = [x_1^j, x_2^j, x_3^j, \ldots\ldots, x_d^j]$, j = 1,2,. . ..,n. The leader salp updates its position using (21).

$$X_i^1 = \begin{cases} F_i + C_1((ub_i - ul_i)C_2 + lb_i), & C_3 \geqslant 0 \\ F_i - C_1((ub_i - ul_i)C_2 + lb_i), & C_3 < 0 \end{cases} \tag{21}$$

where X_i^1 indicates the position of the leader salp in i^{th} dimension, F_i is the food position in the i^{th} dimension, ub_i and lb_i represent the upper and lower boundary in i^{th} dimension and C_1, C_2 and C_3 are the model coefficients. These coefficients are random numbers that are used for certain objectives. The first coefficient C_1 is introduced to make balance between the exploration and the exploitation, which represents the most important parameter in the algorithm. C_1 is defined in (22).

$$C_1 = 2e^{-\left(\frac{4t}{T_{max}}\right)^2} \tag{22}$$

where t is the current iteration and T_{max} is the maximum number of iterations. C_2 and C_3 are random numbers that uniformly generated with values between 0 and 1. The follower salps update their positions based on Newton's law of motion using the following equation:

$$X_i^k = \frac{1}{2}\left(X_i^k + X_i^{k-1}\right) \qquad 2 \leqslant k < n \tag{23}$$

where X_i^k indicates the position of the k^{th} follower salp in i^{th} dimension and n is the total number of salp particles.

4.4. Chaotic salp swarm algorithm (CSSA)

Population based meta-heuristic algorithms share varies advantages include scalability, simplicity and computational time reduction. However, these algorithms have two main disadvantages; recession in local optima and low convergence rate [44]. One way to overcome these problems and enhance the performance of meta-heuristic algorithms is to deploy the chaos theory [45]. Chaotic maps are used instead of random numbers in PSO based algorithms to enhance the convergence.

In this way, authors in [46] introduce a chaotic based SSA (CSSA), which replaces random variables with chaotic ones. CSSA uses chaotic maps to adjust the value of the second coefficient C_2 [39]. The value of C_2 can be replaced by the value of an appropriate chaotic map at the current iteration, as following:

$$C_2^t = \omega(t) \tag{24}$$

where $\omega(t)$ is the value of the chaotic map at the t^{th} iteration. Equation (21) can be rewritten, using the new value of C_2, as following:

$$X_i^1 = \begin{cases} F_i + C_1((ub_i - ul_i)\omega(t) + lb_i), & C_3 \geqslant 0 \\ F_i - C_1((ub_i - ul_i)\omega(t) + lb_i), & C_3 < 0 \end{cases} \tag{25}$$

4.5. Chaotic maps

Chaos theory is a common mathematical approach used to analyze the behavior of dynamic systems with critical initial

Please cite this article as: A. A. Ateya, A. Muthanna, A. Vybornova et al., Chaotic salp swarm algorithm for SDN multi-controller networks, Engineering Science and Technology, an International Journal, https://doi.org/10.1016/j.jestch.2018.12.015

conditions [47]. One way to display this behavior is by using chaotic maps that are either discrete or continuous. Chaotic maps can be deployed only for deterministic systems with a predictable behavior. Recently, chaos theory becomes more attractive to many systems in various fields such as computer science, robotics, physics and microbiology [48].

Chaotic maps become the most powerful solution to enhance the performance of meta-heuristic algorithms by enhancing their randomness parameters. These random parameters are extracted based on a uniform or Gaussian distribution, so they can be better controlled by a chaotic map shares the same characteristic with better performance [49]. Controlling these parameters using chaotic maps reduces the local optima and increases the convergence.

Based on the results obtained in [46], the optimum chaotic map for our optimizer is the logistic map. Logistic map was first introduced by Robert May early in 1976 [50]. The general equation for logistic chaotic map is:

$$\omega(t+1) = a\omega(t)[1-\omega(t)],\ a = 4 \tag{26}$$

where, ω(t) is the value of chaotic map at the t^{th} iteration. The initial condition of the chaotic map is assumed to be 0.7 (ω (0) = 0.7) [51].

4.6. CSSA for optimal number of controllers and optimal allocations

In SSA, the food position, which is the best salp position, represents the solution of the optimized problem. For the problem modeled, we aim to fine the optimal number of controllers and the optimal connection for each switch in the switch set S. This problem is considered to be an NP-hard problem; one way to solve this kind of problems is by using meta-heuristic algorithms, as the deterministic ones fails. We use the SSA, which is a kind of particle swarm algorithms, to solve this problem. SSA is the recent PSO based algorithm, which introduces higher performance than other PSO based algorithms [46].

The main idea behind SSA is, by iteration, multiple salps search in parallel to get the optimal solution. The optimum solution is the optimal number of controllers and the optimal distribution of controllers among switches. Consequently, two algorithms, in a nested loop, are deployed to get the best solutions; the two algorithms are based on CSSA. Algorithm 1 indicates the pseudo code for the CSSA introduced for the defined problem, where each salp represents the number of controllers in the network. The output of this algorithm represents the optimal number of controllers. Algorithm 2 indicates the pseudo code for the CSSA deployed for finding the optimum connections for all switches based on the optimum number of controllers calculated by algorithm 1. Each salp in algorithm 2 represents all available connections for all switches with their dedicated controllers, and it is an M-dimensional vector with each dimension represents a switch. The output of the second algorithm indicates the best distribution of controllers among switches.

The system starts working by setting the initial parameters of CSSA includes lower boundary, upper boundary and the maximum number of iterations, and then initializing n salps randomly with each salp represents the number of available controllers in the network. The fitness of each salp is calculated using (18) and the salp with highest fitness is considered to be the current best solution. The location of highest fitness salp is considered to be the food position. The parameters of SSA are updated and in accordance the positions of salps are updated. The process of updating salps positions and evaluating fitness of each salp is repeated until the optimum solution is found or reaching the maximum iteration.

Algorithm 1 CSSA for optimal number of controllers

```
Initialize ub, lb, t_max, d, n
Initialize positions of salps x_i (i = 1,2,3,...,n)
While (t ≤ t_max)
    Calculate the fitness function of each salp position using (18)
    F = The best salp position
    Update the value of C1 using (22)
    Get the value of chaotic map (Logistic) w(t)
    For (i = 1 : i ≤ n) do
      if (i == 1)
          Update the position of leading salp using (25)
      else
          Update the position of follower salp using (23)
      end if
    end for
    Adjust salps based on the upper and lower bounds
    Calculate the best connections of switches for the best salp (call Algorithm 2)
    Update the best salp based on the results of Algorithm 2
    t ← t + 1
Return F
```

Algorithm 2 CSSA for optimal switches to controllers connections

```
Initialize ub, lb, t_max, d, n
Initialize positions of salps x_i (i = 1,2,3,...,n)
While (t ≤ t_max)
    Calculate the fitness function of each salp position using (18)
    F = The best salp position
    Update the value of C1 using (22)
    Get the value of chaotic map (Logistic) w(t)
    For (i = 1 : i ≤ n) do
      if (i == 1)
          Update the position of leading salp using (25)
      else
          Update the position of follower salp using (23)
      end if
    end for
    Adjust salps based on the upper and lower bounds
    t ← t + 1
Return F
```

5. Performance evaluation

For the performance evaluation of the proposed model, a simulation environment is conducted. The proposed algorithm is implemented, for different topologies, using Matlab. In this part, the simulation set up is provided at first. Then, simulation results and results analysis are carried out.

5.1. 1. Simulation setup

The proposed algorithm is implemented and tested using Matlab over a machine equipped with Intel Core i5 processor and 8 GB RAM. Ten approximated topologies, from the Internet Topology Zoo, are considered [52]. The ten topologies are divided into two main categories; the first category includes topologies with a small number of forwarding devices and the other category contains topologies with a heavy number of forwarding devices. Table 1 illustrates the selected topologies, with the total number of forwarding devices deployed in each topology. The Internet Topology

Please cite this article as: A. A. Ateya, A. Muthanna, A. Vybornova et al., Chaotic salp swarm algorithm for SDN multi-controller networks, Engineering Science and Technology, an International Journal, https://doi.org/10.1016/j.jestch.2018.12.015

Table 1
Considered topologies.

Ref. Number	Topology	Quantity
Category (I)		
1	ARPANET 1969_12	4
2	MREN	6
3	GetNet	7
4	Sprint	11
5	NSF	13
Category (II)		
6	Claranet	15
7	IBM	18
8	Oxford	20
9	FCCN	23
10	AGIS	25

Zoo is considered, because its areal situations for different wide area networks topologies. Every Point-of-Presence (PoP) in the network topology is considered to be a network switch (i.e. forwarding device) [53]. The flow is assumed to be a random variable that follows Poisson distribution. Simulation parameters, included in the evaluation process, are introduced in Table 2. For the first developed SDN controller NOX, the average flow requests handled per second is in order of 30000[54,55]. Thus, the average service rate of controller is set to 30000.

5.2. 2. Simulation results

For the considered ten topologies, two main scenarios are considered for each topology. In the first scenario, we aim to analyze the effect of changing threshold time τ on the decision (i.e. optimal number of controllers). In this situation, the upper bound utilization index of each controller U_{ub} is assumed to be constant and equal to the value in the simulation parameters table (i.e. $U_{ub} = 0.9$). The proposed algorithm is implemented multiple times for each topology; each time represents a case and in each case a certain value of time response threshold τ is considered. Table 3 indicates different values of τ for scenario (I).

In the second scenario, the effect of variation of upper utilization index of each controller is checked, at constant value of threshold delay τ. The upper utilization index is mainly considered, for each available controller, to compensate for any sudden increase in network traffic and to avoid controller working at maximum capacity [31,56].

The proposed algorithm is implemented for each network topology multiple times; in each time a new value of U_{ub} is considered. For all cases in scenario (II), the threshold time response τ is set to the value defined in the simulation parameters table (i.e. $\tau = 2$ ms). Different values considered for U_{ub} are included in Table 3.

Table 2
Simulation parameters.

Parameter	Description	Quantity
λ_S	Average request rate of switch	[1500, 3000]
μ	Service rate of controller	30,000 req/sec
τ	Latency threshold	2 ms
U_{ub}	Upper bound of controller utilization index	0.9
t_{max} (1)	Maximum iterations for Algorithm 1	50
t_{max} (2)	Maximum iterations for Algorithm 2	30
δ_C	Cost weighting factor	18
δ_T	Time weighting factor	25

Table 3
Parameters variation for the two considered scenarios.

Ref. Number	Scenario (I) $U_{ub} = 0.9$ [Const.]	Scenario (II) $\tau = 2$ ms [Const.]
Case (1)	$\tau_1 = 1$ ms	$U_{ub1} = 0.80$
Case (2)	$\tau_2 = 2$ ms	$U_{ub2} = 0.85$
Case (3)	$\tau_3 = 3$ ms	$U_{ub3} = 0.90$
Case (4)	$\tau_4 = 4$ ms	$U_{ub4} = 0.95$

Fig. 3 illustrates results for different cases in the first considered scenario. Fig. 3(a) indicates the optimal number of controllers in each case of scenario (I) for the first category of topologies, while Fig. 3(b) indicates the same for the second category of topologies. For each case, the optimum number of controllers to be deployed for the network is mainly decreased, with the increase of the constraint threshold time τ. Moving from case to case, for the same number of network switches (i.e. the same topology), the best number of required controllers either maintains or decreases. This can be interpreted as the increase of the threshold time τ allows the increase of the response time of each deployed controller; thus, the controller can handle more number of forwarding devices. This comes on the account of the latency, which should support the required QoS of the system.

Results for different cases of the second scenario are presented in Fig. 4. Fig. 4(a) illustrates the optimal number of controllers for each case in scenario (II) for the category (I) topologies, and the results for the category (II) topologies are presented in Fig. 4(b). Results indicate that, with the increase of the upper bound of controller utilization index, the optimal number of controller required to support the network decreases.

For each topology, either first or second category, as the upper utilization index of each deployed controller U_{ub} increases, the required number of controllers decreases. Moving from case to case, for each topology, the number of available controllers either maintains or decreases. Increasing upper utilization index of controller allows the controller to offer more resources and thus, handles more tasks. This is the main reason for the reduction of deployed controllers with the growing of U_{ub}. The best value for upper utilization index is associated with the nature of the network and the probability of rapid flow changing. The most common used value of U_{ub} is o.9, which indicates that 90 percent of the controller resources are in use and ten percent of the resources are separated for abrupt changes.

For better evaluation of the introduced algorithm and for checking the effect of parameters variations, the proposed algorithm is

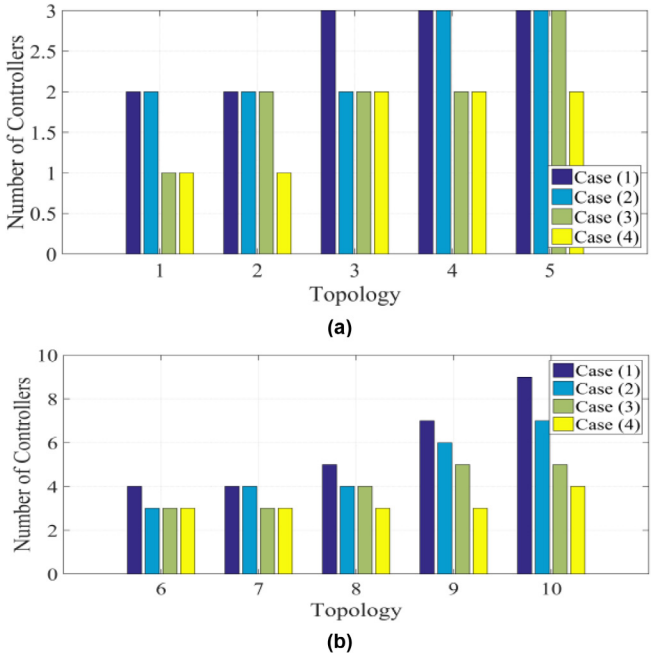

Fig. 3. Simulation results for scenario I (a) Simulation results for Category I topologies, (b) Simulation results for Category II topologies.

Please cite this article as: A. A. Ateya, A. Muthanna, A. Vybornova et al., Chaotic salp swarm algorithm for SDN multi-controller networks, Engineering Science and Technology, an International Journal, https://doi.org/10.1016/j.jestch.2018.12.015

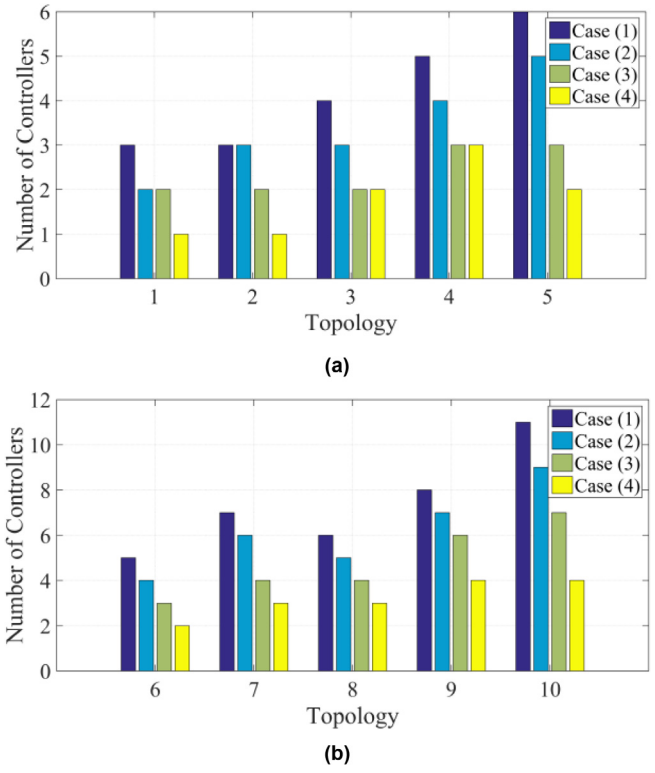

Fig. 4. Simulation results for scenario II (a) Simulation results for Category I topologies, (b) Simulation results for Category II topologies.

implemented for a random network of a 30 switches. The system is checked for the variation of the threshold time τ as well as the upper utilization index U_{ub} at a larger scale than that considered in the previous simulations. This is because of the importance of these parameters and their effects, especially for large scale networks. Therefore, algorithms should be conducted for dense networks to validate the proposed work.

Two main cases are considered; in the first case the algorithm is implemented for different values of threshold time τ with four different values of maximum controller utilization index U_{ub}. The second case checks the system for wide range of values of U_{ub} with four different values of threshold time τ. Table 4 indicates both values of τ and U_{ub} in both cases.

Fig. 5 illustrates the effect of variations of the threshold response of the controller (τ) on the optimal number of controllers required for the network, in four cases. Each case represents a certain value of the maximum utilization index of controller U_{ub}. The four values are chosen such that they cover the possible range of U_{ub} and thus, give a reliable presence. The first curve represents the variation at a utilization index of 80 percent, which is considered to be a minimum utilization. Thus, the curve represents the worst case between the four curves as it is dedicated with the worst value of U_{ub}. In the other side, the fourth curve represents the variation at the maximum utilization index of 95 percent, which is the best case between the four graphs. For the four curves, with the increase of threshold time (i.e. moving from left to right)

Table 4
Parameters variation for random network.

Maximum utilization index (U_{ub})	Threshold time (τ)
U_{ub1} = 0.80	τ_1 = 1 ms
U_{ub2} = 0.85	τ_2 = 5 ms
U_{ub3} = 0.90	τ_3 = 10 ms
U_{ub4} = 0.95	τ_4 = 15 ms

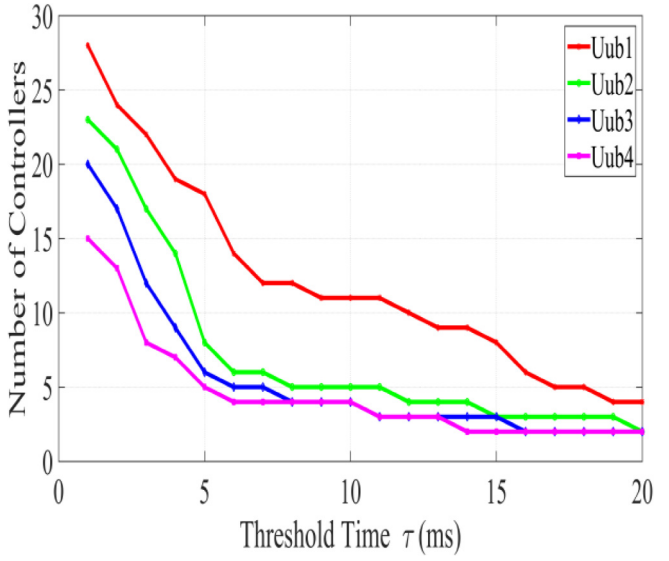

Fig. 5. The effect of variation of threshold time τ on the optimal number of deployed controllers, at different values of U_{ub}.

the required number of controllers decreases. The worst values of the number of controllers needed are dedicated with the first region of curves, at which the threshold time τ is small and thus, controllers are not able to handle much tasks. The best values of number of controllers are located at the last parts of all curves, at which the value of τ is high enough to handle much tasks and serves for much switches.

Fig. 6 indicates the variation of optimal number of controllers with the change of maximum utilization index of each controller, for four different values of the threshold time τ. The first curve represents the worst one as it indicates the highest values of number of controllers required. This is because the small value of the threshold time prevents the controller from handling much tasks and thus, the controller can only serve for a minimum number of forwarding devices.

Unlike the first curve, the fourth curve indicates best results in terms of number of required controllers. This can be interpreted as

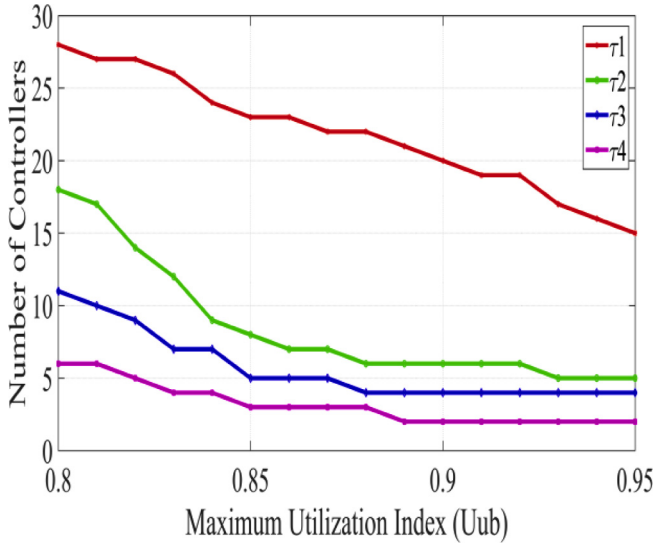

Fig. 6. The effect of variation of maximum utilization index (U_{ub}) on the optimal number of deployed controllers, at different values of τ.

Please cite this article as: A. A. Ateya, A. Muthanna, A. Vybornova et al., Chaotic salp swarm algorithm for SDN multi-controller networks, Engineering Science and Technology, an International Journal, https://doi.org/10.1016/j.jestch.2018.12.015

the high value of the threshold time τ allows the controller to support more switches.

Another important parameter that should be considered is the flow request variation. Changing the rate of flow request of switches leads to a change in the situations of the deployed controllers. The number of deployed controllers should be increased if the flow rate increases. Thus, more network load should be faced by activating more controllers, while the reduction of the network load should produces less number of active controllers. Consequently, the network starts with certain number of active controllers and by the time, the number of active controllers rises and falls based on the network load. Dynamic variation of network load introduces a dynamic change of number of active controllers.

In order to check the effect of dynamic changes of flow requests on the number of active controllers, four cases with different ranges of flow rates of each forwarding device are considered. For each case, three values of the threshold time τ and maximum utilization of each controller U_{ub} are considered. Table 5 indicates the considered cases and the values of τ and $U_{ub,}$ considered in each case, are included in Table 6.

Figs. 7 and 8 indicate the simulation results for the five cases. In Fig. 7, the variation of flow requests is checked in three situations; each of which represents a value of threshold time τ. This takes place at a constant maximum utilization index of each controller U_{ub} of 90 percent. Moving from case to case, the optimal number of active controllers increases. This is because, the increase of flow requests increases the load on active controllers and thus, more controllers should be activated. Fig. 8 illustrates the optimal number of controllers in each case, for three values of maximum utilization index of controller U_{ub}. This is at constant threshold time τ of 10 ms.

Table 7 indicates the numerical results of the system implementation for different values of threshold time τ; for the first two cases of λ_S. Other simulation parameters are conducted from the simulation parameters table (i.e. Table 2). Results include the total system utilization U at the corresponding value of τ and the K-matrix that includes the number of connected switches associated with each controller. Table 8 indicates the numerical results for different values of U_{ub}; for the first two cases of λ_S. The total utilization increases with the increase of the value of the threshold time τ. This is because the small value of τ requires the deployment of more controllers, which decreases the overall utilization. Also, the total utilization decreases with increase of flow requests.

Table 5
Different cases of flow requests of switches.

Case	Range of λ_S
Case (1)	$\lambda_S \in$ [1500, 3000]
Case (2)	$\lambda_S \in$ [3000, 4500]
Case (3)	$\lambda_S \in$ [4500, 6000]
Case (4)	$\lambda_S \in$ [6000, 7500]
Case (5)	$\lambda_S \in$ [7500, 9000]

Table 6
Values of three should time and maximum utilization index for flow requests variation cases.

Parameter	Value
τ_1	5 ms
τ_2	10 ms
τ_3	15 ms
U_{ub1}	0.85
U_{ub2}	0.90
U_{ub3}	0.95

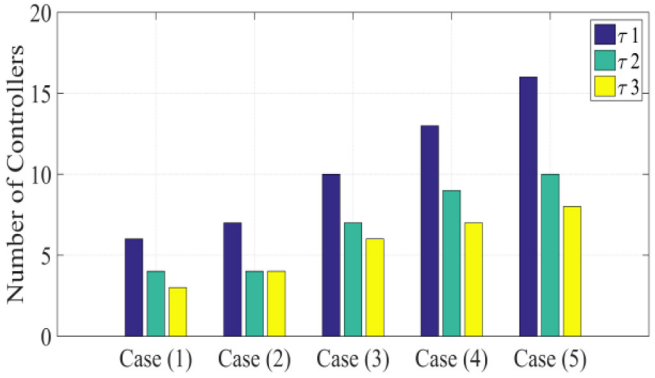

Fig. 7. Variation of flow requests at different values of threshold time τ.

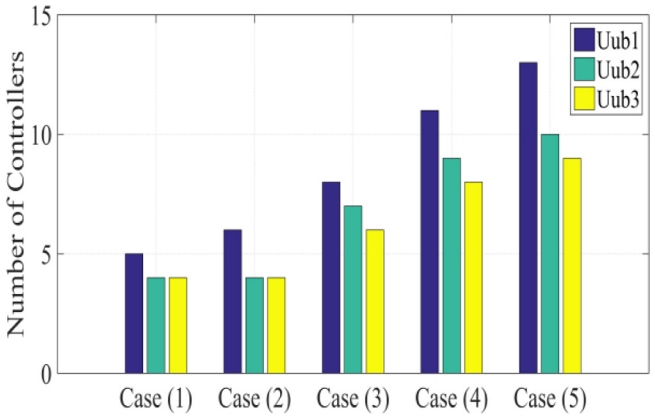

Fig. 8. Variation of flow requests at different values of maximum utilization index $U_{ub.}$

Table 7
Numerical results for threshold time variation.

Threshold Time (τ)	Number of Controllers	$[K]^T$	Total Utilization (U)
U_{ub} = 0.9, $\lambda_S \in$ [1500, 3000]			
4 ms	9	[4 3 3 4 2 4 5 3 2]	0.855
6 ms	5	[7 6 6 5 6]	0.863
8 ms	4	[9 7 8 6]	0.872
10 ms	4	[8 8 7 7]	0.884
12 ms	3	[11 12 7]	0.889
U_{ub} = 0.9, $\lambda_S \in$ [3000, 4500]			
4 ms	10	[3 4 3 4 2 4 4 2 2 2]	0.845
6 ms	7	[5 4 4 5 5 4 3]	0.852
8 ms	5	[6 5 5 7 7]	0.868
10 ms	4	[6 8 8 8]	0.881
12 ms	3	[12 12 6]	0.886

5.3. 3. System comparison

In this part, the proposed CSSA for SDN controller placement is compared with other proposed algorithms developed for the same problem, to evaluate the performance and advantages of the introduced algorithm. Three main algorithms are considered for the comparison; Non-Zero-Sum game algorithm introduced in [31] and two efficient swarm intelligence algorithms. The two

Please cite this article as: A. A. Ateya, A. Muthanna, A. Vybornova et al., Chaotic salp swarm algorithm for SDN multi-controller networks, Engineering Science and Technology, an International Journal, https://doi.org/10.1016/j.jestch.2018.12.015

Table 8
Numerical results for maximum utilization index variation.

Maximum Utilization Index (U_{ub})	Number of Controllers	$[K]^T$	Total Utilization (U)
τ = 10 ms, $\lambda_S \in$ [1500, 3000]			
0.82	9	[3 3 2 4 4 4 3 4 3]	0.864
0.84	7	[4 5 4 3 6 4 4]	0.867
0.86	5	[6 6 7 7 4]	0.874
0.88	4	[8 7 7 8]	0.882
0.90	4	[8 8 7 7]	0.884
τ = 10 ms, $\lambda_S \in$ [3000, 4500]			
0.82	10	[2 3 3 4 2 4 3 4 3 2]	0.857
0.84	7	[6 5 5 4 3 4 3]	0.864
0.86	6	[6 4 6 5 4 5]	0.871
0.88	5	[7 6 6 5 6]	0.878
0.90	4	[6 8 8 8]	0.881

considered swarm intelligence techniques are Particle Swarm algorithm (PSO) and Grey Wolf algorithm (GWO).

Particle Swarm Optimization (PSO) is the most common swarm intelligence technique that simulates the behavior of birds, while navigating and hunting [57]. Gray Wolf Optimization (GWO) is a swarm based technique that mimics the behavior of grey wolves while hunting [58]. The GWO divides the crow of gray wolfs into four types; alpha, beta, delta, and omega. The alpha wolf is the leader of the group and it represents the best solution. Beta and delta wolfs represent the second and third leader, and it give the second and third best solutions of the problem. Equations used to model and describe both PSO and GWO are considered in Appendix A.

The performance metric considered for comparison process, between the proposed algorithm and the three considered algorithms, is the execution time of algorithm. This time represents the delay taken by the algorithm to get the optimal solution, and is considered to be an important metric when evaluating the optimization algorithms. Moreover, this time is critical for SDN networks as if it takes long to activate controllers in case of the network load changes up, current deployed controllers may be over loaded and go into failure.

Fig. 9 illustrates the execution time of each algorithm with the variation of threshold time τ. CSSA achieves better performance, in terms of computing time for all values of τ. Then the GWO algorithm comes, it takes less time than PSO to get the optimum solution. The worst algorithm, in terms of computing time, is the Non-Zero-Sum game algorithm. The execution time represents a very important metric as the fast execution achieves better network performance and supports the QoS, beside it decreases the probability of controller failure.

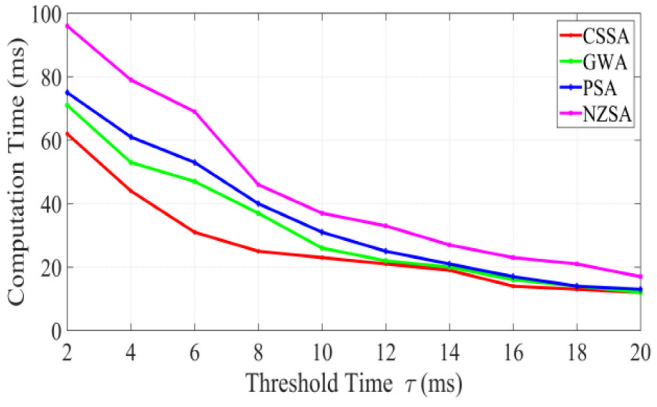

Fig. 9. Computation time of each algorithm with the variation of threshold time.

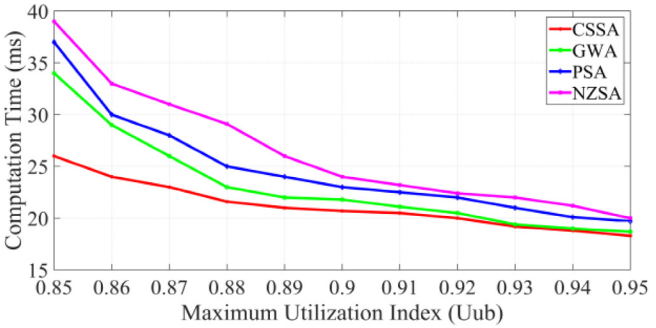

Fig. 10. Computation time of each algorithm with the variation of maximum utilization index.

Moreover, the execution time is recorded with the variation of maximum utilization index of the controller U_{ub}, for proposed CSSA and the other three considered algorithms. Fig. 10 indicates the results for the four algorithms. CSSA achieves the best performance in terms of computing time, for all considered values of U_{ub}. Another main important issue is that the three considered algorithms, in some considered cases, couldn't reach the optimum solution due to local optima, which solved in the proposed algorithm by involving chaotic maps.

6. Conclusion

This work introduces the latency and cost aware controller placement problem. The problem is defined and a meta-heuristic algorithm is presented to solve the problem for the optimum solution. The algorithm is a chaotic SSA that is developed to get the optimal number of controllers and also the optimum allocations of switches to controllers, that minimize the latency and the deployment cost. The introduction of chaotic maps improves the optimizer performance and prevents the local optima. The algorithm is tested for various real topologies extracted from the zoo topology. The effect of variation of different network parameters on the performance is checked. Simulation results validate the proposed work and a comparison with other meta-heuristic algorithms and a game theory based algorithm is presented.

Appendix A

In this appendix, the mathematical analysis and description of PSO and GWO is introduced. In PSO, each particle is described as a d-dimensional vector in the search space. The current position and velocity vectors of each particle are $X_i = [X_i^1, X_i^2, X_i^3, \ldots, X_i^d]$ and $V_i = [V_i^1, V_i^2, V_i^3, \ldots, V_i^d]$, i = 1,2,3,. . .,n. The velocity and position update equations of particles are defined as following [57]:

$$V_i(t+1) = \omega V_i(t) + C_1 r_1 (P_{best} - X_i(t)) + C_2 r_2 (P_{gbest} - X_i(t)) \quad (27)$$

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (28)$$

$$\omega = \omega_{\max} - \frac{\omega_{\max} - \omega_{\min}}{t_{\max}} \times t \quad (29)$$

where P_{best} and p_{gbest} are the individual best position of particle i and the global best position of particles, respectively. Coefficients r_1 and r_2 are random sequences in the range of (0, 1), C_1 and C_2 are the algorithm coefficients and ω is the inertia weight that can be calculated based on (29). For GWO, the three best wolfs (i.e. alpha, beta and delta) update their positions using the following equations [58]:

Please cite this article as: A. A. Ateya, A. Muthanna, A. Vybornova et al., Chaotic salp swarm algorithm for SDN multi-controller networks, Engineering Science and Technology, an International Journal, https://doi.org/10.1016/j.jestch.2018.12.015

$$\overrightarrow{D_\alpha} = \left|\overrightarrow{C_1}.\overrightarrow{X_\alpha} - \overrightarrow{X}\right|, \overrightarrow{D_\beta} = \left|\overrightarrow{C_2}.\overrightarrow{X_\beta} - \overrightarrow{X}\right|, \overrightarrow{D_\delta} = \left|\overrightarrow{C_3}.\overrightarrow{X_\delta} - \overrightarrow{X}\right| \tag{30}$$

$$\overrightarrow{C_1} = \overrightarrow{X_\alpha} - \overrightarrow{A_1}.\overrightarrow{D_\alpha}, \overrightarrow{C_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2}.\overrightarrow{D_\beta}, \overrightarrow{C_3} = \overrightarrow{X_\delta} - \overrightarrow{A_3}.\overrightarrow{D_\delta} \tag{31}$$

$$\overrightarrow{X}(t+1) = \frac{\overrightarrow{C_1} + \overrightarrow{C_2} + \overrightarrow{C_3}}{3} \tag{32}$$

References

[1] J.H. Cox, J. Chung, S. Donovan, J. Ivey, R.J. Clark, G. Riley, H.L. Owen, Advancing software-defined networks: a survey, IEEE Access 5 (2017) 25487–25526.

[2] B.A.A. Nunes, M. Mendonca, X.N. Nguyen, K. Obraczka, T. Turletti, A survey of software-defined networking: past, present, and future of programmable networks, IEEE Commun. Surv. Tutorials 16 (3) (Feb. 2014) 1617–1634.

[3] K. Wang, Y. Wang, D. Zeng, S. Guo, An SDN-based architecture for next-generation wireless networks, IEEE Wirel. Commun. 24 (1) (Feb. 2017) 25–31.

[4] F. Bannour, S. Souihi, A. Mellouk, Distributed SDN control: survey, taxonomy and challenges, IEEE Commun. Surv. Tutorials (2017).

[5] A. Lara, A. Kolasani, B. Ramamurthy, Network innovation using openflow: a survey, IEEE Commun. Surv. Tutorials 16 (1) (Feb. 2014) 493–512.

[6] E. Haleplidis, J.H. Salim, J.M. Halpern, S. Hares, K. Pentikousis, K. Ogawa, W. Wang, S. Denazis, O. Koufopavlou, Network programmability with ForCES, IEEE Commun. Surv. Tutorials 17 (3) (Jan. 2015) 1423–1440.

[7] H.I. Kobo, A.M. Abu-Mahfouz, G.P. Hancke, A survey on software-defined wireless sensor networks: challenges and design requirements, IEEE Access 5 (2017) 1872–1899.

[8] I.F. Akyildiz, A. Lee, P. Wang, M. Luo, W. Chou, A roadmap for traffic engineering in SDN-OpenFlow networks, Comput. Netw. 71 (Oct. 2014) 1–30.

[9] D. Jiang, G. Liu, in: An Overview of 5G Requirements, 5G Mobile Communications, Springer, Cham, 2017, pp. 3–26.

[10] M. Hammoudeh, Applying wireless sensor networks to solve real-world problems. in: Proceedings of the International Conference on Intelligent Information Processing, Security and Advanced Communication, p.1, ACM, 2015.

[11] M.Y. Arslan, K. Sundaresan, S. Rangarajan, Software-defined networking in cellular radio access networks: potential and challenges, IEEE Commun. Mag. 53 (1) (Jan. 2015) 150–156.

[12] N.G.M.N. Alliance, 5G white paper. Next generation mobile networks, White Paper (2017).

[13] A.A. Ateya, A. Muthanna, A. Koucheryavy, 5G framework based on multi-level edge computing with D2D enabled communication, in: Proc. 20th International Conference on Advanced Communication Technology (ICACT), IEEE, Korea, Feb. 2018, pp. 507–512.

[14] M. Farhan, S. Jabbar, M. Aslam, M. Hammoudeh, M. Ahmad, S. Khalid, M. Khan, K. Han, IoT-based students interaction framework using attention-scoring assessment in eLearning, Future Gener. Comput. Syst. 79 (2018) 909–919.

[15] A. Basit, S. Qaisar, S.H. Rasool, M. Ali, SDN orchestration for next generation inter-networking: a multipath forwarding approach, IEEE Access 5 (2017) 13077–13089.

[16] F. Benamrane, R. Benaini, An East-West interface for distributed SDN control plane: implementation and evaluation, Comput. Electr. Eng. 57 (Jan. 2017) 162–175.

[17] X. Yang, L. Wang, SDN load balancing method based on K-Dijkstra, Int. J. Performability Eng. 14 (4) (Apr. 2018).

[18] T. Hu, Z. Guo, P. Yi, T. Baker, J. Lan, Multi-controller based software-defined networking: a survey, IEEE Access 6 (2018) 15980–15996.

[19] A. Tootoonchian, and Y. Ganjali, "Hyperflow: A distributed control plane for openflow, in: Proc. the 2010 internet network management conference on Research on enterprise networking, Apr. 2010, pp. 3–3.

[20] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky, Advanced study of SDN/OpenFlow controllers, in: Proc. the 9th central & eastern european software engineering conference, Russia, Oct. 2013.

[21] O. Blial, M. Mamoun, R. Benaini, An overview on SDN architectures with multiple controllers, J. Comput. Netw. Commun. (2016).

[22] G. Wang, Y. Zhao, J. Huang, W. Wang, The controller placement problem in software defined networking: a survey, IEEE Network 31 (5) (2017) 21–27.

[23] I. Ghafir, V. Prenosil, J. Svoboda, M. Hammoudeh, A survey on network security monitoring systems, in: Future Internet of Things and Cloud Workshops (FiCloudW), IEEE International Conference on, IEEE, 2016, pp. 77–82.

[24] A.K. Singh, S. Srivastava, A survey and classification of controller placement problem in SDN, Int. J. Netw. Manage. (2018).

[25] F. Yonghong, B. Jun, W. Jianping, C. Ze, W. Ke, L. Min, A dormant multi-controller model for software defined networking, China Commun. 11 (3) (Mar. 2014) 45–55.

[26] E. Hernandez-Valencia, S. Izzo, B. Polonsky, How will NFV/SDN transform service provider opex?, IEEE Netw 29 (3) (May 2015) 60–67.

[27] C. Qiu, C. Zhao, F. Xu, T. Yang, Sleeping mode of multi-controller in green software-defined networking, EURASIP J. Wireless Commun. Netw. 1 (2016) 282–2016.

[28] H. Huang, S. Guo, P. Li, B. Ye, I. Stojmenovic, Joint optimization of rule placement and traffic engineering for QoS provisioning in software defined network, IEEE Trans. Comput. 64 (12) (Dec. 2015) 3488–3499.

[29] H. Tahaei, R.B. Salleh, M.F. Ab Razak, K. Ko, N.B. Anuar, Cost effective network flow measurement for software defined networks: a distributed controller scenario, IEEE Access 6 (2018) 5182–5198.

[30] G. Wang, Y. Zhao, J. Huang, Y. Wu, An effective approach to controller placement in software defined wide area networks, IEEE Trans. Netw. Serv. Manage. 15 (1) (Mar. 2018) 344–355.

[31] H. K. Rath, V. Revoori, S.M. Nadaf, A. Simha, Optimal controller placement in Software Defined Networks (SDN) using a non-zero-sum game, in: Proc. IEEE 15th International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM), IEEE, Jun. 2014, pp. 1–6.

[32] A. Sallahi, M. St-Hilaire, Optimal model for the controller placement problem in software defined networks, IEEE Commun. Lett. 19 (1) (Jan. 2015) 30–33.

[33] A. Sallahi, M. St-Hilaire, Expansion model for the controller placement problem in software defined networks, IEEE Commun. Lett. 21 (2) (Feb. 2017) 274–277.

[34] A. Farshin, S. Sharifian, A chaotic grey wolf controller allocator for Software defined mobile network (SDMN) for 5th generation of cloud-based cellular systems (5G), Comput. Commun. 108 (Aug. 2017) 94–109.

[35] M. He, A. Basta, A. Blenk, W. Kellerer, Modeling flow setup time for controller placement in SDN: Evaluation for dynamic flows, in: Proc. 2017 IEEE International Conference on Communications (ICC), IEEE, May 2017, pp. 1–7.

[36] S. Muhizi, G. Shamshin, A. Muthanna, R. Kirichek, A. Vladyko, A. Koucheryavy, Analysis and performance evaluation of SDN queue model, in: Proc. International Conference on Wired/Wireless Internet Communication, Springer, Cham, Jun 2017, pp. 26–37.

[37] W. Kellerer, A. Basta, P. Babarczi, A. Blenk, M. He, M. Klügel, A.M. Alba, How to measure network flexibility? A proposal for evaluating softwarized networks, IEEE Commun. Mag. (2018).

[38] C. Gao, H. Wang, F. Zhu, L. Zhai, S. Yi, A particle swarm optimization algorithm for controller placement problem in software defined network, in: Proc. International Conference on Algorithms and Architectures for Parallel Processing, Springer, Cham, 2015, pp. 44–54.

[39] S. Ahmad, M. Mafarja, H. Faris, I. Aljarah, Feature Selection using Salp Swarm Algorithm with Chaos, John Wiley & Sons, 2018.

[40] E. Sakic, W. Kellerer, Response time and availability study of RAFT consensus in distributed SDN control Plane, IEEE Trans. Netw. Serv. Manage. 15 (1) (Mar. 2018) 304–318.

[41] A. Nair, M.J. Jacob, A. Krishnamoorthy, The multi server M/M/(s, S) queueing inventory system, Ann. Oper. Res. 233 (2015) 321–333.

[42] P. Franses, R. Legerstee, R. Paap, Estimating loss functions of experts, Appl. Econ. 49 (4) (Jan. 2017) 386–396.

[43] S. Mirjalili, A.H. Gandomi, S.Z. Mirjalili, S. Saremi, H. Faris, S.M. Mirjalili, Salp swarm algorithm: a bio-inspired optimizer for engineering design problems, Adv. Eng. Softw. 114 (Dec. 2017) 163–191.

[44] K.L. Du, M.N.S. Swamy, Particle swarm optimization, in: Search and Optimization by Metaheuristics, Birkhäuser, Cham, 2016, pp. 153–173.

[45] Y. Zhang, S. Wang, G. Ji, A comprehensive survey on particle swarm optimization algorithm and its applications, Math. Prob. Eng. (2015).

[46] G.I. Sayed, G. Khoriba, M.H. Haggag, A novel chaotic salp swarm algorithm for global optimization and feature selection, Appl. Intell. (2018) 1–20.

[47] A.T. Azar, S. Vaidyanathan (Eds.), Advances in Chaos Theory and Intelligent Control, Springer, 2016.

[48] C.H. Skiadas, C. Skiadas (Eds.), Handbook of Applications of Chaos Theory, CRC Press, 2017.

[49] A. Kaveh, Advances in Metaheuristic Algorithms for Optimal Design of Structures, Springer, 2016.

[50] R.M. May, Simple mathematical models with very complicated dynamics, Nature 261 (5560) (Jun 1976) 459.

[51] S. Saremi, S. Mirjalili, A. Lewis, Biogeography-based optimisation with chaos, Neural Comput. Appl. 25 (5) (Oct. 2014) 1077–1097.

[52] Internet Topology Zoo, Accessed on: May. 10, 2018 [Online] Available: http://www.topology-zoo.org/index.html.

[53] D.R. Richardson, J. Cormie, C.M. Carthaigh, B.W. Redman, Point of presence management in request routing, U.S. Patent 8,676,918, Mar. 2014.

[54] T. Y. Cheng, M. Wang, X. Jia, QoS-Guaranteed controller placement in SDN, in: Proc. 2015 IEEE Global Communications Conference (GLOBECOM), IEEE, Dec. 2015, pp. 1–6.

[55] A.A. Ateya, A. Muthanna, I. Gudkova, A. Abuarqoub, A. Vybornova, A. Koucheryavy, Development of intelligent core network for tactile internet and future smart systems, J. Sensor Actuat. Netw. 7 (1) (Jan. 2018).

[56] Y. Zhou, Y. Wang, J. Yu, J. Ba, S. Zhang, "Load balancing for multiple controllers in SDN based on switches group, in: Proc. 19th Asia-Pacific Network Operations and Management Symposium (APNOMS), IEEE, 2017, pp. 227–230.

[57] D. Bratton, J. Kennedy, Defining a standard for particle swarm optimization, in: Proc. Swarm Intelligence Symposium, SIS 2007, IEEE, 2007, pp. 120–127.

[58] H. Faris, I. Aljarah, M.A. Al-Betar, S. Mirjalili, Grey wolf optimizer: a review of recent variants and applications, Neural Comput. Appl. (2017) 1–23.

Please cite this article as: A. A. Ateya, A. Muthanna, A. Vybornova et al., Chaotic salp swarm algorithm for SDN multi-controller networks, Engineering Science and Technology, an International Journal, https://doi.org/10.1016/j.jestch.2018.12.015

Abdelhamied A. Ateya received the BSc. and MSc. in Electrical Engineering from Zagazig University, Egypt, in 2010 and 2014, respectively. He is working as an Assistant Lecturer at the Faculty of Engineering, Zagazig University. He is currently a PhD student at St. Petersburg State University of Telecommunication, in St. Petersburg, Russia. Mr. Ateya's main area of research is wireless communication and currently, he is working on5G systems and future Tactile Internet.

Ammar Muthanna is an Associate Professor at the Department of Telecommunication networks and Head of SDN Laboratory (sdnlab.ru), Saint - Petersburg State University of Telecommunications, Russia. He received his B.Sc. (2009), M.Sc. (2011) and as well as Ph.D. (2016) degrees from Saint - Petersburg State University of Telecommunications. In 2012/2013 he took part in Erasmus student program in the University of Ljubljana in Faculty of Electrical engineering. Area of research: wireless communications, 4G/5G cellular systems, IoT applications, and software defined networking.

Anastasia Vybornova graduated from the Bonch-Bruevich State University of Telecommunications, St. Petersburg, Russia in 2009, then earned a Candidate of Science (PhD) degree in the same university in 2014 (research field – wireless sensor networks). She worked as an engineer and product manager in the transport and telecommunication companies of St. Petersburg, then in 2015 Dr. Vybornova joint the Department of Communication Networks and data transmission of the Bonch-Bruevich State University of Telecommunications (St. Petersburg, Russia) as an Assistant Professor, then became an Associate Professor. Research interests: Internet of Things, Swarm intelligence, Tactile Internet.

Abdelrahman Abuarqoub is an Assistant Professor in Computer Science, Dean of the Faculty of Information Technology at the Middle East University of Jordan. He received his PhD in Computer Science from the Manchester Metropolitan University in 2014, his MSc (Distinction) in Data Telecommunications and Networks from the University of Salford in 2011, his BSc in Computer Networks Systems from Applied Science University/Jordan in 2009. His teaching experience includes undergraduate and postgraduate courses of Computer Networks, Network Administration and Management, Distributed Computing, and Information Security.

His research interests are in distributed algorithms, communication, and cross-layered solutions to Wireless Sensor Networks. His central research focuses on optimising performance and energy consumption in ad-hoc wireless and mobile networks; resource allocation data collection and routing; and distributed storage. He also has research interest in ubiquitous and mobile computing, specifically in Internet of Things. He has been researching and publishing work that focuses on developing solutions for Information Extraction form mobile Wireless Sensor Networks. He has been reviewer for leading journals (IEEE TSUSC, IEEE Access, MTAP-Springer, JOMS-Springer). His research work is published in various renowned journals and magazines of Elsevier, IEEE, Springer, and conference proceedings of IEEE and ACM.

Abeer D. Algarni received the B.Sc. (Hons.) in Computer Science from King Saud University, Riyadh, Saudi Arabia in 2007. She received the M.Sc and Ph.D. degrees from School of Engineering and Computer Sciences, Durham University, United Kingdom in 2010 and 2015, respectively. She worked as an assistant professor at College of Computer and Information Sciences at Princess Nourah Bent Abdulrahman University since 2008 up till now. Her current research interests include networking and communication systems, digital image processing, digital communications and cyber security.

Yevgeni Koucheryavy is a Full Professor at the Laboratory of Electronics and Communications Engineering of Tampere University of Technology (TUT), Finland. He received his Ph.D. degree (2004) from TUT. His current research interests include various aspects in heterogeneous wireless communication networks and systems, the Internet of Things and its standardization, as well as nanocommunications. He is Associate Technical Editor of IEEE Communications Magazine and Editor of IEEE Communications Surveys and Tutorials.

Andrey Koucheryavy was born in Leningrad, USSR on 02.02.1952. After graduation from Leningrad University of Telecommunication in 1974 he joined Telecommunication Research Institute LONIIS, where he was working up to October 2003 (from 1986 up to 2003 as the First Deputy Director). He obtained Ph.D. and Dr.Sc. in 1982 and 1994 respectively. Since 1998, A.Koucheryavy is a professor at St. Petersburg State University of Telecommunication (SUT). He is a Chaired professor of the department "Telecommunication Networks and Data Transmission" from 2011. He is honorary member of A.S.Popov's society. Prof. A.Koucheryavy is the Chairman of Study Group 11 ITU-T (Study periods 2017-2020). His scientific areas of interest are network planning, teletraffic theory, IoT and its enablers.

Please cite this article as: A. A. Ateya, A. Muthanna, A. Vybornova et al., Chaotic salp swarm algorithm for SDN multi-controller networks, Engineering Science and Technology, an International Journal, https://doi.org/10.1016/j.jestch.2018.12.015