Journal of Artificial Intelligence and Data Mining (JAIDM), Vol. 9, No. 3, 2021, 309-320.

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Research paper

An Efficient Approach to Solve Software-defined Networks based Virtual Machines Placement Problem using Moth-Flame Optimization in the Cloud Computing Environment

Amir hossein Safari-Bavil[1], Sam Jabbehdari[1*] and Mostafa Ghobaei-Arani[3]

1. Department of Electrical and Computer Engineering, North Tehran Branch, Islamic Azad University, Tehran, Iran.
2. Department of Computer Engineering, Qom Branch, Islamic Azad University, Qom, Iran.

Article Info

Article History:
Received 28 May 2020
Revised 26 January 2021
Accepted 21 April 2021

DOI:10.22044/jadm.2021.9737.2106

Keywords:
Cloud Computing, Virtual Machine Placement, Software-Defined Networks, Moth-Flame Algorithm.

**Corresponding author: S_Jabbehdari@iau-tnb.ac.ir(S. Jabbehdari).*

Abstract

Generally, the issue of quality assurance is a specific assurance in the computer networks. The conventional computer networks with hierarchical structures that are used in organizations are formed using some nodes of Ethernet switches within a tree structure. Open flow is one of the main fundamental protocols of software-defined networks that provides a direct access and changes in the program of sending the network equipment such as switches and routers, physically and virtually. The lack of an open interface in the data sending program has led to the advent of integrated and close equipment similar to CPU in the current networks. In this work, we suggest a solution to reduce the traffic using a correct placement of virtual machines, while their security is maintained. The proposed solution is based on the moth-flame optimization, which is evaluated. The results obtained indicate the priority of the proposed method.

1. Introduction

Generally, the issue of quality assurance is a specific assurance in the computer networks. Nowadays, most of the organizations, companies, and administrations use the computer networks due to their availability. However, the lack of a serious and new change in the computer networks has made the users of these networks to customize them and use new capacities. Moreover, these users are not willing to add new structures and pay more attention to the hardware [1].
Disconformity between the market and network capacities would lead to a deviation in the information technology. According to the given explanations about the server and communicative companies in order to prevent from recession, the software-defined architecture has been introduced, and its relevant standards have been developed. The idea of software-defined networks (SDNs) is not a new one as this idea has been existing for a decade, adding new parts to it. It should be noted that SDNs could still be discussed in research scopes. Straum has defined SDN as a method to separate the data and control the performances of the routers and other infrastructures of the second layer of ordinary networks using a programming interface [2] [3].
SDNs can affect the ideal placement by changing the traffic circulation throughout the network. Traffic may be done by the hypervisor itself instead of passing through a physical router for crossing between the segments of layer 2. This means that rather than placing two virtual machines (VMs), which have a strong connection between the boundaries of layer 2 close to each other and a router, they should be placed as shown in Figure 1, presented by the hypervisors, not physical switches [4] [5].
As the quality of the provided services in SDNs is a specific necessity in presenting the goals and services for the users, assurance of such a quality is a big challenge. Accordingly, the distributed protocols should be used. Open flow is one of the main fundamental protocols of SDNs that provides a direct access and changes in the

program of sending the network equipment such as the switches and routers, physically and virtually. The lack of an open interface in the data sending program has led to the advent of integrated and close equipment similar to CPU in the current networks. There is not any other standard protocol in network performing the tasks of the open flow protocol [6]. Hence, in this work, we propose a solution to reduce traffic using a correct placement of VMs, while their security is maintained. In this research work, the moth-flame algorithm was used to select the best place for VMs trying to consider them with security and some other parameters such as the hardware values of machine and traffic. Achievements of this work include a correct placement of VMs in order to reduce traffic, and propose a solution for placement of machines by maintaining the security level of VMs to protect the security of the whole system and reduce the security problems.

This work has been organized in a way that the background knowledge is explained to understand the proposed method, and the fundamental concepts are discussed. Then the relevant studies will be reviewed, and then the proposed method will be introduced, evaluated, and compared with the other methods. At the end, the conclusions will be expressed and some recommendations will be presented.

2. Technical Work Preparation

2.1. VM Placement

The cloud service providers are becoming larger and numerous; meanwhile, the data centers that provide hosting services tend to reduce their costs. The clouds run a big part of their computational workloads in the form of VMs on top of physical machines (PMs) or hosts that have an installed hypervisor. These hosts are then connected using a network that links the protocol stack of TCP/IP to several layers of physical switches and routers. An effective placement of VMs on the host leads to cost reduction due to a more useful application of resources and reduction in the overall demand.

SDNs change the transporting method through network by providing both the switching and routing services. In the conventional model, the data center of traffic travels through the following three layers of switching infrastructure:

- An edge layer that connects directly to all the hosts.
- A distribution layer that brings the edge switches together.
- A core layer that provides the routing services and connectivity between the distribution switches.

Today, SDNs are used in large data centers and cloud environments in order to facilitate the network management and its data flow [6]. This technology explains how the data flowing within the intermediate tools of network such as router is determined through a central server. To this end, the layers that control and manage the intermediate functions of network are separated from the data transfer layer, and the management layer can control some of the network tools such as the provincial routers based on its comprehensive view of online situation of network.

THE VMS' security is one of the significant and effective parameters in case of SDNs. On the other hand, the VM placement is an influential factor in the traffic of SDNs. Hence, a secure VM placement with a low traffic is an important issue that has been discussed in paper [6], and is called Software-defined Network based Virtual Machines Placement (SDNVMP).

In paper [7], the VM placement has been examined comprehensively, and then more than 12 open issues have been discussed, which can be studied in details. Among these options, the research question is based on the consuming energy reduction. Hence, this work was conducted in order to find a solution for SDNVMP (expressed in paper [6]) based on the moth-flame algorithm to promote the solution presented in paper [6], reduce the traffic, and maintain the security.

1.1.1. Pros and Cons of SDNVMP

The most important advantages of the VM displacement include:

- The utilization of SDNVMP can lead to the traffic reduction, while VMs that tend to create more connection with each other are placed close to each other in a way that the traffic does not travel to the upper layer so that the traffic is not transferred to the upper layer, and so the traffic load will be reduced.

- Reduction in the hardware as the traffic will not travel to the upper layer using SDNVMP; in this case, some hardware such as the routers and switches are less required since the routing of the lower layers is done by a correct placement of VMs, and the traffic is reduced.

- The most important disadvantage of VMP is the possible reduction in the security level because VMs with different security levels may be placed close to each other, which reduces the traffic but the security may be under question.

1.2. Open Flow Protocol

The open flow protocol is now the most acceptable and applicable programming interface for SDNs in the world that provides three information sources for the operating systems of the network, including:

- When the port or linking situation is changing, a message is sent to the controller by the forwarding device.
- Flowing statistics created by the forwarding devices are collected by the controller.
- Input packages into the forwarding devices, which are not matched with none of the conformity rules of the flow tables' records. These informational channels are highly important for providing the network operating systems with the flow-based data.

Although open flow is the most popular programming interface for SDN, it is not the only one, and some other interfaces can be named including ForCES, OVSDB, OpFlex, ROFL, HAL, and PAD [16][17].

2.3. Moth-flame Algorithm

The moth-flame optimization (MFO) algorithm, which is also called the candle and butterfly algorithm, is one of the heuristic and optimization algorithms that finds the solution based on the behaviors of the moths flying around the flame or fire. This algorithm was introduced in 2015 by Seyyed Ali Mirjalili, who wrote a paper entitled "Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm" that was published in the Journal of Knowledge-based Systems. The MFO algorithm is also known as the moth-fire algorithm, flame algorithm, and candle-butterfly algorithm. This algorithm is a novel exploratory model inspired by the nature and behavior of butterflies and their interest in the flame of fire. The inspiring factor in this optimizer is the navigation method used by moths in nature, which is called the transverse orientation. The moths can fly at night and travel to a long distance by keeping a fixed angle with respect to the moon. However, these fantasy insects are trapped in a useless and deadly spiral path around the artificial flames [18].

Optimization is defined as a process in which the best solution(s) for a certain problem are found. Increasing the complexity of problems in the recent decades has made it essential to find new optimization techniques. The mathematical optimization methods had been the only tools for the optimization of problems before proposing the exploratory optimization methods. A majority of the mathematical optimization methods suffer from the local optimums. The meta-heuristic algorithms start with a series of initial solutions, and become closer to the solution after the next iterations. The genetic algorithm, evolutionary differential algorithm, ant colony algorithm, bee colony algorithm, firefly algorithm, gray wolf algorithm, and whaling algorithm are the most popular algorithms in this case [18].

The moth-flame algorithm uses the behavior of these butterflies mathematically for optimization. The MFO algorithm is similar to the other known algorithms inspired by the nature, and the statistical results of standard functions indicate that this algorithm can achieve hopeful and competitive results.

Moths are fancy insects, and are highly similar to the family of butterflies. Basically, there are over 160,000 various species of this insect in the nature. They have two main milestones in their lifetime: larvae and adult. The larvae are converted to moth by cocoons. The most interesting fact about moths is their special navigation methods at night. They have been evolved to fly in night using the moon light. They utilize a mechanism called the transverse orientation for navigation. In this method, a moth flies by maintaining a fixed angle with respect to the moon, a very effective mechanism for travelling long distances in a straight path [18].

$$M = \begin{bmatrix} m_{1,1} & m_{1,2} & \dots & \dots & m_{1,d} \\ m_{2,1} & m_{2,2} & \dots & \dots & m_{2,d} \\ \dots & \dots & \dots & \dots & \dots \\ m_{n,1} & m_{n,2} & \dots & \dots & m_{n,d} \end{bmatrix} \rightarrow OM = \begin{bmatrix} OM_1 \\ OM_2 \\ \dots \\ OM_n \end{bmatrix} \quad (1)$$

$$F = \begin{bmatrix} F_{1,1} & F_{1,2} & \dots & \dots & F_{1,d} \\ F_{2,1} & F_{2,2} & \dots & \dots & F_{2,d} \\ \dots & \dots & \dots & \dots & \dots \\ F_{n,1} & F_{n,2} & \dots & \dots & F_{n,d} \end{bmatrix} \rightarrow OF = \begin{bmatrix} OF_1 \\ OF_2 \\ \dots \\ OF_n \end{bmatrix}$$

In the MFO algorithm, it is assumed that the candidate solutions are moths, and the problem's variables are the position of moths in the space. Therefore, the moths can fly in a 1-D, 2-D or 3-D space with changing their position vectors. Since the MFO algorithm is a population-based algorithm, the set of moths can be illustrated in a matrix (for instance, matrix M). An array (OM) also exists for all of moths to store the fitness values. Another key component in this algorithm is a matrix similar to the matrix of moths, which is flame or flag matrix (F), and an array of OF is used to store the fitness function value [18].

2. Review of Relevant Studies

Nowadays, services are increasing, and SDNs have been created to reduce the traffic load. The article [6] published in 2017 has proposed a solution that is based on the idea of grouping virtual machines because a correct placement of VM leads to a substantial reduction in the network traffic. On the other hand, the VMP position can affect the security of VMs since they have different security levels. Anderson *et al*. [7] have proposed a solution in which VMs are grouped in way that not only VMs with similar security levels are placed but also the network traffic is decreased.

Zhang *et al*. [8] have presented a constraint programming-based virtual cloud resource allocation (CP-VCRA) model that can meet the service quality requirements, and reduce the resources of consumption cost. The authors have considered the performance of the application programs' goals and different kinds of workloads in their study.

Dupont *et al*. [9] have introduced a flexible and expandable framework based on the re-scheduling problem of virtual machines (VSRP) that is used for the allocation of resources in data centers based on an energy aware method considering SLA limits for VMP. This approach allows the user to find the SLA constraints and reduce the energy consumption.

Dong *et al*. [10] have applied a few constraints such as the network connectivity capacity and physical machine size on the VM scheduling using the two-step VMS algorithm. At the first step, they combined the best-fit exploratory algorithm from the bin packing algorithm with cluster-minimal hierarchical clustering. This minimizes the number of activated physical machines and prevents from the traffic in the network using the MLU optimization (maximum link productivity), and this goal can be achieved by modeling the network traffic as a QAP (quadratic assignment problem). At the second step, the allocated VMs are re-optimized.

Song *et al*. [11] have formulated a dynamic resource allocation algorithm based on the bin packing, in which the number of performing and active servers is optimized. They created a minor change in the possible online bin packing [12], and named it as VISB (bin packing with variable item size). They implemented this algorithm using the extensible detector-based simulation, and then compared it with three known algorithms for server integration including Black Box & Gray Box [13], VectorDot algorithm [14], and Offline Bin Packing Algorithm [15]. In fact, the VISBP core is the capability of this algorithm in considering the size change of an item (VM) at the running time. This "changing" process supports the dynamic and demand-based allocation. VISBP performs well in terms of load balance and hotspot detection but it violates the service level to some extent, so it requires improving the VM to physical machine ratio.

Li *et al*. [25] have proposed adaptive hyper-sphere (AdaHS), an adaptive incremental classifier, and its kernelized version: Nys-AdaHS. The classifier incorporates competitive training with a border zone. With adaptive hidden layer and tunable radii of hyper-spheres, AdaHS has a strong capability of local learning like instance-based algorithms but free from slow searching speed and excessive memory consumption.

Kou *et al*. [26] have proposed an approach to resolve disagreements among the multiple criteria decision-making (MCDM) methods based on the Spearman's rank correlation coefficient. The experimental results prove that the proposed approach can resolve the conflicting MCDM rankings and reach an agreement among different MCDM methods.

Various types of VM placement algorithms have been proposed in the literature [20]. Such algorithms largely aim to minimize the amount of unused CPU cycles and RAM on the host. One solution technique is to solve the classic bin packing problem, in which the variably-sized objects must be put into a minimum number of fixed-sized bins. Another technique is to treat the resources required by a particular VM as dynamic, and use the stochastic integer programming in order to make intelligent placement decisions [21]. Some techniques consider a single heuristic at a time, while the others combine several limitations into a single value to determine the appropriate placement. The First Fit (FF) algorithm proposes placing a VM in the first host that has an available capacity to host it, while adding a new host if no such host exists [21]

Wang *et al*. [22] have proposed the energy-efficient and QoS-aware virtual machine placement algorithm, namely EQVMP. In EQVMP, VMs are placed in a scheme designed to minimize the network traffic between VMs. EQVMP is very similar to our proposed algorithm; however, our work differs from EQVMP in that our algorithm allows a variable number of VMs to be assigned to each cluster based on the network closeness and VM resource sizing at the same time.

Table 1 reports a comparison between different approaches considered in this work.

Table 1. A comparison between previous studies.

Author(s)	Description
Anderson *et al.* [7]	VMs are grouped in a way that not only VMs with similar security level are placed but also the network traffic is reduced.
Zhang *et al.* [8]	Service quality was examined in this research work without considering security of VMs for grouping.
Dupont et al. [9]	The main goal in this research work is energy reduction, trying to maintain the security while many of machines have security problems since ports and machines are not grouped.
Dong *et al.* [10]	In this research work, grouping was done well in terms of service quality but security and energy did not receive any attention.
Song *et al.* [11]	Like the researches [12], [13], and [14], energy has been considered in this work with a specific attention toward service quality, while VM security has not been considered in this approach so some machines with different security levels may be placed close to each other.

3. Proposed Method

The proposed method for VM placement using the moth-flame algorithm has been investigated in this section.

4. Objective Function and Constraints

The moth-flame algorithm-based placement method makes it possible to apply different objective functions. Moreover, the introduced constraints in this section include the probable change and development.

The closer the highly used VMs to each other, the lower the network traffic will be. Meanwhile, security should be maintained in a way that VMs with high security levels should not be placed close to the machines with highly low security levels. In general, a parameter should be considered herein as these two objectives can be integrated. Consider w for the importance degree for network traffic and s for the importance degree of network security. Thus we have:

$$F = \alpha s + (1-\alpha)w \qquad (2)$$

Here, the coefficient is a real number, and the formula shows that a number should be given to α in order to keep a balance between the security maintenance and the network traffic reduction, and find which one is more important. F is the final output. Therefore, the α coefficient should be determined at first.

To this end, some constraints exist:

$$Minimizing: \sum_i Y_i - k \times \qquad (3)$$

$$\sqrt{\sum_i \left(\left(\frac{\sum_j \text{Mem\_}usage_j \times x_{ij}}{\text{Mem\_cap}_i}\right)^2 + \left(\frac{\sum_j cpu\_usage_j \times x_{ij}}{cpu\_\text{cap}_i}\right)^2\right)}$$

$$\sum_i X_{ij} = 1 \rightarrow \forall j$$

$$Y_i \geq X_{ij} \rightarrow \forall i, j$$

$$\sum_i X_{ij} = Y_i \rightarrow \forall j$$

$$Y_i \times cpu\_cap_i \geq \sum_i (X_{ij} \times cpu\_usage_j) \rightarrow \forall i, j$$

$$Y_i \times Mem\_cap_i \geq \sum_i (X_{ij} \times Mem\_usage_j) \rightarrow \forall i, j$$

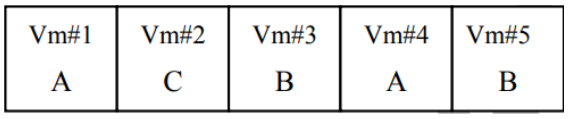
$$F_i \times F\_cap_i \geq \sum_i (X_{ij} \times F\_usage_j) \rightarrow \forall i, j$$

The symbols used in these constraints are described in Table 2.

Table 2. Symbols used in the constraints.

symbol	Description
j	VMs' index
i	PMs' index
X_{ij}	Binary variable; it equals 1 if VM j corresponds to PM i
Y_i	Binary variable; it equal 1 if PM i is used
Mem_cap_i	Initial memory capacity of PM i
cpu_cap_i	Initial CPU capacity of PM i
F_cap_i	Initial F value related to VM i
Mem_usage_j	Memory requested by VM j
$\text{cpu\_}usage_j$	Requested CPU by VM j
$\text{F\_}usage_j$	Requested F value by VM j

The responses have been coded with the method used in the proposed method in paper [19]. In order to describe this method, Figure 1 can be considered. This figure illustrates a proposed solution for the placement of 5 VMs.



Vm#1	Vm#2	Vm#3	Vm#4	Vm#5
A	C	B	A	B

Figure 1. A solution for VMP.

As it can be seen in Figure 1, each solution is an array whose elements are equal to the number of VMs requested for placement. The index of each element of this array indicates the number of VM and the corresponding value is the PM index considered for running that VM. For instance, in Figure 1, PM A corresponds to two VMs 1 and 4 running, PM C for VM 2 running, and VMs 3 and 5 corresponding to PM B.

4.1. Initialization

In order to generate the initial solutions and to calculate the fit values for each solution, any stochastic distribution can be used. The following method was used as a suitable one in this work:

Table 3. Calculating fitness value.

```
for i = 1:n
    for j = 1:d
        M(j,j)=(ub(i)-lb(i))*rand()+lb(i);
    end
end
OM = FitnessFunction(M);
```

where ub is the upper bound and lb is the lower bound of variable 1. Thus the n×d initial solutions can be generated. After initialization, the fitness of each solution can be calculated; therefore, the OM value indicates the fitness function of matrix M or matrix of moths. The proposed MFO-based VMP algorithm has introduced a modern method for the moths move and generation of new solutions. A new concept- named agent- has been added to the MFO algorithm. This concept will be introduced, and then its display will be presented in this section. In this research work, d indicates 9 dimensions and n shows the number of moths that are VMs here. Flames are also shareware machines in which VMs are placed. In this work, d equals 9 because we have 8 constraints and one objective function. Therefore, the general solution of this work includes 9 dimensions.

4.2. Agent and How to Displays It

As it was discussed earlier, each solution is an array whose size equals the number of VMs required for placement. Considering the introduced objective function in the previous section, the required items for fitting a solution include the number of PMs used, utilization rate of each PM, and security level of each machine existing in the solution. Hence, a structure is introduced in order to collect the mentioned data.

The agent of each solution is an array whose size is equal to the number of physical and virtual machines used in a certain solution. The value of each element is the index of each used PM. The index of such PM is used to compare its utilization rate.

Consider the solution shown in Figure 1 again for more clarification. In total, this solution uses the three PMs of A, B, and C. Assume that the traffic for these three machines equals 70, 40, and 30, respectively. Accordingly, the agent of this solution can be shown as Figure 2.

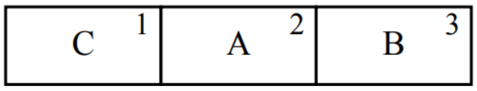


Figure 2. Agent of the solution shown in Figure 1.

4.3. Problem formulation

The MFO algorithm was used in this research work in order to find the correct place for VM. To this end, the MFO algorithm has been applied in this research work. According to the pre-determined parameters of this algorithm, the best place should be chosen for VM, and also the security level should be maintained while considering the computational overload and the required resources. To this end, a function is presented in which the best place is determined for machine, and its general algorithm has been illustrated in Figure 3.

```
Algorithm 1: choosing best place
1: Begin
2:    for each (Time interval $\Delta t$ in execution time) do
3:        inputs = CheckStatus()
4:        allocationMap = MFO(inputs)
5:        Allocate VMs to right places
6:    end for each
7: End
```

Figure 3. Algorithm of choosing best place for VM.

MFO chooses the best place for machines in each iteration. In this algorithm, the required machine receives the system situation considering the CheckStatus function, and then the best place is chosen for that machine based on the pre-determined parameters. As the pseudo-code shows, the best hardware machine is determined for placement at the first step for each machine at each time interval using the MFO algorithm. The system situation is determined with respect to Figure 4. This means that the status of machines is determined, and then the best place is chosen for the input machine until the best placement is done for machine.

```
Algorithm 2: Monitoring status of system
1: inputs: VMS
2:outputs: status of VMs
3:Begin
2:    for each (vm in inputs) do
3:        outputs = Monitor ⟦(Mem_usag, CPU_usage, F, N)
4:    end for each
5: End
```

Figure 4. Algorithm of examining the status of system.

As it can be seen, the machines are checked at each iteration, and their security level, CPU capacity, memory capacity, and number of existing machines are determined so all the machines are under the supervision; then the MFO algorithm is used in order to find the best place for the considered input machine.

In this algorithm, moths are the search agents, and flames are the best places chosen for the machines at the time. In general, the MFO algorithm includes a set of moths in the following form:

$$M = \begin{bmatrix} M_{1,1} & \dots & M_{1,d} \\ \dots & \dots & \dots \\ M_{n,1} & \dots & M_{n,d} \end{bmatrix} \quad (3)$$

In this matrix, M_{ij} indicates the moths, d is the number of all the input parameters, and n is the total number of used moths. In fact, M_{ij} indicates VM and n indicates the total number of input VMs. In this work, d equals 9 dimensions; the first dimension relates to the objective function and the rest of dimensions correspond to the parameters mentioned in the constraint part. Various solutions are obtained in the algorithm; the fitness value (MO) of this solution can be illustrated as follows:

$$OM = [OM_1, OM_2, \dots, OM_n]^T \quad (4)$$

The flames that are the problem's solutions are shown in another matrix. There are various solutions for the problem since various places can be chosen for one machine but the best solution should be selected; hence, a matrix is designed for solutions in order to find the best solution at the end. The matrix of flames (Matrix F) is shown as follows:

$$F = \begin{bmatrix} F_{1,1} & \dots & F_{1,d} \\ \dots & \dots & \dots \\ F_{n,1} & \dots & F_{n,d} \end{bmatrix} \quad (5)$$

Similar to the matrix of moths, in this matrix, also d is the number of parameters and n is the total number of moths, and so n solutions should be achieved.

Matrices F and M are the dimensions. Since the fitness matrix was determined for Matrix M, the simila matrix (OF) should also be designed for F. It is obvious that OM and OF are both dimensions.

$$OF = [OF_1, OF_2, \dots, OF_n]^T \quad (6)$$

Generally, the MFO algorithm applies the three parameters of I, P, and T; I includes the initial inputs:

$$I : \phi \rightarrow \{M, OM\} \quad (7)$$

P refers to the motion function that causes movement of moths, and it will be explained that this motion is done in a spiral form.

$$P : M \rightarrow M \quad (8)$$

Function T is designed for the finishing condition of the algorithm; if this function is satisfied, a true value is obtained, meaning that the algorithm has finished, and the best solution has been obtained.

$$T : M \rightarrow \{true, false\} \quad (9)$$

4.4. Algorithm Iteration

After the initialization, the moths' motion or function P is run repetitively until the T performance (convergence limit to flame) returns to a true value. The P performance is the main performance that moves the moths in the search space. As it was mentioned earlier, this algorithm is inspired by a transverse orientation.

4.5. Updating Moths

The logarithmic spiral has been chosen as the main mechanism for updating moths in the proposed method for VMP using the moth-flame algorithm:

$$S(M_i, F_j) = \mathrm{D}_i \,.\, e^{bt} \,.\cos(2\pi t) + F_j \quad (10)$$

- Spiral's initial point should start from the moth.
- Spiral's final point should be the position of the flame.
- Fluctuation of the range of spiral should not exceed the search space.

Considering these points, a logarithmic spiral can be defined for the MFO algorithm, as follows:

$$D_i = |F_j - M_i| \quad (11)$$

In the logarithmic spiral equation, the spiral-flying path of moths can be simulated in the moth-flame algorithm. As it can be seen in the equation, the next position of moth is defined based on the flame. The t parameter in the spiral equation defines how much the next position of the moth should be close to the flame (t = -1 is the closest position to the flame, while t = 1 shows the farthest), which is finding the best value in this research work.

5. Evaluation

The implemented system in this research work includes the specifications reported in Table 4.

Table 4. Specifications of the implemented system.

Specification	Value
Operating system	Windows 10 64 bit
Ram	8G
Cpu	Corei7
HDD	256SSD

The two programming languages of Python and MATLAB have been used in this implementation; these two languages are popular and common in the implementation field, which have been employed in this research work. The implementation environments of mininet and Matlab R2017b have been used in this work.

In the simulation experiment, a single network switch is allowed to connect γ = 12 hosts together in addition to the necessary uplinks into the core network. The fixed number of hosts used for EQVMP is m = 20, which matches the average values observed in the proposed algorithm and FF. The total number of VMs is set to n = 1000. In this work, we demonstrated the performance results under two environments: simulated networks and real networks. Under the simulated network, we used randomly generated 100 networks based on the uniform distribution. The edge weights are given as the integer numbers based on the uniform distribution with the range of [1, 1000]. The results obtained are the average values of the given metrics based on 100 simulation realizations. Under the real networks, the packet captures from several data center environments were used in order to evaluate the performance of the algorithms. Two of these packet captures are taken from an earlier study on data center traffic [24], while the third one is the packet captures of a month of network traffic from a hosting provider conducted by the SimpleWeb project [23]. Each one of these datasets was pre-processed to build the network model using each IP address as a single vertex in the network representing a VM. An edge is created between two IP addresses when a packet is sent between the two addresses. This edge is weighted by the sum of the lengths of all packets going between the two addresses. Each vertex also has the source ports and protocols associated with it in the traffic.

In order to examine the proposed approach, this method was compared with the method presented in reference [6], and SDNVMP and analyses were carried out based on such a comparison in simulated networks.

The number of hosts used at the first step of simulation was equal to 40; however, this is the initial value, and the required value can be added to it, and 1000 VMs were considered to see the results in a better way.

In this research work, VMs have different security levels based on the following risks:

- Risk 4: a high number of 5 allocated ports at the threshold segment is changeable, while the VM with 5 or more ports is the most important one.
- Risk 3: some ports are specific such as the UDP/TCP 20, TCP 21, and TCP 23 ports, which do not have encrypted protocols.
- Risk 2: a high number of links in one virtual machine has been considered as a security state in this research work; this means a VM with a high number of links and a high-level popularity.
- Risk 1: low-risk machines that are considered as ordinary machines.

Therefore, four security levels have been chosen and classified (from 4 to 1), as mentioned above.

In this research work, some parameters have also been determined that indicate the capabilities of a specific solution:

- The number of applied hosts: the minimum number of hosts required to cover all VMs in a way that algorithm works without any problem. The fewer the number of this parameter, the better the situation will be; in this case, less resources are required, and energy consumption is reduced.
- Total network traffic for each host; the fewer the number of this parameter, the lesser the traffic load will be. Distribution of traffic on different hosts leads to a higher speed in the users' requests without any accident.
- Standard deviation of security risk that is calculated using formula (11). This parameter is used in order to see how each algorithm has done the categorization.

$$R_{SD} = \frac{\sum_{j=0}^{N_H} sd_j}{N_H} where: sd_j = \sqrt{\frac{\sum_{i=0}^{n}(r_{ij} - \overline{r_j})}{n-1}} \tag{12}$$

- Cumulative sum of edge weights ($\hat{C}$) is the average cumulative sum of the edge weights to measure the network traffic between hosts on different network switches. Lower is better, incurring less network traffic. $\hat{C}$ is obtained by:

$$\hat{C} = \sum_{v_i, v_j \in T_x, v_i \in h_x, v_j \in h_y, h_x \neq h_y} w'_{ij} + 2 \sum_{v_i \in h_x, v_j \in T_y, T_x \neq T_y} w'_{ij} \tag{13}$$

  Recall that w'_{ij} is a raw integer weight indicating the degree of network traffic flow. T_x refers to a set of hosts connected to the same network switch.

Figure 5 indicates that increasing the rate of α (that is the ratio between the security risk and the traffic) leads to a higher importance of the security risk. Obviously, with an increase in the importance of security level in VMP, the machines with similar security risks are put in one category, and this, in turn, leads to an increase in the number of required hosts. In this diagram of the proposed method, the number of hosts is a little greater than SDNVMP [6] at first because of considering the hardware specifications at the first step of the proposed method. Therefore, this specification indicates that more categorizations are done at the first step but then the number of hosts is slowly reduced due to use of the moth-flame method to select the proper hosts; therefore, α~1 in the proposed method indicates that fewer number of hosts are required compared to the SDNVMP approach, and this approves the better situation of the proposed method. In this case, the proposed algorithm has used fewer hosts with a higher accuracy considering the security levels, so it requires less resources compared to the SDNVMP approach.

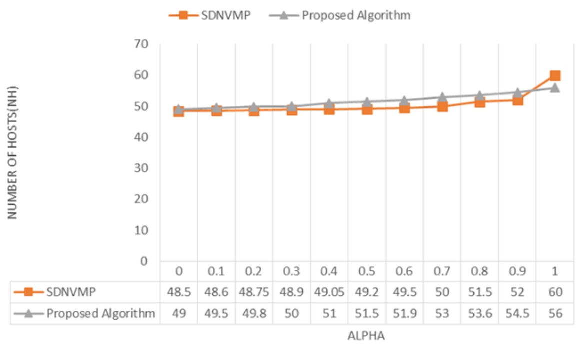


Figure 5. Number of hosts in various α values of the proposed algorithm and approach used in [6].

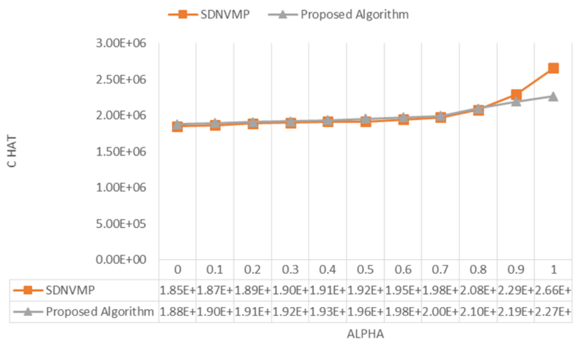


Figure 6. Total number of applied traffics considering alpha value of the proposed algorithm and SDNVMP.

Figure 6 indicates that an increase in the alpha value leads to increase in traffic, which is one of reasons for the increase in number of hosts while this value becomes less gradual in the proposed algorithm due to more suitable leveling. It can be concluded that a better categorization has been done by the proposed algorithm leading to a less traffic and an out-of-switch traffic; this means that traffic exists just in the relevant switch. It is obvious that smaller alpha values in the proposed algorithm have acted worsen compared to SDNVMP, and an increase in the alpha value has led to a better performance of the proposed algorithm compared to SDNVMP. Therefore, the proposed algorithm has performed better in the alpha values between 0.8 and 1, and this is the optimal value for the proposed algorithm as the better performance of the proposed method relates to this value range.

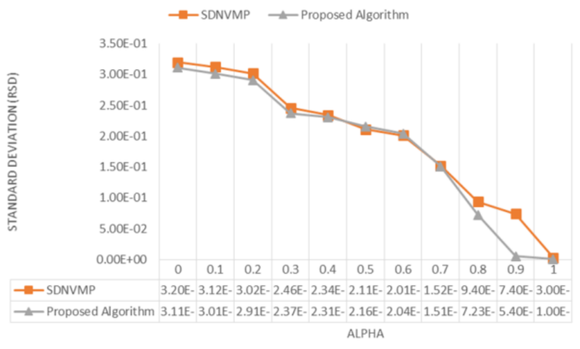


Figure 7. SD diagram considering alpha value of the proposed algorithm and SDNVMP.

According to Figure 7, the standard deviation of the proposed algorithm is smaller than the SDNVMP method due to the utilization of more efficient algorithm in categorizing these VMs on optimal hosts. In this case, the security level in the proposed algorithm is higher than the other approaches. It can be seen that an increase in the alpha value in the proposed algorithm has led to a better leveling, categorizing, and results due to more attention paid to the security leveling part. Therefore, better results have been achieved showing the superiority of the proposed method compared to the SDNVMP method.

For a better comparison of the proposed algorithm, the out algorithm has been compared with the FF method [21] and EQVMP [22] in the simulated and real networks.

5.1. Simulated Networks

Figures 8 to 10 show the performance comparison of the three VM placement algorithms, FF, EQVMP, and the proposed algorithm, in the simulated networks. Figure 8 shows the comparison of the three algorithms in terms of the average number of hosts, NH. Overall, FF and the proposed algorithm perform roughly the same, while EQVMP requires only a few more hosts. As α is close to 1, the proposed algorithm begins to trend upwards; when α = 1, substantially more hosts are required. This implies that using the proposed algorithm with a high α provides a less efficient host clustering.

Figure 9 shows the average cumulative sum of edge weights, $\hat{C}$, of the three algorithms. In Figure 9, the proposed algorithm significantly performs well compared to FF and EQVMP even when the VM risk is weighted above the network clustering up to α = 0.7. As α is close to 1, $\hat{C}$ approaches the performance of FF. This is because the network characteristics are no longer taken into account in this configuration. This implies that the performance of the network efficiency will be equivalent to that of FF, which is not traffic-aware.

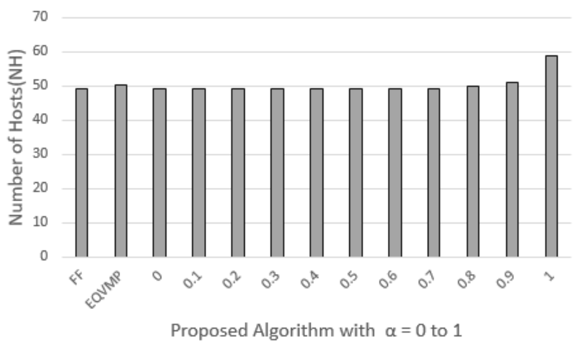


Figure 8. Comparison of the three algorithms in terms of the average number of hosts, NH in the simulated networks.

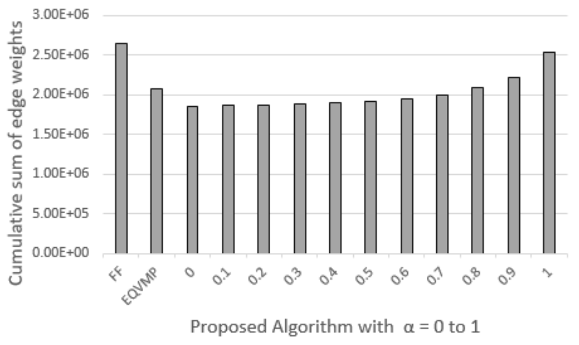


Figure 9. Average cumulative sum of edge weights of FF, EQVMP, and proposed algorithms in the simulated networks.

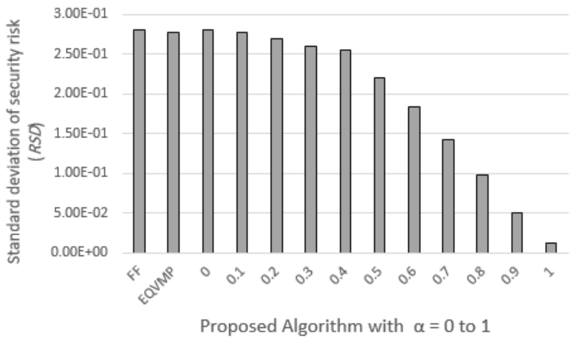
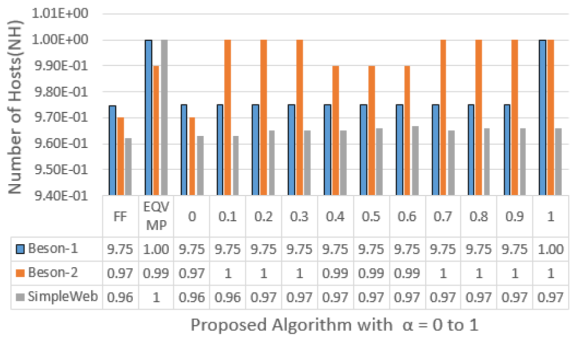


Figure 10. Average of the standard deviation of the security risk of FF, EQVMP, and proposed algorithms in the simulated networks.

Figure 10 shows the average of the standard deviation of the security risk, RSD, of each VM on a per-host basis for each placement algorithm. For α = 1, RSD is very close to 0, a significantly higher performance compared to those of FF and EQVMP or the proposed algorithm with lower α. However, as shown in Figures 8 and 9, a higher α is not always preferred in terms of the number of hosts required to handle the given VMs and network traffic generated from the placement. In order to achieve the conflicting goals of minimizing high network/resource cost and minimizing the VM security risk, we suggest choosing an optimal α (e.g. 0.6 in this case study), which can best balance between the imposed system requirements associated with these two goals.

5.2. Real Network

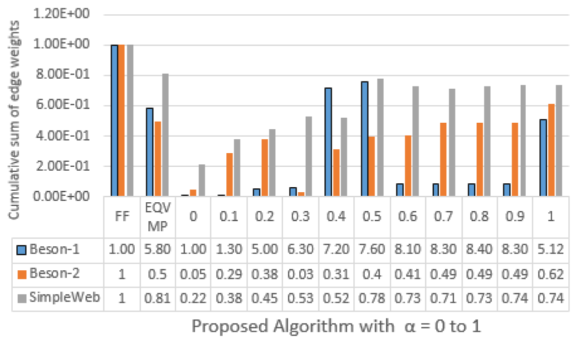
We also conducted the experiment on the real networks [24], [23] in order to investigate the performance of the three algorithms, as shown in Figures 11 to 13. Figure 11 shows the number of hosts used by the three algorithms. While EQVMP does outperform FF, the proposed algorithm provides a better placement when α = 0 and α = 0.5. Figure 12 shows the cumulative sum of edge weights, $\hat{C}$, of the three algorithms in the real networks.



	FF	EQVMP	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
🟦 Beson-1	9.75	1.00	9.75	9.75	9.75	9.75	9.75	9.75	9.75	9.75	9.75	9.75	1.00
🟧 Beson-2	0.97	0.99	0.97	1	1	1	0.99	0.99	0.99	1	1	1	1
▨ SimpleWeb	0.96	1	0.96	0.96	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97	0.97

Figure 11. Comparison of the three algorithms in terms of the average number of hosts, NH in the real networks.

Notice that EQVMP does indeed outperform FF but the proposed algorithm performs far better, particularly for lower α ≤ 0.7. As α is close to 1, the performance does slip to worse than EQVMP, as the grouping decisions are being made based on the security risk rather than the network communication.



	FF	EQVMP	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
🟦 Beson-1	1.00	5.80	1.00	1.30	5.00	6.30	7.20	7.60	8.10	8.30	8.40	8.30	5.12
🟧 Beson-2	1	0.5	0.05	0.29	0.38	0.03	0.31	0.4	0.41	0.49	0.49	0.49	0.62
▨ SimpleWeb	1	0.81	0.22	0.38	0.45	0.53	0.52	0.78	0.73	0.71	0.73	0.74	0.74

Figure 12. Average cumulative sum of edge weights of the FF, EQVMP, and proposed algorithms in the real networks.

Figure 13 shows the average standard deviation of the security risk, RSD, of each VM on a per-host basis for each placement algorithm on the three real networks. As α rises, RSD decreases; the effect is not nearly as obvious as in the random networks generated, and appears to be much less effective. This is due to the differences between the randomly generated risk measurements and those calculated for the actual network traffic, which generate different patterns of risk. Despite this behavior, the proposed algorithm still outperforms the other two algorithms, as they do not make security-aware decisions.

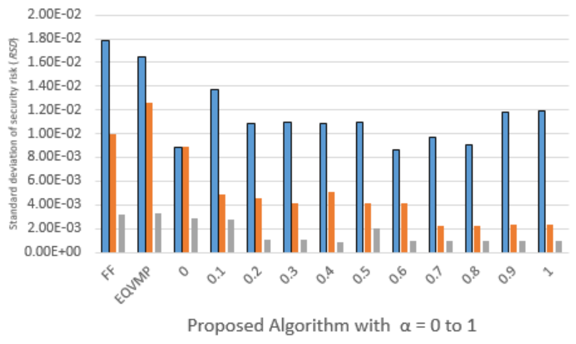


Figure 13. Average of the standard deviation of the security risk of the FF, EQVMP, and proposed algorithms in the real networks.

6. Conclusions

The VMs' security is one of the most influential and significant parameters involved in the field of software-defined networks. On the other hand, the VM placement is a highly important factor for the SDN traffic. Hence, a VM placement with a proper placement and a low traffic is an important issue that has been discussed in reference [6] under the title of SDNVMP (software-defined network virtual machine placement). The previous approaches did not examine this case; so it is a new subject without many solutions and approaches regarding it. This paper proposed a solution based on the moth-flame optimization algorithm in order to consider the best host for a VM. In the proposed algorithm, the presented approach not only considers the security risks but also classifies the risks based on the hardware specifications. The alpha parameter was used in this approach in order to indicate the importance level of the network traffic and security risk. All the implementation and conclusion processes are based on the alpha parameter. In this research work, the proposed algorithm was compared with the SDNVMP method that was similar to the proposed algorithm in this work. The results obtained indicated that the proposed algorithm could perform better than the approach used in reference [6] in terms of the classification and achievement.

In addition to the considered parameters in this research work, other parameters can be used in further studies. For instance, the amount of energy consumed in hosts was not considered, so it could be added to the alpha value as a new option.

References

[1] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Zomaya (2011). "A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems", pp. 47–111.

[2] Tajamolian, M., Ghasemzadeh, M. (2019). Analytical evaluation of an innovative decision-making algorithm for VM live migration. *Journal of AI and Data Mining*, vol. 7, no. 4, pp. 589-596. doi: 10.22044/jadm.2018.7178.1847.

[3] Mabhoot, N., Momeni, H. (2021). An Energy-aware Real-time Task Scheduling Approach in a Cloud Computing Environment. *Journal of AI and Data* Mining, (), -. doi: 10.22044/jadm.2021.10344.2171.

[4] Donyagard Vahed, N., Ghobaei-Arani, M., & Souri, A. (2019). Multiobjective virtual machine placement mechanisms using nature-inspired metaheuristic algorithms in cloud environments: A comprehensive review. *International Journal of Communication Systems*, 32(14), e4068.

[5] Masdari, M., Gharehpasha, S., Ghobaei-Arani, M., & Ghasemi, V. (2019). Bio-inspired virtual machine placement schemes in cloud computing environment: taxonomy, review, and future research directions. *Cluster Computing*, 1-31.

[6] J. Anderson and J.-H. Cho (2017). "Software Defined Network Based Virtual Machine Placement in Cloud Systems," in MILCOM 2017 IEEE Military Communications Conference (MILCOM), pp. 876–881.

[7] M.C. Silva Filho, C.C. Monteiro, P.R.M. Inácio, and M. M. Freire (2018). "Approaches for optimizing virtual machine placement and migration in cloud environments: A survey," J. Parallel Distrib Comput, vol. 111, pp. 222–250.

[8] L. Zhang, Y. Zhuang, and W. Zhu (2013). "Constraint Programming based Virtual Cloud Resources Allocation Model," Int. J. Hybrid Inf. Technol., vol. 6, no. 6, pp. 333–344.

[9] C. Dupont, T. Schulze, G. Giuliani, A. Somov, and F. Hermenier (2012). "An energy aware framework for virtual machine placement in cloud federated data centres," in Proceedings of the 3rd International Conference on Future Energy Systems Where Energy, Computing and Communication Meet- e-Energy'12, pp. 1–10.

[10] J. Dong, H. Wang, and S. Cheng (2015). "Energy-performance tradeoffs in IaaS cloud with virtual machine scheduling," China Commun., vol. 12, no. 2, pp. 155–166.

[11] W. Song, Z. Xiao, Q. Chen, and H. Luo (2014). “Adaptive Resource Provisioning for the Cloud using Online Bin Packing,” IEEE Trans. Comput., vol. 63, no. 11, pp. 2647–2660.

[12] G. Gambosi, A. Postiglione, and M. Talamo (2000). “Algorithms for the Relaxed Online Bin-Packing Model,” *SIAM J. Comput*., vol. 30, no. 5, pp. 1532–1551.

[13] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif (2009). “Sandpiper: Black-box and gray-box resource management for virtual machines,” Comput. Networks, vol. 53, no. 17, pp. 2923–2938.

[14] A. Singh, M. Korupolu, and D. Mohapatra (2008). “Server-storage virtualization: Integration and load balancing in data centers,” in 2008 SC-International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1–12.

[15] N. Bobroff, A. Kochut, and K. Beaty (2007). “Dynamic Placement of Virtual Machines for Managing SLA Violations,” in 2007 10th IFIP/IEEE International Symposium on Integrated Network Management, pp. 119–128.

[16] R. Enns. NETCONF Configuration Protocol. RFC 4741 (Proposed Standard), Dec. 2006. Obsoleted by RFC 6241.

[17] Brent Salisbury (2012). The Northbound API- a Big Little Problem, www.networkstatic.net.

[18] Seyedali Mirjalili (2015). “Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm”, Knowledge-based Systems 89, 228–249.

[19] S. Agrawal, S. Bose, and S. Sundarrajan (2009). “Grouping genetic algorithm for solving the server consolidation problem with conflicts,” Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation, pp. 1-8.

[20] Ghobaei-Arani, M., Rahmanian, A. A., Shamsi, M., & Rasouli-Kenari, A. (2018). A learning-based approach for virtual machine placement in cloud data centers. *International Journal of Communication Systems*, 31(8), e3537.

[21] R.K. Gupta and R. Pateriya (2019). “Survey on virtual machine placement techniques in cloud computing environment,” *International Journal on Cloud Computing: Services and Architecture (IJCCSA)*, Vol. 4, No. 4, pp. 1–7.

[22] S.-H. Wang, P.P.W. Huang, C.H.P. Wen, and L.-C. Wang (2020). “EQVMP: Energy-efficient and qos-aware virtual machine placement for software defined datacenter networks,” in IEEE International Conference on Information Networking.

[23] R.R.R. Barbosa, R. Sadre, A. Pras, and R.V.D. Meent (2010). “Simpleweb/university of twente traffic traces data repository,” Tech. Rep. [Online]. Available: http://eprints.eemcs.utwente.nl/17829/

[24] T. Benson, A. Akella, and D. Maltz (2010). “Network traffic characteristics of data centers in the wild,” in ACM Proceedings of the 10th ACM SIGCOMM conference on Internet measurement, pp. 267–280.

[25] Tie Li; Gang Kou; Yi Peng; Yong Shi (2017). “Classifying With Adaptive Hyper-Spheres: An Incremental Classifier based on Competitive Learning”, *IEEE Transactions on Systems*, Man, and Cybernetics.

[26] Gang Kou, Yanqun Lu, Yi Peng, and Yong Shi (2012). “Evaluation of Classification Algorithms using MCDM and Rank Correlation, International *Journal of Information Technology & Decision Making*”, Vol. 11, Issue: 1, 197-225.

رویکردی بهبود یافته برای مساله جایابی ماشین مجازی مبتنی بر شبکه‌های نرم‌افزار محور در محیط رایانش ابری

امیرحسین صفری باویل[۱]، سام جبه داری[۲*] و مصطفی قبائی آرانی[۳]

[۱و۲] گروه مهندسی کامپیوتر، واحد تهران شمال، دانشگاه آزاد اسلامی، تهران، ایران.

[۳] گروه مهندسی کامپیوتر، واحد قم، دانشگاه آزاد اسلامی، قم، ایران.

ارسال ۲۰۲۰/۰۵/۲۸؛ بازنگری ۲۰۲۱/۰۱/۲۶؛ پذیرش ۲۰۲۱/۰۴/۲۱

چکیده:

به صورت کلی در زمینه شبکه‌های کامپیوتری، مسئله تضمین کیفیت دارای یک ضرورت خاص می‌باشد. شبکه های کامپیوتری که در سازمان‌ها استفاده می‌شود، به صورت سنتی بوده و دارای ساختاری سلسله مراتبی می‌باشد که با استفاده از گره هایی از سوئیچ‌های اترنت در یک ساختار درختی شکل می‌گیرد. یکی از اصلی ترین و بنیادی ترین پروتکل‌های SDN، پروتکل Open Flow است که امکان دسترسی مستقیم و ایجاد تغییر در برنامه ارسال تجهیزات شبکه نظیر سوئیچ‌ها و مسیریاب‌ها را، هم به صورت فیزیکی و هم مجازی فراهم می‌کند. نبود یک واسط باز در برنامه ارسال داده، باعث شده است تا تجهیزات شبکه‌های امروزی به صورت یکپارچه، بسته و شبیه پردازنده مرکزی شده باشد. در این مقاله راهکاری ارائه شده است که می‌تواند ترافیک را با استفاده از مکانیابی درست ماشین‌های مجازی کاهش دهد و در کنار آن امنیت این ماشین‌ها را نیز دچار مشکل نکند. راهکار پیشنهادی مبتنی بر الگوریتم شعله و پروانه می‌باشد و با راهکارهای مشابه با روش پیشنهادی نیز مورد ارزیابی قرار گرفته است و نتایج حاکی از برتری روش پیشنهادی دارد.

کلمات کلیدی: حفظ امنیت، جایابی ماشین‌های مجازی، شبکه‌های نرم افزار محور، الگوریتم شعله و پروانه.