Computers and Electrical Engineering 93 (2021) 107265

Contents lists available at ScienceDirect

Computers and Electrical Engineering

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

A protocol for cluster confirmations of SDN controllers against DDoS attacks

Amir Iranmanesh *, Hamid Reza Naji

Faculty of Electrical and Computer Engineering, Department, of Computer Engineering and Information Technology, Graduate University of Advanced Technology, Kerman, Iran

ARTICLE INFO

Editor: Dr. M. Malek

Keywords:
Cluster Confirmations
DDoS
Genetic Algorithm
SDN

ABSTRACT

Software-Defined Networking (SDN) makes it easier to manage the network by separating the Data plane and Control plane, but these networks are susceptible to DDoS attacks. Most well-known algorithms for reducing the effects of DDoS attacks are based on SDN traffic analysis and prediction. Their main problems are false positive/negative results. Therefore, we propose PATGEN, a Protocol to reduce the effects of DDoS ATtacks using an advanced GENetic algorithm with optimized and new operators, thereby significantly reducing the effects of attacks and increasing the efficiency of the multi-controller SDN. Using the robust initial population procedure increases the convergence speed of the algorithm. Unlike other methods, PATGEN is possible without using external resources. Experimental results show that PATGEN increases the throughput and reduces the average delay compared to states of the art methods.

1. Introduction

Software-Defined Networking (SDN) is emerging rapidly to support dynamic network operations by separating the data plane from the control plane. The logical process and efficiency of SDN networks are mainly based on software controllers (control plane). The SDN enables network-programming capability and can solve many of the challenges of network management.

The use of SDN is increasing by the day. In the transportation industry, businesses from small local networks to massive cloud computing for process large amounts of data all benefit from SDN. Simplicity, high flexibility, and low cost are essential reasons for the widespread use of SDN. However, separating the control plane from the data plane has made these networks always attractive targets for attackers because disrupting the control unit will disrupt the whole network. In Distributed Denial of Service (DDoS) attacks, multiple systems attack a resource through numerous Internet connections, while in a DOS attack, a system attacks a source through an Internet connection.

In the SDN-managed network, when a new packet arrives at the switch and the switch is unable to find the matching and correct flow entry, it sends the packet to the controller. When the DDoS attacks occur, it exploits the same SDN feature and sends a large number of packets to the SDN network. These packets reach the controller for management, and the controller creates new flow entries corresponding to these fake packets, which fills the entire flow table in SDN switches and reduces or cuts performance.

This paper introduces PATGEN, a *P*rotocol to reduce the effects of DDoS *AT*tacks using an advanced *GEN*etic algorithm on multi-

This paper is for regular issues of CAEE. Reviews processed and recommended for publication by Area Editor Dr. G. Martinez Perez.

* Corresponding author.
E-mail addresses: amir.iranmanesh@student.kgut.ac.ir (A. Iranmanesh), h.naji@kgut.ac.ir (H. Reza Naji).

https://doi.org/10.1016/j.compeleceng.2021.107265
Received 22 August 2020; Received in revised form 9 June 2021; Accepted 11 June 2021
Available online 29 June 2021

controller SDN networks. Besides, a load balancing procedure is used to prevent controllers from being idle or overloaded. The network we have considered in this article consists of different domains, and the SDN controller is provided for each domain. We have chosen the multi-controller SDN systems for the study because a system with only one controller will fail in large and bulky networks due to the massive influx of requests. With the explosive growth of scale and traffic networks, multiple controls must be deployed to improve the scalability and reliability of the control plane. We use a load balancer to avoid the excessive number of flow tables and lack of resources in the network. Finally, the load balancer reduces network latency and increases performance.

By experimenting on the SDN network with different scenarios, as shown in this paper, we concluded that our proposed method achieved consistently better performance than either the basic SDN or recent method in this field. The proposed protocol is also evaluated with both malicious workloads and legal workloads. The throughput results, the number of packets processed by the controllers, the CPU speed, and the delay indicate that the proposed protocol is highly efficient.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 introduces the PATGEN protocol, and its details will be provided. Details of the evaluation and experiments are given in Section 4, and conclusions with consideration for future work are presented in the final section of the paper.

2. Related work

For preventing the impacts of DDoS attacks, most related works focused on analyzing attacks. However, the fundamental defects of these methods are false positives or false negatives. Notably, several mitigation methods have been proposed, with the focus being on the use of centralized control operations to reduce the effects of DDoS attacks. Alqahtani's method [1] for mitigation of the effects of DDoS attacks is based on replication, so that it also uses several auxiliary controllers in addition to having one master controller. Although this method uses additional resources, it did not consider the load balancing or idleness of these resources during execution time. Deeban et al. proposed a queue-based scheduling method to mitigate the effects of DDOS attacks that utilize multiple queue bases [2]. Tran et al. introduced a flow-guard method that uses a robust firewall to prevent and reduce the effect of DDoS attacks and is an attack-based separation method [3]. However, this method is based on traffic flow analysis and also does not consider load distribution at runtime. Macedo et al. used a cluster-based approach to mitigate DDoS attacks [4], and their method was named PATMOS. Specifically, the advantage of Macedo method is that it did not use additional resources. However, this method focuses more on preventing the bottleneck and less on the genetic algorithm's effectiveness since using random population as the initial population of classic genetic algorithm. As we prove in terms of efficiency and rate of convergence to a near-optimal solution, PATMOS has poorer results than our proposed algorithm. In particular, our algorithm has higher accuracy in identifying all possible solutions, as shown in this paper.

As mentioned before, most of the methods to mitigate the impact of DDoS attacks are based on traffic analysis, which has many weaknesses. Cao et al. introduced a process that uses network traffic analysis to detect DDoS attacks [5]. The authors analyzed the controllers' traffic data and increased security by disposing of malicious traffic if a controller is overloaded. Jafarian et al. Introduced a hybrid method [6] that used the NetFlow protocol to gather information and generate a dataset and Information Gain Ratio (IGR) to select superior features and then used the ensemble learning method (stacking) to develop an efficient structure for anomaly detection in SDN. However, this method used a random initial population; and did not consider load balancing at runtime. Asis et al. proposed a near real-time SDN security system [7] to detect and prevent DDoS attacks based on a Convergent Neural Network (CNN). However, this method also used statistical analysis from both the data plane and the control plane and did not consider the load balance between the controllers. All of these mentioned methods are based on network traffic analysis. In the paper [8], the authors propose a flexible security overlay network deployed on-demand and grows as needed, and utilizing cloud providers' virtual infrastructure. In the introduced overlay network, protected endpoints are disconnected from the Internet, and only requests without malicious are accepted. The firewall in this method performs control operations in various ways. This method used the cloud for detecting attacks, also used network layer filtering, admission control, and congestion control methods. Moreover, it utilized auto-scaling and network flexibility to reduce DDoS attacks. This method is based on network traffic analysis.

Karapoola et al. introduced the Net-Police service, a patrol system, to reduce the impact of DDoS attacks. The Net-Police identifies the sources of the attack in order to filter those sources and reduces the effects of the attack as quickly as possible [9]. However, this method used traffic analysis to identify susceptible areas of attack and also did not consider resource efficiency. In [4], only one solution for the number of predetermined clusters was generated per run of the algorithm, while in PATGEN, a range of solutions with different numbers of clusters is generated per run of the algorithm. In [10], the authors presented a DDoS-type attack detection and mitigation framework called ProDefense. The SDN-oriented ProDefense was a proactive framework for securing smart city data centers against DDoS attacks. ProDefense allowed to implement application-specific requirements to detect and reduce DDoS attacks. ProDefense also used distributed controllers, thus providing an efficient method for load distribution and improving network reliability.

Similarly, in [11], the authors proposed a lightweight method that identified packets sent to switches are malicious or correct based on a set of policy rules. The authors' approach used a table that is responsible for storing the IP addresses of incoming packets. Their method used three counters. The first counter determined the number of packets in each flow, and the size of packets in each flow was determined by the second counter. Finally, the third counter determined how long the flow has been in FlowTable. Besides, this system used two thresholds (hard time-out and idle time-out) to improve its performance in SDN.

Most mitigation methods of the effects of DDoS attacks used generic types of replication, such as controller checking and attack isolation. WANG et al. introduced a safeguarded scheme (SGS) to protect the control plane against DDoS attacks [12]. The main feature of SGS was the deployment of multiple controllers in the control plane by clustering the controllers. However, this method was based on the replication of the controllers. Krishnan et al. introduced a security framework to prevent controllers from overloading so that

they set up a separate controller for each switch [13]. However, this method used extra equipment outside the system. Chen et al. Considered the problem of network schedule as a problem of maximizing overall network efficiency. To prevent controllers from overloading, the authors have introduced a scheduling-based approach that they set for each controller switch [14]. However, the authors assumed that the attacker only implemented IP address spoofing.

Other methods have been introduced to reduce the effects of DDoS attacks, including methods that utilize centralized authentication within clusters. Macedo et al. proposed a method using network traffic analysis and a classic genetic algorithm that eliminates dependency between DDoS attack identifiers [4]. This method also reduced the effects of DDoS attacks by clustering controllers via the basic genetic algorithm, and the superior feature of the Macedo method was that it did not use any additional resources to reduce DDoS attacks. Despite the advantages of this method, the classic genetic algorithm for controller clustering cannot investigate all the possible solutions, and the convergence rate of this algorithm was extremely low, as shown in this paper. In contrast, PATGEN can find the optimal global solution and provide the most efficient clustering method for controllers due to the use of optimized and new operators in the genetic algorithm as well as load distribution routine.

Most existing methods [5][6][159] used machine learning and deep learning algorithms to detect DDoS attacks in the SDN environment and mitigate impacts of attacks based on their founding from the detection phase. Besides, they need to analyze network traffic to learn based on it. Therefore, these methods require high computational resources depending on the scale, which becomes a problem in large networks with Big data. Also, in neural network-based methods, the nature of the network itself imposed a high time complexity.

In short, for calculating the delay, we used some analysis in our calculations. But in order to analyze the whole network traffic like machine learning algorithms [6] or deep learning algorithms [15], a lot of computational resources of the network must be used. In contrast, our proposed method has avoided the analysis of the whole network traffic. As a result, PATGEN consumes fewer computing resources.

Macedo et al. also introduced a smart cluster-based framework for mitigation that continues to benefit from random initial populations [16]. The authors proposed a method based on a classic genetic algorithm to reduce the impact of DDoS attacks on the network. However, this method has the problem of population selection and diversity. The clustering solution presented by the authors' genetic algorithm is not the best, as shown in this paper. So we decided to improve this framework to an efficient way of reorganizing the SDN controllers into clusters to reduce the effects of DDoS attacks. Our work differs from these efforts because we have altered the basic genetic algorithm and used an efficient load balancing routine. Moreover, our proposed method does not use additional resources that will enforce cost to the system. Also, because traffic analysis methods are often false positive or false negative, we have not used these methods to reduce the effects of DDoS attacks.

Our proposed protocol first identifies overloaded controllers via message exchange. The next step, among the available controllers, selects a higher performance controller as the header for cluster management. Nominated nodes are the k top controllers in terms of performance level, which are identified as the coordinator recommendation group. If, for any reason, the header controller fails, chosen controllers select the header controller. Choosing the following header from the nominated nodes saves the number of messages and time. Finally, without the use of additional resources and scheduling of requests, it reduces the effects of DDoS attacks by using cluster confirmations of controllers via the enhanced genetic algorithm.

3. PATGEN design

In this section, we present the design of PATGEN, a controller clustering and DDoS attack effect reducing protocol. PATGEN reduces the effects of DDoS attacks using a multi-level method. Our method has three levels include finding bottlenecks, controller selecting, and SDN controller cluster conformations. At the level of finding bottlenecks, controllers that are overloaded are identified. At the selection level, a controller is selected as the selection header to coordinate the clustering process. At the lower level, the controllers combine in clusters to reduce the effects of DDoS attacks.

At each level, the controllers have specific characteristics. At the level of finding bottlenecks, each of the available controllers must regularly perform message exchange to determine which controller is overloaded. At the selection level, controllers can have one of five roles: nominate, header, substitute, regular, and worker.

Nominated controllers are controllers that have the right to choose a header. At any given moment, only one node is selected as the header controller. The node that is chosen as the header is agreed upon by all the nodes. Because in SDN, every controller is at risk of DDoS attacks and can overload quickly, the algorithm selects a controller as the header, which is the best in quality (the controller with highest P according to Eq. (3)). The header controller selects the best configuration for the clusters to reduce the effects of DDoS attacks. The second highest quality controller is labeled as a substitute header controller. When a controller is under DDoS attacks in a cluster, it is the job of the worker controller to perform the tasks of the overloaded controller.

3.1. Bottleneck finding layer

In order for a system to be robust against DDoS attacks, there must be a way to identify the overloaded controllers. When a controller is attacked, a large amount of malicious messages is sent to the controller, causing all controller resources to be consumed, and the controller is called an overloaded controller. At this level, we use the optimized gossip protocol to identify overloaded controllers [17]. The optimized gossip protocol has acceptable time complexity and is able to detect new nodes without wasting resources, so in the proposed method, we have used the optimized gossip protocol to detect failures. Gossip protocols provide a tool by which failures can be detected in large, asynchronously distributed systems without the limitations associated with reliable multi-component

communications. The Gossip Message (GM) content and its interpretation are based on the study of Ranganathan [17].

In the PATGEN protocol design, we assume that there are N nodes in the network, and each of these nodes has its own identifier. These nodes are connected as a complete graph so that each node can be directly connected to the other nodes. Each controller has two attributes in its normal state, including a unique id number and a heartbeat value, which is initially zero.

Controller b selects controller a to send a gossip message, as shown in Fig. 1. As such, the edges of the graph represent Gossip Messages (GMs). The duration of this operation is t_{gossip} seconds.

In the proposed method, we make this selection based on the binary Round-Robin (RR) procedure [18] because it consumes fewer resources and makes fewer rounds than the classic RR method. If the number of nodes in the network is n, the selection operation completes in the time complexity of O(logn).

The controller selection at the destination is made using Eq. 1.

$$D_{ID} = 2^{round-1} + S_{ID}1 \leq round < log_2(n) \tag{1}$$

Where n is the number of controllers available in the network, and S_{ID} is the Gossip Message (GM) sending node, and D_{ID} is the GM receiving node, and round represents the current round. Table 1 presents the symbols and their descriptions used by PATGEN.

Then, in the network, node b sends a GM message to node a and waits for the response. In the method presented in this article, we assume that each node has enough memory to store messages. As soon as node a receives the message, it updates its contact list. Then it gives the most valuable heartbeat to the message received from node b and then responds with the GM to controller b. After the contact list is updated, then a and b automatically identify new nodes. Alternately, nodes broadcast GM to be recognized on SDN.

In our proposed method, controller b in the network checks for two characteristics to determine the overload of controller a: the controller stability, the delay in the control messages. The delay control messages of these parameters are indicated by $flag_{sa}$ and $flag_{oa}$. The value of these flags is zero or one. By default, the value of these flags is zero, and if they become 1, the controller overloads or is no longer stable. If node b waits for node a response for more than t_f seconds, then the $flag_{da}$ value changes to 1, indicating that controller a has a delay in responding to the Gossip message. After the delay is detected by node b, since this time delay is not detected by all controllers in the SDN environment at the same time, node b waits a while and then deletes node a from its contact list. Node b sends GM to other nodes containing $flag_{da}$ and waits for $t_c >= t_f$, then removes node a from the contact list.

The stability of each controller is calculated using Eq. (2).

$$S_a = \frac{Ar_a}{CO} \tag{2}$$

Where Ar_a is the number of current requests reached to the controller, and Co is the number of completed requests of the controller a.

It can be deduced from Eq.2 that if $s_a = 1$ then $flag_{oa} = 0$, but if $s_a > 1$, then $flag_{oa} = 1$. The gossip message will contain $flag_{oa}$ to inform other nodes about the stability of node a.

When a DDoS attack occurs, the attacker sends a large number of fake packets to the switch. The controller corresponding to each switch must create a new entry for each packet and will overload very quickly. If the switch connected to controller a becomes overloaded, flagoa = 1. GM's $flag_{oa}$ informs other nodes about overloading node a. Fig. 2 shows this stage of the operation.

3.2. Selection layer

This layer selects a controller as the header to streamline the process of reducing the impact of DDOS attacks. The quality of all controllers must be carefully checked, then the controller with the highest quality is selected as the header. When executing this procedure, the controllers are in two states: normal or overloaded. If the total number of controllers is n_c and n_o is the number of controllers in an overload state, then n_o-n_c denotes the number of regular controllers.

In the proposed method, each node only has information about its neighbors, and there is no node that knows the whole network. Even if we select the highest quality controller as the header, it may be overloaded under DDoS attacks and may not work properly, or the header node may crash and fail, so the fault-tolerant method for the header node should be considered at this level. To overcome the challenges posed, if controller b detects controller a is overloaded, it broadcasts a message to other controllers and waits t_e seconds after sending the message.

All the controllers on the network find out through the Election message that the header selection operation has started on the network, as shown in Fig. 3. When the election message reaches a third controller such as c, controller c acts as the nominate controller and, by sending an acknowledgment message to controller d, declares its priority to be the header, as shown in Fig. 4.

The score of each controller in the network is denoted by P_k and represents each controller's performance in the SDN network. In

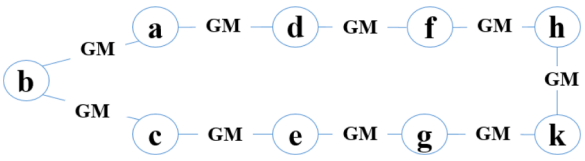

Fig. 1. The Gossip messages.

Table 1
List of Notations.

Symbol	Description
t_g	time to send a message in unicast-form
t_b	time to send a message in broadcast-form
t_f	time for a regular controller to fail
t_c	time to remove a controller from the contact list
hb	numeric value to display heartbeat
co_a	number of requests fully processed by the controller a
s_a	stability Rate of controller a
Ar_a	number of flow requests reached to controller a
R	current round used in binary round GP
ID_d	destination node of gossip message
$flags_{s_a}$	flag to display stability of controller a
$flag_{o_a}$	flag to display overloaded controller a
$flag_{D_a}$	flag to display delay for controller a
t_e	timeout score to announce the election result
n_s	node score to become a header
C_{ch}	Cabinet of chosen controllers to participate in the selection
P_k	The score of each controller in the network
P	The node priority to become header in the election process

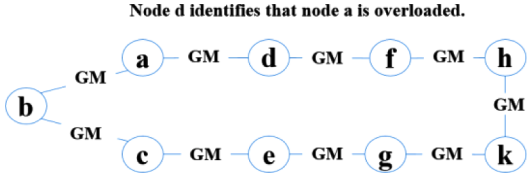

Fig. 2. Identifying the DDoS attack in node a.

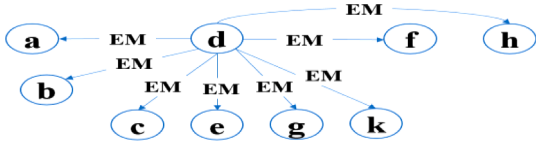

Fig. 3. Election messages in PATGEN.

the proposed protocol for each node, we have different characteristics such as storage capacities, processing power, and processing speed. DDoS attacks disrupt the characteristics and specifications of each network node.

The value of P is calculated from Eq. (3). If controller c is a nominate controller for header and controller a is overloaded, then:

$$P = \frac{(cp1 \times M_{c,a}) \times (cp2 \times u_{c,a}) \times (cp3 \times n_{c,a})}{cp1 + cp2 + cp3} \tag{3}$$

Where $M_{c,a}$ is the average of unused memory between nodes c and a. $u_{c,a}$ is the average CPU performance used, and $n_{c,a}$ is the average of the packets exchanged between the controller c and a, Which means that $M_{c,a} = \frac{M_c + M_a}{2}$, $u_{c,a} = \frac{u_c + u_a}{2}$, and $n_{c,a} = \frac{n_c + n_a}{2}$. However, $cp1$, $cp2$, and $cp3$ are control parameters used to modify P and are provided for SDN network administrators to make adjustments based on their expertise and experience.

After t_e seconds, controller a sends a coordinator message to all the nodes in the network, including the header election results, as shown in Fig. 5. The node with the most top p is the header, the node with the second-highest P is the header substitute, and C next nodes with the highest p are selected as the coordinator recommendation group.

When the controller k receives the CM message, if selected, it will play a specific role. Otherwise, it will have a regular role in the

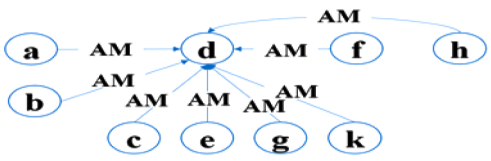

Fig. 4. Acknowledgment messages in PATGEN.

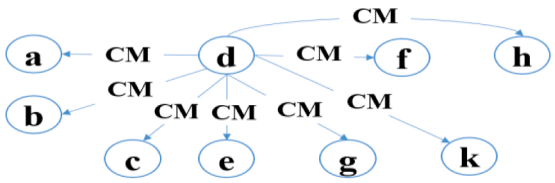

Fig. 5. Coordinator Messages.

network. Suppose the header controller is under DDoS attack, or for some other reason, the header performance is lower than the average performance. In that case, the header substitute node becomes the header node and replaces the header. However, the best node of the nominated nodes is introduced as the header substitute controller, thus maintaining the fault-tolerant of the header controller in the network.

3.3. Combination layer

At this layer, three basic operations are done to reduce the effects of DDoS attacks, which are:

1 The best combination of controllers is determined in separate clusters.
2 The clusters are reset.
3 The network settings are reset.

In this paper, we consider the SDN network as multiple domains that are geographically distributed. The multi-controller system assumes that each controller is responsible for one domain in the network. Each controller exchanges information about its own domain, including the status of switches and routers, with other controllers on the same network, and the controllers within a cluster are able to work together. At this level, to minimize the effects of DDoS attacks, the optimal number of clusters and controllers in the cluster is considered. For the problem of controller clustering, the classic solution is to put all the controllers in one cluster. However, this solution is only applicable to a local network composed of a few interconnected nodes. When it comes to a network consisting of a large number of geographically distant nodes, the classic methods will not have an acceptable solution. Therefore, we need a multi-controller system for ease of communication management and efficiency enhancement. The optimizing of the clustering of controllers is a multi-objective problem. The first objective is to optimize the number of clusters, and the second is to optimize the number of elements in each cluster.

The genetic algorithm is one of the most commonly used solutions. Although, in the case of the paper [16], which uses the genetic algorithm to solve the problem because of the low variation of the initial population as well as the use of a random initial population, the algorithm does not provide the optimal solution. Nevertheless, in the algorithm used in this paper, we have used both the new operators and the modified operators rather than the basic genetic algorithm in a way that offers twice the efficiency of the solution rather than the classic genetic algorithm. Each of the controllers in the SDN system implements Algorithm 1.

Genetic algorithm inputs consist of parameters defined by the SDN administrator and data collected from network controllers. The controller receives the Acknowledgement message containing the access information of other network controllers, as shown in Fig. 4.

Then the controller *b* inserts this information into an array *C* and sends it to the header *a* by the Clustering Confirmation Message, as shown in Fig. 6. Each element in array *C* represents the capacity available to each controller after the DDoS attacks. The numbers of generations (N_g) and population size (N_p) are the parameters that are set by SDN admins as inputs for the genetic algorithm to generate the optimal solution.

In Algorithm 1, the first line produces the initial population N_p. The population consists of a set of elements called chromosomes. Traditional coding for clustering is highly disruptive, and hence, this coding is harmful to the clustering process. In the method presented in this article, we have used a new encoder that does not have these disadvantages.

For a simple example of the problem, we present a solution using the proposed method and a chromosomal representation. It should be noted that the capacities of the controllers in representing array P for this simple example can be written according to the problem conditions and the network.

We have designed the chromosomes in a 2-dimensional way to be more efficient than other related works. Fig. 8 displays the

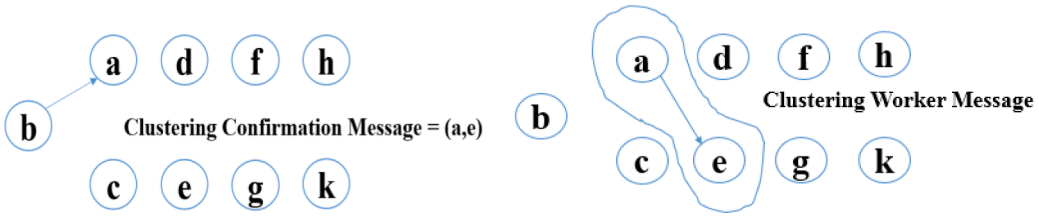

Fig. 6. Cluster Messages in PATGEN.

chromosome (a possible solution for the problem), and The SDN controllers are located in the array *P*. The capacitors of the controllers in the array *P* are (20,40,5,10,30,15,5,15,10), respectively, and the optimal solution is displayed in the form PS = {(a,e), (b,d), (c,f,k,h, g)} that the total capacity of each cluster is 40 and besides, all the controllers in the network are also used. The way it is represented in the algorithm makes it possible to execute modified crossover, mutation, and inversion operators to ensure population diversity and achieve an optimal global solution.

In Algorithm 1, The number of generations (N_g) and population size (N_p) can be adjusted by the SDN admin. In experiments for simplicity, we have considered N_g and N_p both equal to 100. With trial and error in our experiments, we find that the optimal values for Elitism and Mutation probability are 0.79 and 0.019, respectively. N_c and N_m are the number of crossover and mutation operations, respectively. Specifically, PATGEN, independent of the values of these parameters, produces the optimal solution.

Two arrays named *C* and *V* are used, where the array *C* represents the capacity of each controller after the DDoS attack, and the array *V* represents the capacity of the controllers in the normal status. In the evaluation step of Algorithm 1 (lines 3 through 6), each generation's chromosome is evaluated according to the fitness function. The fitness of the chromosome *x* is evaluated by Eq. 4.

$$F(X) = Y_1(X) + Y_2(X) + Y_3(X) \tag{4}$$

Where $Y_1(X)$ indicates the extent to which clustering prevents controllers from being overloaded by DDoS attacks, and the total capacity of each cluster is calculated. In particular, for each cluster in the solution that is able to avoid overload, $Y_1(X)$ is incremented. Specifically, $Y_2(X)$ determines that the controller set can be broken so that we have the maximum number of solutions. For computing $Y_2(X)$, the algorithm uses a list of all the controller elements. All chances of removing the controllers are checked by using this list. If the system is able to reduce the effect of DDoS attacks by removing even one controller, $Y_2(X)$ will not increase. Otherwise, $Y_2(X)$ is incremented. As such, $Y_3(X)$ specifies the equal amount of capacity per cluster. A list is created and filled with equal capacities of clusters. For each input iteration, $Y_3(X)$ increments. The result of the function $F(X)$ for each chromosome X is used to classify the chromosome and to keep the better chromosome, and discard the other chromosomes. Therefore, the proposed genetic algorithm always maintains the best solution. In the proposed genetic algorithm, we use an initial population procedure that produces the initial population in a way that has a higher convergence rate than the other methods presented in this field. Fundamentally, the methods based on traditional genetic algorithms use a random initial population, which slows the algorithm to provide a near-optimal solution.

In Algorithm 2 (Procedure Selection), the appropriate chromosomes are selected by the binary tournament for selection operation. Then, the crossover and mutation operators are applied to the selected chromosomes. The two crossover and mutation operators in the PATGEN have been modified rather than other genetic algorithms in the field of clustering of the controllers. The efficiency of both operators is doubled due to the simultaneous application of single-point and double-point methods. The new and optimal inversion operator is then applied to selected chromosomes to increase population diversity. The pseudo-code for crossover and mutation procedures is shown in Algorithm 3 and Algorithm 4, respectively. Each generation of the proposed algorithm ends on line 14 and jumps to line 2 to start the next generation (Algorithm 1). The proposed algorithm is repeated until the criteria are reached.

The general steps of the two-point crossover operator are shown in Fig. 9. Initially, two integers are randomly chosen between 0 and the number of controllers representing two points of the crossover. Each of these points divides the parent chromosomes into the middle, head, and tail. The chromosome son is made up of the middle part of the father and the mother's chromosome's head and tail, and the chromosome daughter is produced in reverse.

Because some solutions are not produced by the crossover operator, and to increase chromosome diversity after the crossover operation, the mutation operator is applied to the chromosomes. Steps of the procedure mutation are presented in Algorithm 4.

The inversion operator is introduced to increase population diversity and guarantees a variety of samples. It replaces random populations with optimal populations, converges to optimal solutions faster than other proposed algorithms in this field, and converges to optimal solutions with fewer iterations of the genetic algorithm. As a result, as shown in Fig. 10, the quality of the scheduling increases for more complex networks with more switches.

We generate controllers using Python-based POX Mininet controller and use Python libraries like PySDN mininet.cli, mininet. topolib, mininet.net, mininet.log, mininet.node, and so on.

We segment the SDN controllers based on the level of the processing power of the SDN controller. Controllers with the same processing power fall into the same segment, as shown in Fig. 11. The details of the inversion operator are shown in Fig. 12. The criterion for terminating the algorithm is to reach the solution with the most fitness.

Finally, the header controller sends a Clustering Message (ClM) to the controller that is overloaded. Overloaded controllers with worker controllers make a cluster to mitigate the effects of DDoS attacks. Afterward, to restore the initial state of the network, the header controller sends the Network Status Message (NSM) to all SDN controllers to restore the network status, as shown in Fig. 7.

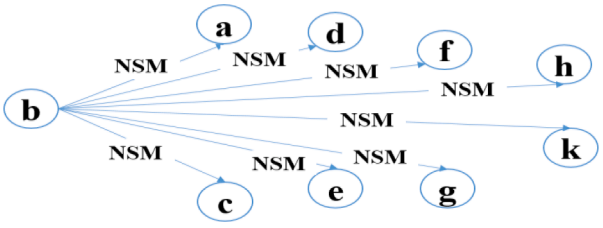

Fig. 7. NSM messages in PATGEN.

In the enhanced genetic algorithm, the best cluster is determined by Eq. (4). Specifically, the best cluster contains the best combination of controllers. In the case of identification of the header controller in each cluster, identification is through Eq. (3) and also based on the study of the method [19]. Figure 16 shows the efficiency of the proposed method.

4. Performance evaluation

We have implemented the simulation within the SDN simulator, the Mininet. The SDN network is simulated with five controllers as a multi-controller. We used a Memcached system [20] to share status information between network controllers. Memcached is a general-purpose memory caching system. The system used a client-driven architecture and can run across several different machines. For the evaluation of PATGEN, we have used concurrent malicious and correct datasets. A DDoS attack is performed by 54 hosts on 18 switches for 60 seconds and injects the malicious workload into the system. The Cbench tool generates legal workloads using 32 switches and 2000 hosts. Specifically, in our experiments, the DDoS attack's assumptions and the nature of DDoS traffic are based on the study of Lim [21].

We have set a switch for every three hosts and set the DDoS attack time to 60 seconds to study and evaluate the number of entities in the flow table, CPU usage, and bandwidth during the attack. Besides, after the completion of the DDoS attack time, we can see changes in these parameters. Our proposed method is independent of the number of hosts, the number of switches, and the DDoS attack duration. We repeated the experiments with different values of these parameters, but the same results are obtained. We have used System Monitoring Tools in Linux to measure the consumption of system resources, including CPU. In the experiments, we used isolcups and taskset commands to separate CPU resources between the controllers and Cbench [22]. Also, the summary of system configuration is listed in Table 2.

Packet production is done by Cbench [22]. Cbench is a powerful program for generating interactive packets. It can forge or generate a large number of packets of different protocols, send packets over the wire, process requests and responses, record packets, and more. We used Cbench to generate packets and spoofed the source IP addresses. We generated two types of traffic, which are normal traffic and malicious traffic. Malicious traffic has a higher rate than normal traffic [23]. The controller can separate the normal traffic from malicious traffic based on the study [23].

Five scenarios are implemented, the first of which is the multi-controller system scenario without the use of PATGEN, which is the base method scenario. In the second scenario, PATGEN operates on clusters of two controllers, and we call it 2CPATGEN. Specifically, in this scenario, PATMOS operates on two controllers and is called 2CPATMOS. In the third scenario, PATGEN operates on clusters of three controllers, and we call this scenario 3CPATGEN. In this scenario, PATMOS operates on three controllers and is called 3CPATMOS. Likewise, the next scenarios are called 4CPATGEN, 5CPATGEN, 4CPATMOS, and 5CPATMOS. We adjust the capacity array of the genetic algorithm to include these scenarios. Henceforth, C1 to C5 specify controller 1 to controller 5 in Figs. 13,14,15,16.

We used four parameters to evaluate performance:

1) Number of received packets
2) Throughput
3) CPU utilization
4) Delay time

Delay time and the number of received packets are collected by Apache JMeter 5.0 tool [24]. At PATGEN, we take the end-to-end delay across the path and calculate it based on the study [23]. Besides, to accurately calculate link latency, OpenFlow packet-in message is injected into a standalone VLAN, and the OpenFlow packet-out message is retrieved. This operation is performed to calculate the time of sending and receiving the message accurately.

Fig. 13 shows the average rate of CPU utilization (with a 95% confidence level). All results are averaged by run the algorithm over 100 times. In the case of CPU utilization, in the baseline scenario, the average consumption rate by controller *C5* was 22.5%, and in the 2CPATMOS scenario [16], 16.5%, while in the 2CPATGEN scenario, the CPU consumption decreased to 9.8%. The average usage in the 3CPATMOS scenario is 13.30%, which in the 3CPATGEN scenario is 6.36%. In 4CPATMOS, CPU utilization drops to 12.78%, but in the 4CPATGEN scenario, it falls to 5.76%. Finally, in the 5CPATGEN, this value drops to 3.89%. The average CPU utilization reduction rate of PATGEN is about 57.33% compared to the PATMOS because PATGEN uses both an optimal initial population and an efficient load distribution procedure at execution time. As can be seen in Fig. 13, the average CPU utilization in the 2CPATGEN mode is very high because of the use of resources that have remained unused so far. For example, compared to the 5CPATMOS scenario for the controller *C5*, 5CPATGEN has reduced the average CPU utilization by approximately 15.91% than 5CPATMOS.

Fig. 14 shows the more appropriate distribution of packets between controllers than the base method and PATMOS. Because using random initial populations and stuck in local optima, PATMOS could not search for all the solutions; therefore, it was not able to find

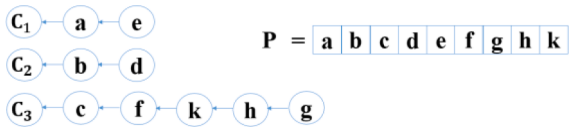

Fig. 8. An example of a two-dimension chromosome and the array of SDN controllers.

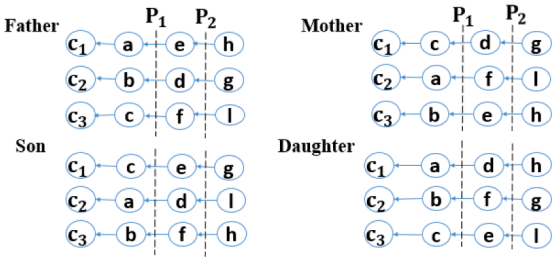

Fig. 9. An example of the two-point crossover operation.

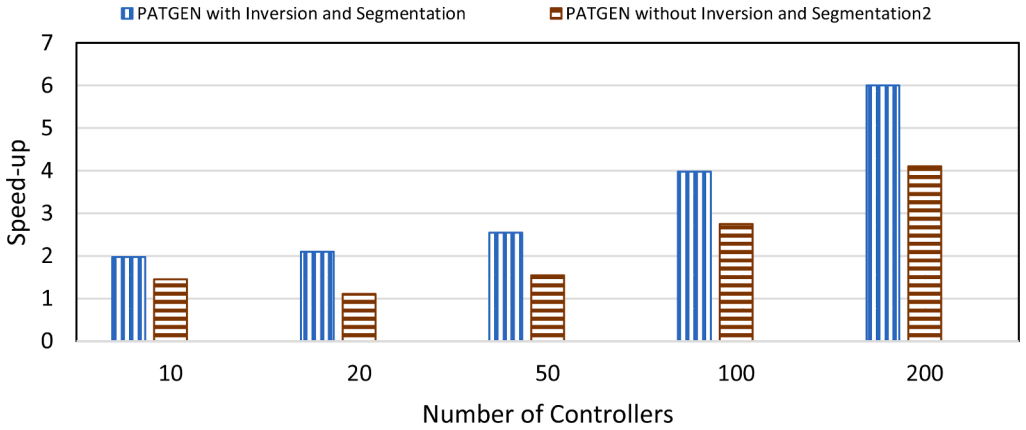

Fig. 10. Comparison of speed-up between PATGEN with inversion operator and without inversion operator.

Seg 1	c	k
Seg 2	d	g
Seg 3	h	f
Seg 4	a	
Seg 5	e	
Seg 6	b	

Fig. 11. An example of segmentation for SDN controllers.

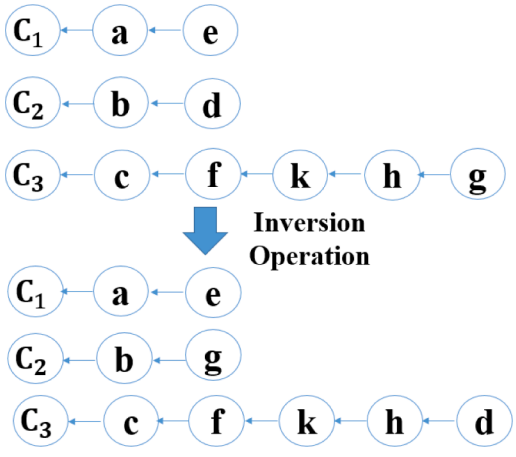

Fig. 12. An example of Inversion Operation.

Table 2
Parameters for system configuration.

Experiment configuration	
Parameters	Values
CPU	Intel Core i7
Memory	8GB
OS	Ubuntu 14.04

Algorithm 1
Finding the optimal combination of controllers within clusters.

Input: V Damage Array
C Capacity Array
N_g Number of Generations
N_p Population Size
a Elitism Probability
b Mutation Probability
Output: S Near-Optimal Solution
1) initial population generation of size N_p
2) while ($i < N_g$ and solution not found) do
// Evaluation
3) while $j < N_p$ do
4) Calculate the fitness value of each chromosome of the population by using the fitness function
5) $j = j + 1$
6) end while
//Elitism
7) The number of elite chromosomes $= a \times N_p$
8) Select E chromosomes that have the highest fitness and set them to N_e.
9) Call the Procedure Selection (Algorithm 2)
10) Call the Procedure crossover (Algorithm 3)
11) Call the Procedure mutation (Algorithm 4)
12) Call the Procedure Inversion
// Next Generation
// Insert the selected chromosomes into $N(s)$
13) $N(p) = N(s) + N(E)$
14) end while
15) Return S

Algorithm 2
Selection Procedure.

1) select two chromosomes at random from the current population
2) while ($k < N_g$) do
3) if (fitness(ch_a) < fitness(ch_b))
4) add ch_b to the initial population
5) endif
6) $k = k+1$
7) endwhile

Algorithm 3
Crossover Procedure.

1) N_c (Number of Crossover operations) $= \delta \times N(S)$
2) while ($k < N_c$)
3) Select two chromosomes x_i and x_j, at random
4) apply the two-point crossover to x_i and x_j and place the outputs in x_k and x_l
5) apply the one-point crossover to x_i and x_j and place the outputs in x_m and x_n
6) Calculate the fitness values of the four chromosomes x_k, x_l, x_m, and x_n
7) Select two of the best chromosomes with the highest fitness as $N(s)$
8) $k = k+1$
9) endwhile

the near-optimal solution. Nevertheless, our proposed method can explore the problem space more rigorously, quickly, and optimally. Therefore, it offers a general optimal solution because of using new and improved operators and the benefit of the initial population procedure.

Figs. 15 and 16 represent the throughput and delay of the DDoS attacked controller, respectively. The delay was obtained using the

Algorithm 4
Mutation Procedure.

1) Number of Mutation $N_m = b \times N_s$
2) while ($i < N_m$) do
3) randomly select the x_a chromosome from the population N*(s)*
4) apply a single-point mutation on x_a and place the result in x_b
5) apply a double-point mutation on x_a and place the result in x_c
6) calculate the fitness values of the two chromosomes x_b and x_c using the fitness function
7) put the best chromosome with the highest fitness in *N(s)*
8) $i = i+1$
9) endwhile

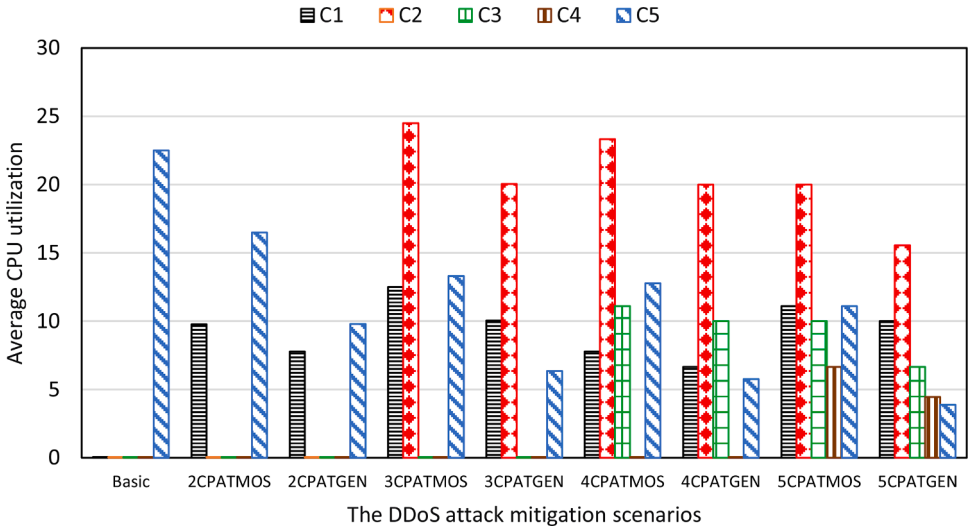

Fig. 13. Average CPU Utilization for the test scenarios.

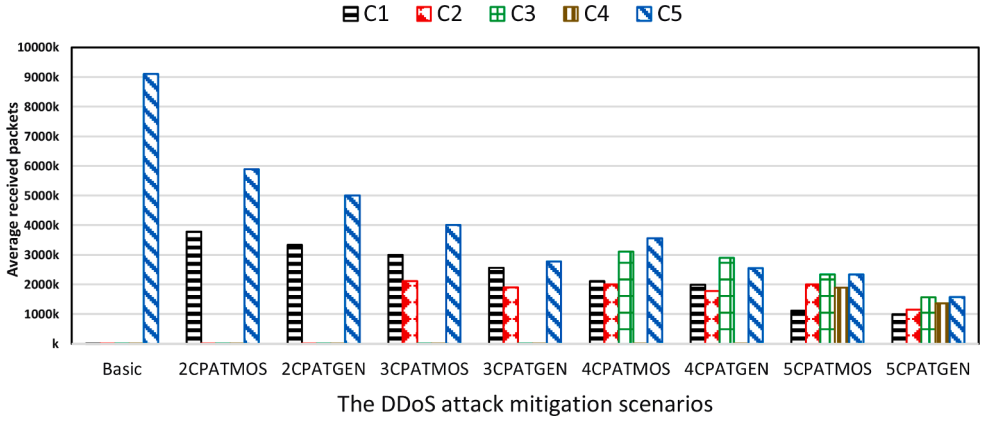

Fig. 14. Average received packets for the test scenarios.

hping tool 3, and the Cbench tool was used to measure throughput. The total number of packets sent to record throughput is equal to 10×10^6. In basic mode, the throughput is 30 req/sec, in 2CPATMOS 1625 req/sec and in 2CPATGEN, 2100 req/sec. In 3CPATMOS it reaches 2235 req/sec and in 3CPATGEN it reaches 3329 req/sec. The throughput reaches 4121 req/sec in 4CPATMOS and 5031 req/sec in 4CPATGEN. In 5CPATMOS, this reaches 4321 req/sec and 5225 req/sec in 5CPATGEN. The average delay is 1831ms in the classic scenario. In the 2CPATMOS scenario, it reaches 1521.82 ms, and in the 2CPATGEN scenario, it decreases to 1121.71, and the latency in the 3CPATMOS scenario is 1412.88ms, while the delay in the 3CPATGEN scenario is 821.66ms. The latency in the 4CPATMOS scenario is 800.75ms, and in 4CPATGEN, it reaches 255.59ms. In the 5CPATGEN scenario, this is reduced to 120.53ms. These results indicate that PATGEN is 35% more efficient than the PATMOS on average and about 252.34 times better than the basic method. In terms of reducing the average latency of requests flow, our proposed method is three times less than the basic method and

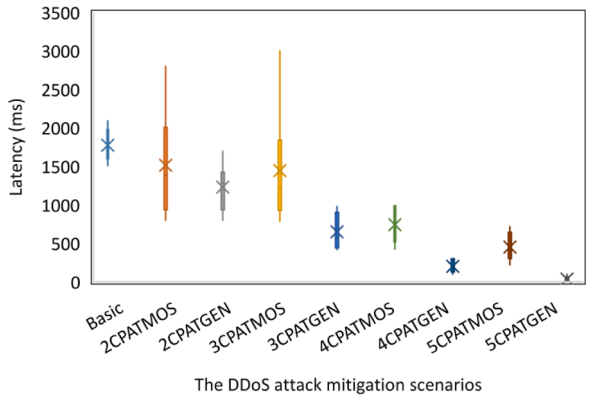

Fig. 15. Delay time for the test scenarios.

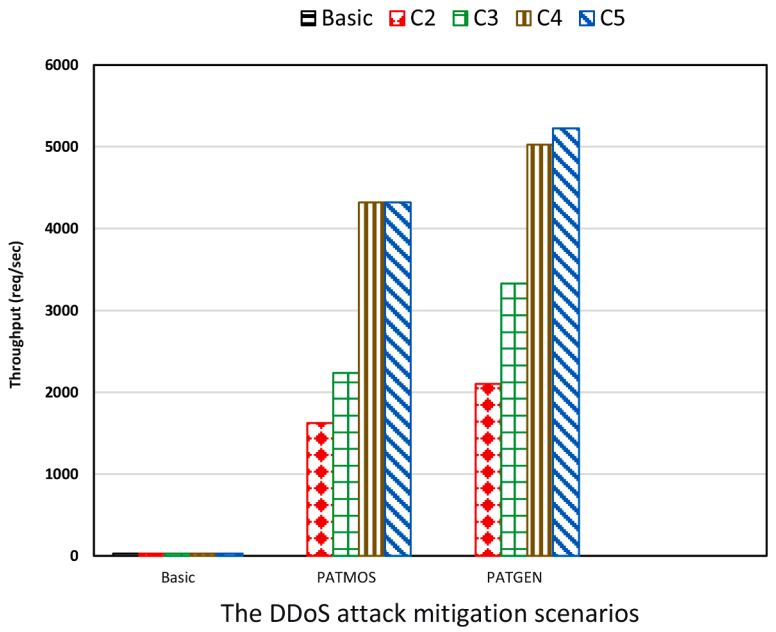

Fig. 16. Average Throughput for the test scenarios.

reduces the latency by 30% on average compared to PATMOS. The reason for this superiority is PATGEN uses high-quality initial population production and as well as the use of efficient and optimal operators that make the algorithm converge faster to the optimal solution.

In general, the calculation of the average of the simulated experiment results, significantly when the network traffic fluctuates highly, is based on the study of Salah [25]. Besides, we monitored CPU utilization on the Linux machine to evaluate the DDoS attack mitigation method's added effect on the POX controllers.

Regarding the limitations of the proposed algorithm, it should be noted that if we divide the network topology into the access, aggregation, and core levels, then the PATGEN is available as an efficient and robust solution against DDoS attacks only in the access level. However, more features and modifications are needed to deploy in aggregation and core levels. One of our assumptions was that DDoS attacks are performed through IP spoofing if we consider static IP configuration for routers. Routers generally use a fixed IP address for a wide area network (WAN), and this method of detection is very efficient, but if the non-static IP configuration of the network is configured, tracking the target source of the attacks is a challenging task that we intend to do in the future.

5. Conclusions and Future work

This paper introduces a protocol for reducing the impact of distributed denial of service attacks in software-defined networks. A gossip-based method is used to detect attacks, and multiple controllers are clustered by optimized and improved the genetic algorithm. Moreover, a robust algorithm is used to select the header controller and increase the genetic algorithm's convergence speed to

determine the optimal number of controllers in each cluster. We have used a load balancer at runtime to reduce overloaded resources due to attacks and increase the network's efficiency. Experimental results demonstrate the effectiveness of our proposed algorithm in the multi-controller software-defined networks. Also, the improved and new operators search the problem space with greater accuracy and thus provide the most optimal clustering of controllers in the least amount of time. In the near future, we will improve our algorithm by optimizing more objectives simultaneously. Furthermore, we intend to combine our method with other intelligent algorithms.

Author Statement

We thank the anonymous reviewers for their constructive comments that helped improve the quality of the article. We also thank the esteemed editor and his cooperation and feedback that helped further the paper.

Declaration of Competing Interest

The authors declare no conflict of interest.

References

[1] Alqahtani F, Al-Makhadmeh Z, Tolba A, Said O. TBM: A trust-based monitoring security scheme to improve the service authentication in the Internet of Things communications. Comput. Commun. 2020;150:216–25. Jan.
[2] Deeban Chakravarthy V, Amutha B. Software-defined network assisted packet scheduling method for load balancing in mobile user concentrated cloud. Comput. Commun. 2020;150:144–9. Jan.
[3] Tran CN, Danciu V. A general approach to conflict detection in software-defined networks. SN Comput. Sci. 2020;1(1). Jan.
[4] Macedo R, De Castro R, Santos A, Ghamri-Doudane Y, Nogueira M. Self-organized SDN controller cluster conformations against DDoS attacks effects. In: 2016 IEEE Global Communications Conference, GLOBECOM 2016 - Proceedings; 2016.
[5] Y. Cao, J. Wu, B. Zhu, H. Jiang, Y. Deng, and W. Luo, "A cross-plane cooperative ddos detection and defense mechanism in software-defined networking," 2019, pp. 231–243.
[6] Jafarian T, Masdari M, Ghaffari A, Majidzadeh K. Security anomaly detection in software-defined networking based on a prediction technique. Int. J. Commun. Syst. 2020;33(14):e4524. Sep.
[7] de Assis MVO, Carvalho LF, Rodrigues JJPC, Lloret J, Proença ML. Near real-time security system applied to SDN environments in IoT networks using convolutional neural network. Comput. Electr. Eng. 2020;86:106738. Sep.
[8] Salah K, Alcaraz Calero JM, Zeadally S, Al-Mulla S, Alzaabi M. Using cloud computing to implement a security overlay network. IEEE Secur. Priv. 2013;11(1):44–53.
[9] Karapoola S, Vairam PK, Raman S, Kamakoti V. Net-Police: a network patrolling service for effective mitigation of volumetric DDoS attacks. Comput. Commun. 2020;150:438–54. Jan.
[10] Bawany NZ, Shamsi JA, Salah K. DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions. Arabian Journal for Science and Engineering 2017;42(2):425–41. 01-Feb-.
[11] Gkountis C, Taha M, Lloret J, Kambourakis G. Lightweight algorithm for protecting SDN controller against DDoS attacks. In: *Proc. - WMNC 2017 10th Wirel. Mob. Netw. Conf.*, vol; 2017. p. 1–6. 2018-Janua.
[12] Wang Y, Hu T, Tang G, Xie J, Lu J. SGS: Safe-Guard Scheme for Protecting Control Plane Against DDoS Attacks in Software-Defined Networking. IEEE Access 2019;7:34699–710.
[13] Krishnan P, Duttagupta S, Achuthan K. SDN/NFV security framework for fog-to-things computing infrastructure. Softw. Pract. Exp. 2019 spe.2761, Nov.
[14] Chen K, Liu J, Martin J, Wang KC, Hu H. Improving integrated LTE-WiFi network performance with SDN based flow scheduling. In: Proceedings - International Conference on Computer Communications and Networks, ICCCN; 2018. vol. 2018-July.
[15] Hussain B, Du Q, Sun B, Han Z. Deep Learning-Based DDoS-attack detection for cyber-physical system over 5G network. IEEE Trans. Ind. Informatics 2020. 1–1, Feb.
[16] Macedo R, Melniski L, Santos A, Ghamri-Doudane Y, Nogueira M. SPARTA: a survival performance degradation framework for identity federations. Comput. Networks 2017;121:37–52. Jul.
[17] Ranganathan S, George A, Todd R, Chidester M. Gossip-Style Failure Detection and Distributed Consensus for Scalable Heterogeneous Clusters. Cluster Comput 2001;4(3):197–209.
[18] Chatterjee M, Mitra A, Setua SK, Roy S. Gossip-based fault-tolerant load balancing algorithm with low communication overhead. Comput. Electr. Eng. 2020;81. Jan.
[19] Kakahama HK, Taha M. Adaptive software-defined network controller for multipath routing based on reduction of time. UHD J. Sci. Technol. 2020;4(2):107. Nov.
[20] "memcached - a distributed memory object caching system." [Online]. Available: https://memcached.org/. [Accessed: 03-Mar- 2021].
[21] Lim S, Ha J, Kim H, Kim Y, Yang S. A SDN-oriented DDoS blocking scheme for botnet-based attacks. In: International Conference on Ubiquitous and Future Networks, ICUFN; 2014. p. 63–8.
[22] Dharma NIG, Muthohar MF, Prayuda JDA, Priagung K, Choi D. Time-based DDoS detection and mitigation for SDN controller. In: 17th Asia-Pacific Network Operations and Management Symposium: Managing a Very Connected World, APNOMS 2015; 2015. p. 550–3.
[23] Mousavi SM, St-Hilaire M. Early detection of DDoS attacks against SDN controllers. In: 2015 International Conference on Computing, Networking and Communications, ICNC 2015; 2015. p. 77–81.
[24] "Apache JMeter - Apache JMeterTM." [Online]. Available: https://jmeter.apache.org/. [Accessed: 03-Mar- 2021].
[25] Salah K. On the deployment of VoIP in Ethernet networks: methodology and case study. Comput. Commun. 2006;29(8):1039–54. May.

Amir Iranmanesh received the B.S. degree from Bahonar University and received his master's degree in information technology engineering from the Graduate University of Advanced Technology in 2017, Iran. he is currently a Ph.D. student in the School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Iran. His research interests include network security, cloud computing, and software-defined networking.

Hamid Reza Naji is Associate Professor of Computer Engineering with a demonstrated history of working in the higher education and industry. He has PhD in Computer Engineering from The University of Alabama in Huntsville, AL, USA. He is currently Associate Professor at The Department of Computer Engineering and Information Technology in The Graduate University of Advanced Technology, Iran.