# Edge-Computing with Graph Computation: A Novel Mechanism to Handle Network Intrusion and Address Spoofing in SDN

**Rashid Amin[1, *], Mudassar Hussain[2], Mohammed Alhameed[3], Syed Mohsan Raza[4], Fathe Jeribi[3] and Ali Tahir[3]**

**Abstract:** Software Defined Networking (SDN) being an emerging network control model is widely recognized as a control and management platform. This model provides efficient techniques to control and manage the enterprise network. Another emerging paradigm is edge computing in which data processing is performed at the edges of the network instead of a central controller. This data processing at the edge nodes reduces the latency and bandwidth requirements. In SDN, the controller is a single point of failure. Several security issues related to the traditional network can be solved by using SDN central management and control. Address Spoofing and Network Intrusion are the most common attacks. These attacks severely degrade performance and security. We propose an edge computing-based mechanism that automatically detects and mitigates those attacks. In this mechanism, an edge system gets the network topology from the controller and the Address Resolution Protocol (ARP) traffic is directed to it for further analysis. As such, the controller is saved from unnecessary processing related to addressing translation. We propose a graph computation based method to identify the location of an attacker or intruder by implementing a graph difference method. By using the correct location information, the exact attacker or intruder is blocked, while the legitimate users get access to the network resources. The proposed mechanism is evaluated in a Mininet simulator and a POX controller. The results show that it improves system performance in terms of attack mitigation time, attack detection time, and bandwidth requirements.

**Keywords:** Software Defined Networking (SDN), edge computing, Address Resolution Protocol (ARP), ARP inspection, security, graph difference.

[1] University of Engineering and Technology, Taxila, Pakistan.

[2] University of Wah, Wah Cantt, Pakistan.

[3] Jazan University, Jazan, Saudi Arabia.

[4] Abasyn University Islamabad, Islamabad, Pakistan.

[*] Corresponding Author: Rashid Amin. Email: rashid.sdn1@gmail.com.

Received: 28 May 2020; Accepted: 25 July 2020.

## 1 Introduction

Traditional networks have always been suffered and seriously affected by attacks such as DDoS, link flooding, packet spoofing, etc. [Modi, Patel, Borisaniya et al. (2013)]. One of the main symptoms of these attacks is traffic congestion is caused by the ARP broadcast storm. In some other attacks such as Denial of Services (DoS), Brute Force, Browser, Shellshock, Botnet [Sharafaldin, Gharib, Lashkari et al. (2018)], etc., the reliability and efficiency of network services are extremely lowered. SDN has become an emerging network control paradigm widely recognized as a management platform. It has recently gained attention due to its efficiency in management tasks, where security issues are most critical. Moreover, it has a central manager to control the entire network that may be overwhelmed from thousands of users' messages and queries from Amin et al. [Amin, Reisslein and Shah (2018)]. Among the important and challenging security issues, Distributed Denial of Service (DDoS) detection, ARP Spoofing, and network intrusion are the most critical. In this context, several solutions and approaches have been developed. Edge computing is an emerging paradigm to deal with these types of problems. Its main feature is that the processing such as analysis and information engineering is performed at the nearest devices from the request generator. It reduces the network latency and bandwidth requirements among different network devices. This paradigm appears as a game changer and it has revolutionized the entire computing mechanism.

SDN being the forerunner is the backbone of network applications. To control and manage the network devices and services by using the abstraction of low-level functionality, it decouples the network control plane from the forwarding plane [Nunes, Mendonca, Nguyen et al. (2014); Shalimov, Zuikov, Zimarina et al. (2013)]. Moreover, it offers valuable support for the complex digital networks in terms of scalability, dynamic computing, and storage requirements. It also provides efficient network control and operation cost-effectively. In SDN, two planes of traditional networking (control and data plane) are effectively separated [Sezer, Scott-Hayward, Chouhan et al. (2013)], as the data plane is left with forwarding mechanism while the control plane is shifted to the controller as shown in Fig. 1. This separation results effectively in a centralized application to deploy networking policies [Amin, Shah, Shah et al. (2016); Hussain and Shah (2018); Hussain, Shah and Tahir (2019)], management tools [Moyano, Cambronero and Triana (2017)], security measures [Dargahi, Caponi, Ambrosin et al. (2017)], etc. Furthermore, it provides network virtualization functionalities [Siddiqui, Escalona, Trouva et al. (2016)], data flow optimization, flexibility, accuracy and consistency in the configuration as compared to the manual configuration of devices in the traditional network [Scott-Hayward, O'Callaghan and Sezer (2013)]. In SDN, according to common practice network security measures and control techniques are deployed at the controller which secures the network users from various types of attacks. Due to the central manager, for the entire network, most of the processing is performed at the controller level and it is an almost overwhelming task. That also creates the latency and bandwidth problem for all users and applications. To handle these limitations, a graph computing-based approach is adopted to secure SDN. In this approach, ARP related processing is shifted to an edge computer that handles and analyzes the ARP requests/replies.

Network security attacks like Link Flooding Attack (LFA) and Distributed Denial of Service (DDoS) [Shin and Gu (2013)] attacks are mostly launched by using the famous Address Resolution Protocol (ARP) spoofing [Abad and Bonilla (2007)] or sometimes the Internet Protocol (IP) spoofing method. In a computer network, ARP packets are mostly used to identify the MAC/IP address translation of the user/system. These ARP packets can be easily altered by an attacker, and as such the IP/MAC address can be modified. Being the prime element (brain) in an SDN, the controller is extremely susceptible to several attacks. ARP spoofing is the most common attack where ARP packets are sent by a malicious node (attacker), resulting in poisoning the network topology. Sometimes this situation leads to a denial of services, serious hijacking, and it results in an entire network failure. Moreover, along with ARP spoofing, link flooding and Man in the Middle attacks are further considered with a brief description in the indexes.

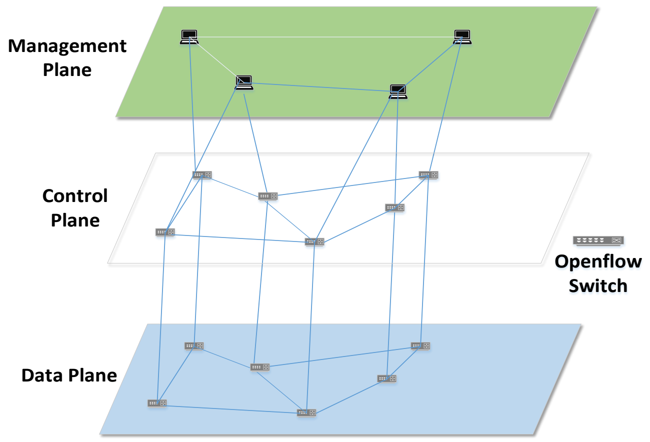


**Figure 1:** Software defined networking overview

To mitigate ARP spoofing attack and network intrusion, several approaches [Masoud, Jaradat and Jannoud (2015); Xing, Zhao and Li (2010); Xu and Liu (2016)] are proposed for SDN. These approaches comprise of various applications that are deployed on the controller. If the network is under attack due to continuous packets arrival at the controller ports then further execution at the controller is severely suffered. In Schneider et al. [Schneider, Bifulco and Matsiuk (2016)], ARP poisoning attack is handled through the SDN paradigm, a custom application is deployed on the controller that checks for the possible ARP attacks. When an attack is launched, the controller may be unresponsive to the entire network, thus, the performance is degraded. In mitigating techniques, a legitimate user whose IP address is used for address spoofing can not restore to the connected state. To mitigate this situation, we propose a graph computing-based network intrusion and address spoofing attack detection and prevention mechanism. In this technique, an edge computer is used to deal with address translation queries. On this edge computer, packets are further analyzed for possible attacking conditions. The edge computer is programmed to check the malicious packets against the legitimate host. A malicious traffic blocking algorithm blocks the specific port after the detection of threats. Moreover, to detect the position of an attacker or intruder in the network, graph computation is performed by using a graph difference algorithm. After getting the exact location of the attacker, the respective port is blocked and the legitimate user is resumed. In this way, the controller is spared from unnecessary processing that enhances its performance and reliability. Our contributions are summarized as follows:

- We identify the potential problems of address spoofing and network intrusion in SDN and their effects on the network.
- We propose an edge computing-based solution in SDN, to process the ARP related traffic on an edge system, instead of the SDN controller that minimizes the influence of the attack on the controller.
- We are original in devising edge computation methods to mitigate address spoofing and network intrusion by taking snapshots of network status at different time intervals.
- We develop algorithms that automatically detect the attacker or intruder location in the network and block its port.
- Our system is developed using event-driven POX controller functions. Simulation results indicate the reduction in attack detection and mitigation time as well as normalized overhead.

The remaining parts of the paper are arranged as follows. The related work is placed in Section 2. Problem definition containing normal execution and attack condition is explained in Section 3. The proposed solution consists of graph computation and graph difference methods are presented in Section 4. In Section 5, simulation setup and performance evaluation are discussed and Section 6 concludes the paper.

## 2 Related work

ARP cache poisoning attacks in Local Area Networks (LANs) are prevented using two different scenarios as discussed in Alharbi et al. [Alharbi, Durando, Pakzad et al. (2016)]. The first one is an SDN DYN, where the network host is assigned a dynamic IP address using the DHCP protocol. It looks for the mapped IP-MAC address of the gateway. Then, it inspects all the ARP replies and drops the replies not matching the recorded IP address to the MAC address table. The second scenario is an SDN STA, which indicates the assignment of a static IP address. After the construction of the forwarding tables, the controller records each sent packet and maps its IP address to the MAC address which is obtained from the forwarding tables. Afterward, the controller checks if there is an ARP response against the recorded pairs.

In the case of DDoS attacks, there is a great asymmetry between the inflows and outflows of the victims [Masoud, Jaradat and Jannoud (2015)]. The proposed approach is based on SDN's flow steering capabilities consisting of two main parts. First, it installs control rules in order to capture regular and anomalous flows in the network. Then, it measures the available resources in the network (including the effective use of all available TCAM inputs) to balance the coverage and granularity of attack detection by coordinating the monitoring rules on all switches. Thus, it can quickly locate potential DDoS victims and attackers [Xu and Liu (2016)] in this network. However, with all the encouraging results, it still needs to be evaluated for real DDoS attacks.

Ma et al. [Ma, Ding, Yang et al. (2016)] discuss the security issues in large scale cloud data centers based on the SDN paradigm. Man-in-the-Middle and Denial of Service attacks (DoS) are considered to be resolved. To calculate the probability of occurrence of an attack in Virtual Machine (VM), the Bayesian theorem is used. When an attack is launched, prevention techniques are adopted to mitigate it. However, this mechanism lacks in the detection of an attack, if the prediction algorithm does not respond timely and the attacker succeeds to launch an attack, it results in the interception of network traffic between a client and the controller [Wang, Li, Jiang et al. (2016)].

Cox et al. [Cox, Clark and Owen (2016)] address the ARP spoofing attacks in a traditional network as well as in SDN. SDN offers more control and security over the entire network than a traditional network. To protect the network from ARP spoofing attacks, Network Flow Guard for ARP (NFGA) module is proposed by augmenting the controller in order to detect and mitigate the address spoofing attacks which are triggered by the unauthorized users as shown in Fig. 2. NFGA observes Dynamic Host Configuration Protocol (DHCP) to build a table that contains MAC:IP:Port mapping plus other similar information to get ARP traffic. In this way, NFGA detects any address spoofing attack in the network.

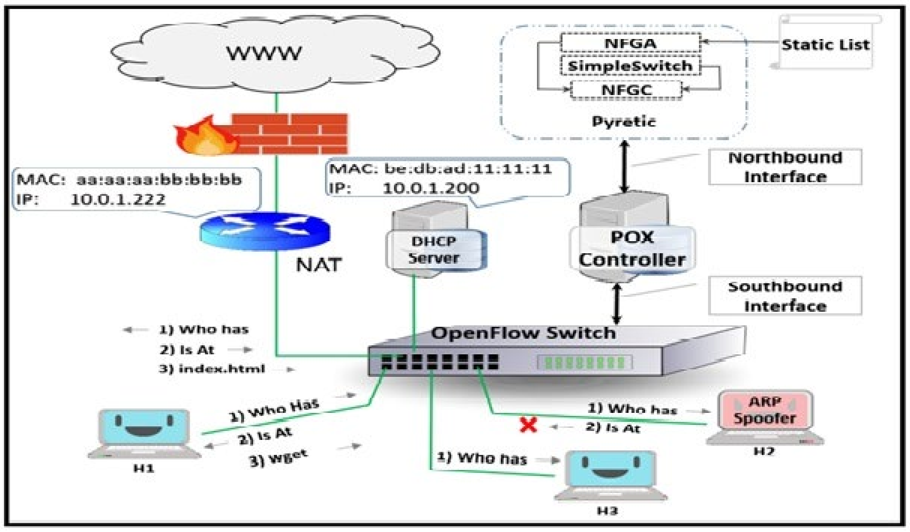


**Figure 2:** Network flow guard for ARP

Rengaraju et al. [Rengaraju, Ramanan and Lung (2017)] discuss the DoS attack in cloud and large scale networks. The authors propose a distributed firewall with Intrusion Prevention System (IPS) for software designed clouds. In this mechanism, the IPS module observes the network traffic at the frame level and detects the anomalous packets. If anomalous packets are found, the Openflow switch informs the IPS. IPS and firewall analyze these packets for some kind of security threats. If some malicious activity is found then IPS mitigates the intrusion by indicating alarm, dropping these packets, and taking severe actions against the malicious node. Finally, it blocks the port associated with this malicious activity. The entire system is shown in Fig. 3.

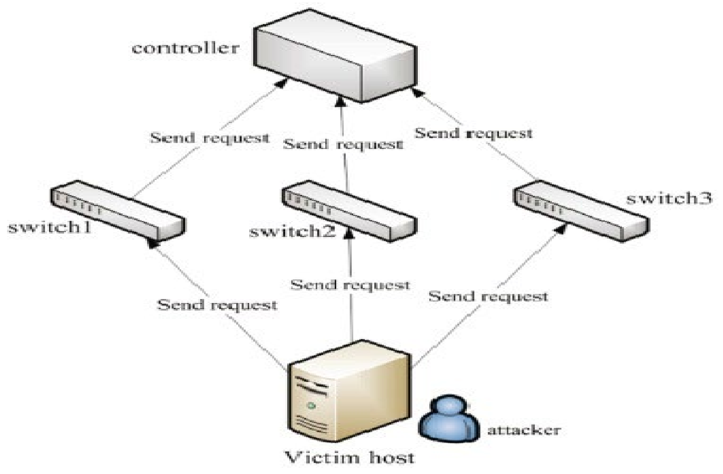


**Figure 3:** DoS attack and man-in-middle attack in the data center

Hameed et al. [Hameed and Ahmed Khan (2018)] propose a lightweight and efficient collaborative DDoS scheme for SDN. A secure controller-to-controller (C-to-C) communication protocol for multiple SDN controllers is designed. These multiple controllers are mapped with several autonomous systems. This scheme provides many benefits, i.e., blocking the malicious network flows inside the network and informing the neighbor Autonomous System (AS) about the attack condition inside the domain.

Fan et al. [Fan, Xiao, Nayak et al. (2019)] discuss the security threats and issues in SDN. They indicate the need for a mechanism that can evaluate the security measures of the network in advance so that protection against attacks can be made early. This approach handles the attacks, i.e., scanning attack, Openflow flooding attack, ARP attack, and switch comprised attacks. For these four attacks, there are twelve characteristics offered to mitigate these attacks. By using these characteristics, different states of the network are measured from time to time by using a multi-observation Hidden Markov Model. This model is deployed in the Ryu controller with Openflow switches to assess the network status by using different test scenarios.

The existing studies discussed above and in Deng et al. [Deng, Gao, Lu et al. (2017); Kalkan, Gur and Alagoz (2017); Khan, Gani, Wahab et al. (2016)] conclude that address spoofing and network intrusion are not well addressed in past. In most of the solutions, the proposed techniques are deployed on the SDN controller that is already considered as a single point of failure. Therefore, it is required to find such a solution that can handle the attack condition separately without influencing the controller's performance.

## 3 Problem statement

The structure of an ARP packet is shown in Fig. 4 where the Ethernet frame contains the ARP payload. This payload encompasses the MAC addresses, Target Hardware Address (THA) and Sender Hardware Address (SHA). Similarly, each frame includes the two IP addresses, the first is Sender Protocol Address (SPA) and the second one is Target Protocol Address (TPA). The operation field denotes that the packet is a request for a response.

If a node wants to get an IP address of a node's MAC address, then the ARP request is sent by setting SHA to its current MAC address and SPA to its IP address. TPA field of the frame is set as the IP address of the target node and THA field has a dummy value 00:00:00:00:00:00. When the target node receives this packet, it sets the THA field to its own MAC address and generates a response to the sender. The main problem with ARP is that it is a stateless protocol, it means, it considers each request or reply individually regardless of past communication. Resultantly, a node accepts packets for gratuitous ARP without any corresponding request information. There is no method to check the authenticity of the ARP request or response or any integration of provided information.

In SDN, network intrusion and address spoofing attacks are not well addressed as observed in the related work. These attacks affect the Controller ARP Table (CAT) by inserting the wrong information, resulting in an entrance of the unauthenticated packet to the restricted portions of the network. When an attack is launched successfully, it can easily poison the network topology that is a basic building block for SDN core elements. Due to this poisoned network topology, different network applications and SDN services may be misconfigured. Sometimes, this condition leads to a severe security threat or network hijacking. Some other attacks, i.e., Man-in-Middle and DDoS attacks are also caused by address spoofing. The security of the entire network is on risk, thus, network performance is degraded.

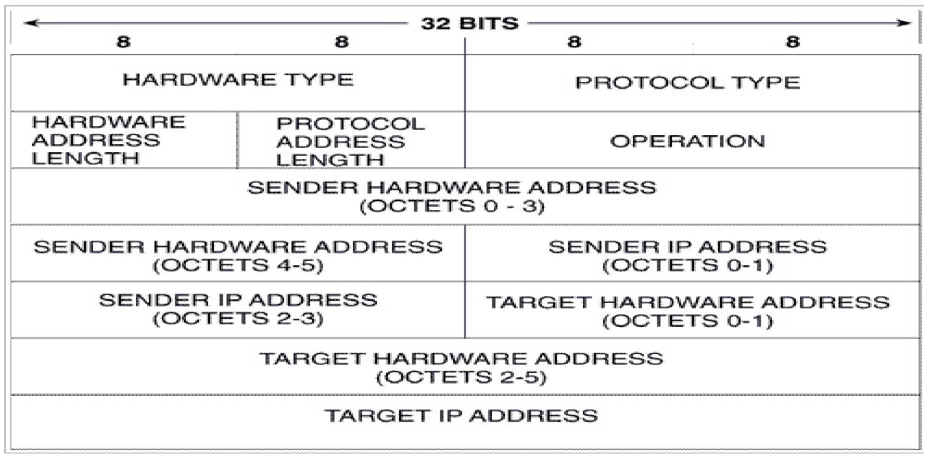

32 BITS
8
8
8
8
HARDWARE TYPE
PROTOCOL TYPE
HARDWARE ADDRESS LENGTH
PROTOCOL ADDRESS LENGTH
OPERATION
SENDER HARDWARE ADDRESS (OCTETS 0 - 3)
SENDER HARDWARE ADDRESS (OCTETS 4-5)
SENDER IP ADDRESS (OCTETS 0-1)
SENDER IP ADDRESS (OCTETS 2-3)
TARGET HARDWARE ADDRESS (OCTETS 0-1)
TARGET HARDWARE ADDRESS (OCTETS 2-5)
TARGET IP ADDRESS

**Figure 4:** Structure of an ARP packet

From literature, it is obvious that the different SDN controllers, i.e., Floodlight [Wallner and Cannistra (2013)], Open Daylight [Medved, Varga, Tkacik et al. (2014)], ONOS [Berde, Gerola, Hart et al. (2014)], etc., can be easily poisoned by network intrusion and address spoofing attacks. If the topology of the network is poisoned, all consequent network applications and services are misconfigured. For example, the SDN controller may experience the situation of a Man-in-the-Middle attack or black hole attack, if the routing services are not working properly. Similarly, intruders can easily get into the network to steal or modify critical files. The problem is expressed in detail by taking an example of an enterprise network as shown in Fig. 5. Assuming, some organization having different site offices, at one of the sites there is a network which comprises five

SDN switches, i.e., OF1, OF2, OF3, OF4, and OF5. These switches are connected to each other and to the SDN controller as shown in Fig. 5. There are ten users ( user1 to user10) using PCs ( PC1 to PC10) having IP address range (12.0.0.1 to 12.0.0.10) are connected to this network. Each switch has exactly two users connected, i.e., PC1 and PC2 are connected to OF1. We show two possible conditions of the network, first is the normal network operation when the network is running normally and the second one is attacking condition when an attacker has launched an attack on the network.

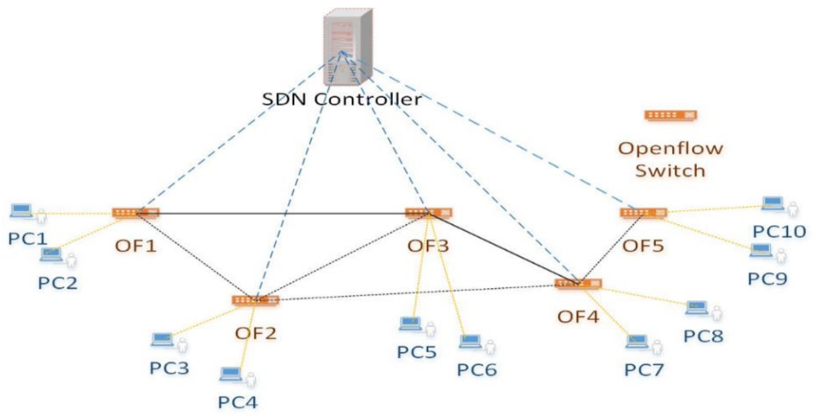


**Figure 5:** Ideal condition

### *3.1 Normal network operation*

In the normal network operation, user1 having IP address 12.0.0.1 wishes to communicate with user8 having an IP address 12.0.0.8 as shown in Fig. 5. For this communication, user1 needs MAC address of user8 which is obtained by sending ARP packet to the SDN switch OF1. OF1 checks the flow entries to find the requested IP address. If it exists then it is sent to the requester, otherwise, an ARP packet is forwarded to the controller. If the controller does not have the MAC address, the controller broadcasts the packet to all the switches to get the MAC address. The packets reach every switch in the network and finally, it gets the required MAC address. The controller sends the MAC address to the ARP request generator. Moreover, the SDN controller implements the flow rules on the respective network devices to forward packets accordingly. After this, packets move to the corresponding network device and get the MAC address of the destination node. In this way, the normal system runs very well.

### *3.2 Attack condition*

Suppose, an attacker using the Kali Linux system [Beggs (2014)] launches broadcast gratuitous ARP packets for the user5 with an IP address (12.0.0.5). Gratuitous ARP packets are the broadcast packets used to advertise any alteration by the network device in their MAC or IPv4 address. By using these gratuitous ARP messages, the attacker embeds the IPv4 address of user5 and captures all the network traffic of user5 as shown in Fig. 6. Kali Linux is an OS in which some hacking tools are pre-installed and the hackers use this OS for hacking purposes. In our case, Kali Linux is used to launch an ARP spoofing attack which leads to network intrusion and Man-in-the-Middle attacks.

The network is hijacked by using a single attack, due to this successful attack, network information stored on the controller is poisoned and the attacker can sniff the network traffic. After getting the network information, the attacker can easily launch a DDoS attack and can manipulate it as an intruder. These attacks keep the controller busy, consequently, other users cannot access the controller.

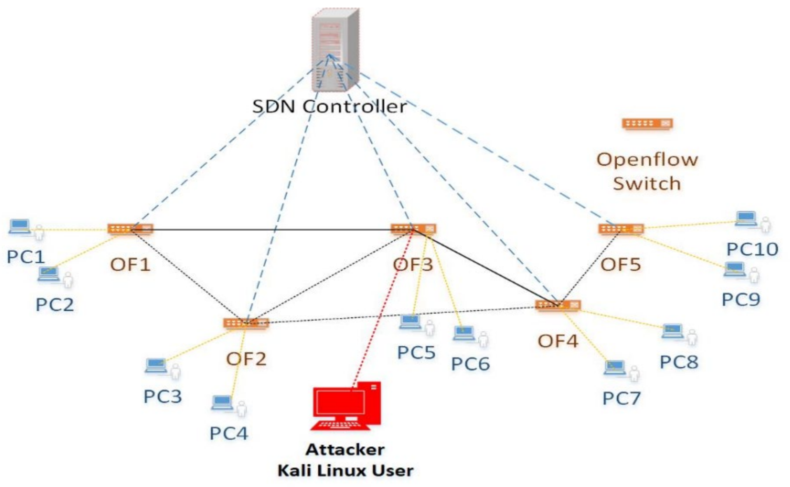


**Figure 6:** Attack condition

In this way, the entire network is affected and the performance is degraded to an extent that even legitimate users cannot interconnect the SDN controller. An intruder easily gets access to the network resources and can affect critical files and configurations. Apart from the loss of business, these situations may lead to damage to the client's confidence and organization reputation. To address this problem, we propose an edge computing-based solution in which an edge system is used for the analysis of ARP traffic to detect possible attack conditions. Moreover, graph computation is employed to further detect the location of an attacker or intruder.

## 4 Problem formulation

To deal with the problems discussed in Section Problem Statement, automatic attack detection, and mitigation techniques have been proposed that are based on edge computing as well as graph computation. In this mechanism, an edge system is deployed in the network in which the proposed algorithms are programmed. Initially, the Controller ARP Table (CAT) maintains all the network device's information by mapping IP and MAC addresses. We collect data about the neighbors, hops, and links from the forwarding tables to construct the entire network topology. After constructing network topology, the corresponding graphs are generated for the topology. To detect an attack condition, custom methods are used to indicate malicious traffic activities. The graph difference algorithm detects the position of an attacker as well as a legitimate user. In this way, the attacker is blocked for further processing, and the legitimate user is resumed.

The system designed is shown in Fig. 7, which contains an edge computer with some custom development. In the proposed solution, the first component gets topology information and constructs the graph. The second one installs the flows on the devices to forward ARP traffic to the edge computer. The third component analyzes the address spoofing and intrusion-related activities and the fourth component performs graph computation to detect the position of the attacker or intruder.

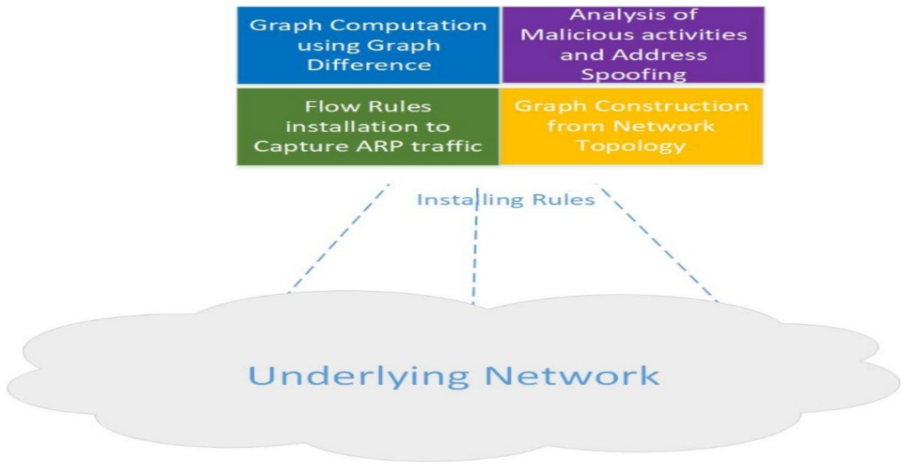

Graph Computation using Graph Difference
Analysis of Malicious activities and Address Spoofing
Flow Rules installation to Capture ARP traffic
Graph Construction from Network Topology
Installing Rules
Underlying Network

**Figure 7:** Overview of the proposed solution

### 4.1 Graph construction for network topology

When the network is established, Openflow network devices exchange their link-state information with the controller. Then network topology is built, flow rules are installed to redirect the ARP traffic towards the edge computer for analysis. The controller is programmed to install respective flow rules at the network devices.

### 4.2 Flow rule installation

The link-state information is shared among the switches, hosts, and controller after the establishment of the network, then flow rules are installed on the switches. These special flow rules direct the ARP traffic towards the edge computer. To implement these flow rules on all network devices, the SDN controller installs flow rules on the switches. After this, ARP traffic is forwarded to the edge computer. The pseudo-code describes the flow rule installation as depicted in Algorithm 1.

**Algorithm 1: Installation of Flow rules for ARP traffic**

```
Input: n nodes, m switches, controller
Output: Path to destination
1: n=total number of switches
2: m=total number of nodes
3: Controller get entire network topology from n,m
4: Controller examines the network traffic
5: if (Pkt.dest=broadcast) or (Pkt belongs to ARP) then
```

```
6:      Pkts are moved to an Edge computer for further analysis
7: endif
8: else
9:   Pkts are transferred to controller || Forwarded according to flow rules
10: Controller installs flow rules accordingly
11: endif
```

### *4.3 Analysis of ARP traffic and malicious activities*

When ARP traffic arrives at the edge computer, an address spoofing attack is detected by examining ARP request packets from a particular host. At the first step, we confirm that either packet belongs to their network or it does not by sending an ARP request. If ARP request is generated for this network, appropriate actions are taken which are explained in the following situations:

In the first situation, when an ARP packet is forwarded by a user in the network, it arrives at the Openflow switch where flow entries of the switch are checked for a possible match. If there is a mismatch, then the packet is forwarded to the edge computer for further analysis for the possible attack condition as shown in Fig. 8. Now, if these packets belong to our network, then edge computer responds with the appropriate address, otherwise, it is discarded. The following example illustrates the procedure in detail:

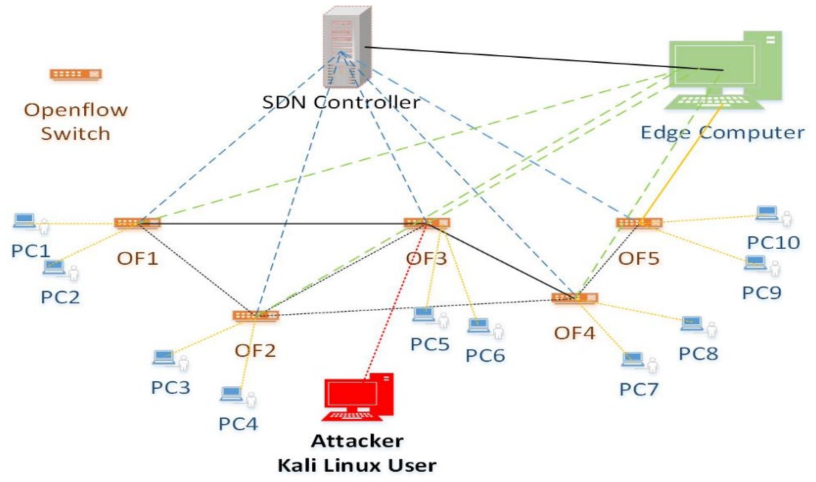


**Figure 8:** Architecture of the proposed mechanism with an edge computer

Assume, PC1 with IP address 12.0.0.1 initiate ARP packets to communicate with PC6 as shown in Fig. 8. Openflow switch OF1 checks its flow table to find the requested entry. If it does not find, then the packets are forwarded to an edge computer according to the flows installed on the switch. The edge computer analyzes the respective packet whether it pertains to this network or not. There may be two conditions, first, this packet belongs to this network then proper actions are taken by sending with ARP reply message to PC1. Now the sender is authenticated, and further communication is possible. As a second case, if packets belong to some other network and are confirming through IP and MAC addresses of the packet with CAT then it is dropped. If these malicious ARP packets are not dropped, then these can be used very easily by an attacker to launch an attack.

Hence, we are considering the second situation where an attacker impersonates as a legitimate user and sends an ARP packet by spoofing the IP address of some other user. These packets are analyzed by checking the IP address with already recorded MAC to IP mapped table. If the respective entry for IP address is present, the source MAC address of the packet is mapped to the MAC address table. In the case of mismatching, the packet is dropped. Moreover, this malicious node continuously sends packets to the switch which blocks the corresponding controller port. A graph computation mechanism checks the attacker's position in the network. All these steps are clearly shown in Algorithm 2.

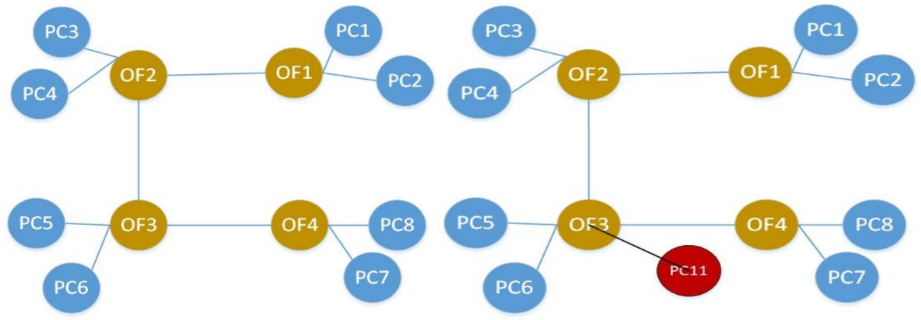
**Algorithm 2: Detection of network intrusion or ARP spoofing**

```
Input: packets containing ARP request or broadcast Address, k number of nodes
Output: Indicate Address spoofing malicious activity
1: CAT [ ] is initialized
2: for j in range (1,2, 3..., k) do
3:  Store MAC Address and IP address in CAT[ ]
4: end for
5: if (Pkts.src is not in CAT[ ] table and Pkts.dst not in CAT[ ] table ) then
6:      discard the pkts
7: else
8:  if (Pkts contain ARP request) then
9:     check whether IP and MAC Addresses matching in CAT[ ] table or not
10:   send respective IP/MAC address
11:  endif
12:  if (Pkts.dst==FF:FF:FF:FF:FF:FF && Pkts.Src in CAT[ ] table) then
13:     Implement flows for broadcasting
14:  endif
15: endif
```

### *4.4 Graph computation for detection of attacker or intruder location*

Graphs are an important tool to manipulate network topology for different types of operations, i.e., policy formation, shortest paths finding, security, etc. We use graph computation for the identification of intruders or attackers. We compute the graph from the network topology in which edges represent the links and nodes represent the vertices. By taking these two components, we take the snapshot of the network at different time intervals to get network status. It is used to discover network attacks by identifying the position of the attacker. For this purpose, the snapshots of the network are taken initially when the network is established and whenever topology is updated. If any malicious activity occurs, that is identified by Algorithm 2 then graph computation is used to spot the attacker's position. For example, in Fig. 9(a), an initial network topology is shown where multiple nodes are connected to four switches. After some time, the network topology is updated when another node PC11 is connected to the network as shown in

Fig. 9(b). Graph difference technique identifies the location of an attacker or intruder as well as a legitimate user.



(a) Initial Network Topology (b) Network topology after an Attacker's connection

**Figure 9:** Representation of topology change using Graphs

The following definitions are used to compute graph differences from the network snapshots captured at different time intervals.

### *4.4.1 Definition a*

A triple $M=(P, R, \mu)$ is described by the directed and labeled graph and signifies the network topology for the time interval$\delta t$, where:

• P represents the finite group of vertices

• $R \subseteq P \times P$ is the group of edges where g (k, j) represents the direction of the edge from k to j

• $\mu : P \rightarrow L_n$ is a function assigning unique labels to each vertex in M such that $\mu(j)=\mu(k)$ where $j, k \in P$ and $j=k$

In this description, M represents the communication network topology having a group of symbols indicating a unique node. A directed edge g (j, k) represents the directed link or channel for network traffic from node j towards k.

### *4.4.2 Definition b*

A bijective function $f : P -> V$ is a graph isomorphism from $M=(P, R, \mu)$ to $M`=(P`, R`, \mu`)$ if

• $\mu(a)=\mu`(f(a))$, a P

• For all P, Q P, the edge g=(P, Q) R, if the edge g0=(f` (P ), f`(Q)) R

### *4.4.3 Definition c*

A specified graph $M=(P, R, \mu)$ represents all available edit functions $\theta$on M as follows:

• $(a \rightarrow S`)$, $a \in R$: represent the deletion of a node 'a' from M and all edges in M indices to a.

• $(S` \rightarrow a)$: represents the insertion of a node 'a' into M with specific node label $\mu(a)$ $L_n$ (a)

• $(g \rightarrow S`)$, $g \in R$: represents the deletion of an edge g from M

• (S`→g), g=(P, Q) and P, Q∈P: represents the Insertion of an edge g among two nodes P and Q in M

*4.4.4 Definition d*

Assuming a graph M = (P, R, μ) with an edit operation θ on M, the edited graph θ (M) becomes the graph θ (M)=($P_\theta$ , $R_\theta$ , $\mu_\theta$) where:

$$P_\theta = \begin{cases} \text{P - }\{p\} & \text{if } \theta = (p \rightarrow S') \\ \text{P}\cup\{p\} & \text{if } \theta = (S' \rightarrow p) \\ \text{P} & \text{Otherwise} \end{cases}$$

$$R_\theta = \begin{cases} \text{R}-\{g\} & \text{if } \theta = (g \rightarrow S') \\ \text{R}\cup\{g\} & \text{if } \theta = (S' \rightarrow g) \\ \text{R} & \text{Otherwise} \end{cases}$$

$$\mu_\theta = \begin{cases} \mu\theta{=}\mu|\text{a } -\{P\} & \text{if } \theta = (a \rightarrow S') \\ Ext.of\theta toP \text{ U } \{\text{p}\} & \text{if } \theta = (S' \rightarrow a) \\ \mu & \text{Otherwise} \end{cases}$$

*4.4.5 Definition e*

Assuming a graph M=(P, R, μ) and there are several edit operations performed in sequence φ=(θ1 , θ2 , ..., θk), k≥1, the resultant succinct graph φ(M) becomes:
φ(M)=θk (. . . .. θ2(θ1(M )). . . .)

*4.4.6 Definition f*

Assume two graphs M=(P, R, μ) and M`=(P`, R`, μ`) are given and with φ represents the number of edit operations performed in a sequence on M in such a way that φ(M ) is a graph isomorphic to M` the edit distance d(M, M`) between graphs M and M` becomes the minimum sum of edit costs. d(M, M`)=C`(φ)

*4.4.7 Definition g*

Suppose a graph M=(P, R, μ) represents the network topology at certain time t, and assume M`=(P`, R`, μ`) indicates the similar network topology at different time interval t1 where t1=t+δt. The edit distance of the network d(M, M`) is described by:

d(M, M`)=|P|+|P`|−2|P ∪ P`|+|R|+|R`|−2|R∩R`|

The above equation shows the edit distance representing the change in network topology and at different time interval δt. Edit distance shown in the above equation d(M, M`) is nearly bounded below by d`(M, M`)=0 when M and M` are isomorphic to each other (i.e., there is no alteration), and above by d(M, M`)=|P|+|P`|+|R|+|R`| when M∩M`=0, the case where the networks are completely altered.

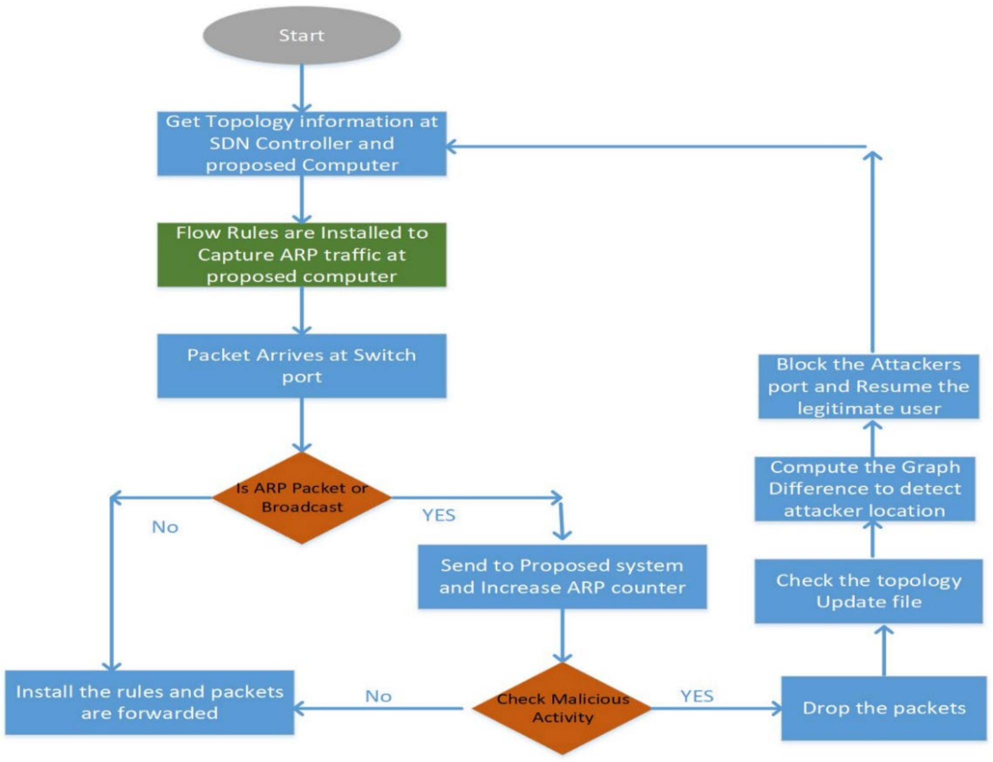

Start
Get Topology information at SDN Controller and proposed Computer
Flow Rules are Installed to Capture ARP traffic at proposed computer
Packet Arrives at Switch port
Is ARP Packet or Broadcast
No
YES
Send to Proposed system and Increase ARP counter
Check Malicious Activity
No
YES
Install the rules and packets are forwarded
Drop the packets
Check the topology Update file
Compute the Graph Difference to detect attacker location
Block the Attackers port and Resume the legitimate user

**Figure 10:** Flow chart for the proposed system

If any malicious activity is reported, or continuous ARP packets are arriving at the controller then graph difference algorithm is executed to detect the change in topology. In this way, the location of the attacker or intruder is identified properly, and subsequent actions are performed to block the respective port connecting that malicious node. Thus, the legitimate user can access the resources with an updated IP address.

In Fig. 10, the flow chart indicates the actual processing of the proposed system. In this system, the network topology is obtained, and the graph is constructed. Afterward, the flow rules are installed on the network devices to redirect ARP traffic towards the edge computer. The edge computer analyzes the malicious traffic and graph computation is performed to indicate the position of an attacker or intruder.

**Algorithm 3: Graph matching algorithm**

```
Input: An undirected Graph M, matching of graph M' on graph M
Output: An augmenting path P is found in graph M or empty route
1: function search augmenting route( M, M' ) : P
2:    T ← blank forest
3:    Uncheck all nodes and links in M, check all link of M'
4: while there is an exposed vertex c
5:    form a singleton type tree {c}, add this singleton tree to F
```

```
6:  end while
7:  while there exists an unchecked node c in forest T having a distance( c, root( c
) ) even do
8:    while an unchecked link l={c, d} exist do
9:      if d node is not the part of the forest T then
10:         // d is matched, l and d's matched links are added to T
11:         x ← node matched to d in M'
12:         add edges {c, d} and {d, e} to the tree of c
13:       else
14:         if distance(d, root(d)) is odd then
15:           // do nothing
16:         else
17:           if root(c)=root(d) then
18:             // Indicate an augmenting path in forest T { g }
19:             P ← path (root(c) → ... → d) → (e → ... → root(e))
20:             return P
21:           else
22:// arrange a bloom in graph M and search for the respective route in the respective
      graph
23:             B ← bloom made by g and edges on the path c → d in T
24:             M", M"' ← contract M and M, by B
25:             P' ← search augmenting path( M", M"')
26:             P ← lift P' to M
27:             return P
28:         end if
29:       end if
30:     end if
31:     mark edge e
32:     end while
33:     mark vertex v
34:  end while
35:  return an empty path
36:  end function
```

## 5 Performance evaluation of proposed solution

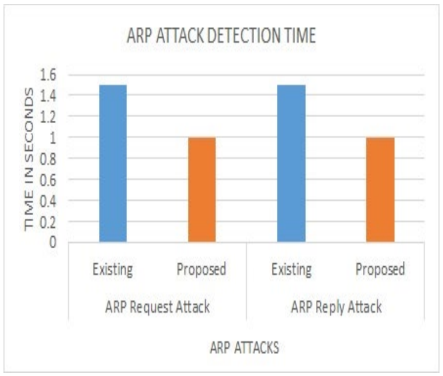
In this section, we discuss the simulation setup and performance evaluation factors with results. In order to evaluate the proposed solution, we performed experimentation on Ubuntu Operating System with Virtual Machine having 6 numbers of cores and 8 GB of RAM with hypervisor server [Huang, Griffioen and Calvert (2014)]. The hypervisor consists of 16 GB of RAM with 32 logical cores. A famous simulation tool Mininet [De Oliveira, Schweitzer, Shinoda et al. (2014)] is used which provides several features to simulate Openflow switches functionality and operations. This tool provides a simulation-based environment as well as a real-time network performance evaluation. POX SDN

controller is used along with Mininet to install network and security policies on the network devices. Several scenarios are formed by using multiple SDN switches, hosts, and other nodes. These nodes are connected using various network topology models and attackers are also included in the topology to verify the system security and stability.
Our initial proposed topology is shown in Fig. 5 which consists of one SDN controller, 5 SDN switches, and at least 10 users. We increased the number of SDN switches to 50 gradually and the number of users to 150 for our experimentation and analyzed the performance and efficiency of the system. An attacker node having IP address (12.0.0.11) is also part of the system, where "Kali Linux" [Beggs (2014)] operating system is installed. Kali Linux is mostly used by a hijacker for attacking purposes. The attacker is used to generate ARP request packets and other types of spoofed packets to poison the network topology. Our edge computer is connected to the controller for the data exchange and flow rules installation on the SDN switches. To evaluate the performance of the proposed system, we measured several parameters including successful packet delivery, attack detection and mitigation time, bandwidth and round-trip time calculation, and normalized overhead using different attack scenarios. Several attacks e.g., ARP request attack, spoofed ARP request and ARP reply attacks are launched in the system.

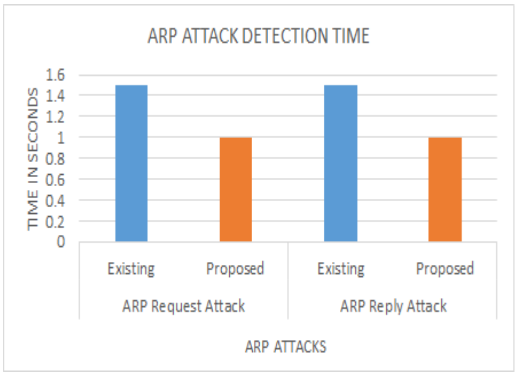
- **Attack Detection and Mitigation Time:** It is the total time in which an adversary user or party launches an attack on the network and the SDN controller detects the attack. Mitigation time is the time to resolve the attack after the detection of the attack.
- **Bandwidth and Round Trip Time (RTT):** Bandwidth factor indicates the amount of bandwidth available when attacks are launched and RTT is the amount of time it takes for a packet to be sent plus the amount of time it takes for an acknowledgment.
- **Successful Packet Delivery Percentage:** It is the ratio of the total number of packets that are delivered successfully to the total number of packets initiated from a source when an attack is launched.
- **Normalized overhead:** It is the total number of packets transmitted, divided by the total number of packets successfully received at the destination within a defined time interval.

## *5.1 Attack detection time*

Fig. 11(b) shows the comparison of attack detection time for the proposed approach and the existing technique. Based on the results, we can conclude that our proposed approach performs better against malicious attacks than the existing approach. In this experiment, we used 20 hosts to test the proposed approach. Due to the policy of port blocking, while detecting any intruder or fake user by using ARP spoofing techniques, it has resulted in the minimization of traffic load of the network. This increases 100% bandwidth of the links between the hosts. The bandwidth per user is given as follows:
Bandwidth per user=(Total bandwidth)/(Number of users)
However, our proposed algorithm blocks intruder or hacker early, so the bandwidth of the links is efficiently utilized which results in an increase of available bandwidth per user which is computed via the below equation.
*Bandwidth per user=(Total bandwidth)/(Number of users-Intruder or hacker)*



(a) Attack Mitigation Time



(b) Attack Detection Time

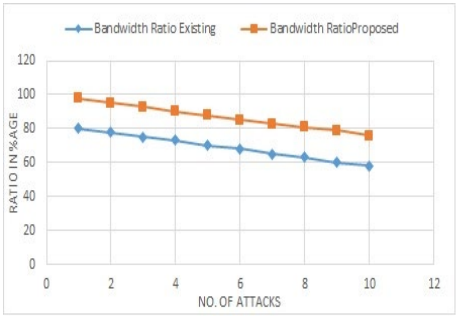
**Figure 11:** Attack detection and mitigation time

### *5.2 Attack mitigation time*

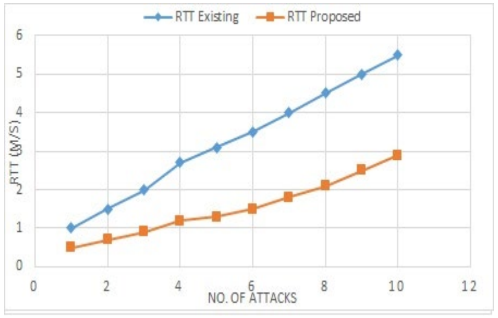
Fig. 11(a) shows ARP attack mitigation time in the case of the proposed approach and the existing approach. The results suggest that the proposed approach takes less time to mitigate attacks as compared to the existing approach. This is because the proposed approach implements the ARP mitigation strategy more efficiently and effectively. When an attack is launched by an adversary party then the proposed technique identifies the malicious activity by analyzing the ARP traffic timely. After the identification of the malicious user, the respective port is blocked to secure the entire network.

### *5.3 Bandwidth and RTT calculation*

The bandwidth of the network is a key component of network traffic and transmission. Network attacks mainly affect the network bandwidth for all users. In an ARP attack, the network server is kept too busy to answer the queries of the legitimate users. In Fig. 12(a), bandwidth consumption for a different number of attacks is shown. From Fig. 12(b), it is observed that latency is on the lower side for the proposed approach than the existing approach. It indicates the less RTT for the proposed approach. It is because ARP traffic arrives at the edge computer that is examined for possible threat conditions.



(a) Available Bandwidth Ratio

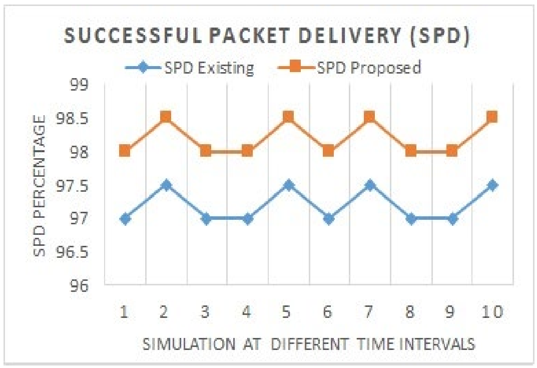


(b) Round Trip Time (RTT) calculation

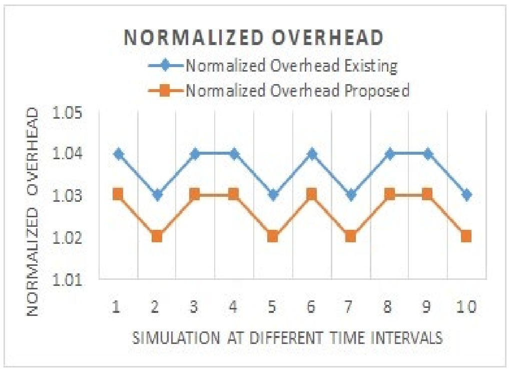
**Figure 12:** Available bandwidth ratio and RTT

### 5.4 Successful packet delivery percentage

Fig. 13(a) shows successful packet delivery (SPD) percentage at different time intervals in case of the proposed and existing approaches. The results indicate that a successful delivery percentage is much better in the case of the proposed solution as compared to the existing mechanism. This is because when an attack is launched then our system automatically detects the attack and minimizes its effect on the system.
If the attack is launched in the network, then our system detects the attack and locates the host in addition to blocking the desired port. Due to this mechanism, a large amount of unnecessary traffic is minimized and SPD is increased.



(a) Successful packet delivery



(b) Normalized overhead

**Figure 13:** SPD and normalized overhead

The normalized overhead increases with the increase in SPD percentage in the case of our proposed approach and decreases with the decrease in SPD percentage. If fewer packets are lost due to the ARP attacks then normalized overhead decreases otherwise it increases. The results in Fig. 13(b) show that normalized overhead in the case of our proposed approach is lower than the existing approach in all instances of time. Although our approach takes extra time to detect the attacks by analyzing the ARP packets, yet due to the effective mitigation technique, the attack is detected and mitigated very quickly. Therefore, the normalized overhead is minimum than the existing approach. This ensures that the proposed approach performs well to detect and mitigate ARP attacks.

## 6 Conclusion

In communication networks, most of the attacks are launched by spoofing the ARP packets and poisoning the network topology using the ARP spoofing method. ARP spoofing and network intrusion are the most common attacks that affect network performance very badly. Our proposed solution comprises an edge computer and customized algorithms to mitigate the ARP spoofing attack and network intrusion. We have used graph computation to locate the attacker or intruder and immediately blocked its communication port. In this way, the entire network is secured and the legitimate users gained access to the network resources. Simulation results prove that the proposed technique is efficient and competent for attack detection and mitigation. In the future, we wish to consider other attacks, i.e., Distributed DoS (DDoS), Man in the Middle attacks with distributed control planes where multiple controllers are deployed.

**Funding Statement:** The author(s) received no specific funding for this study.

**Conflicts of Interest:** The authors declare that they have no conflicts of interest to report regarding the present study.

## References

**Abad, C. L.; Bonilla, R. I.** (2007): An analysis on the schemes for detecting and preventing ARP cache poisoning attacks. *27th International Conference on Distributed Computing Systems Workshops*, pp. 60-60.

**Alharbi, T.; Durando, D.; Pakzad, F.; Portmann, M.** (2016): Securing ARP in software defined networks. *IEEE 41st Conference on Local Computer Networks*, pp. 523-526.

**Amin, R.; Reisslein, M.; Shah, N.** (2018): Hybrid SDN networks: a survey of existing approaches. *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3259-3306.

**Amin, R.; Shah, N.; Shah, B.; Alfandi, O.** (2016): Auto-configuration of ACL policy in case of topology change in hybrid SDN. *IEEE Access*, vol. 4, pp. 9437-9450.

**Beggs, R. W.** (2014): *Mastering Kali Linux for Advanced Penetration Testing*. pp. 20-40. Packt Publishing Ltd. https://www.packtpub.com/networking-and-servers/mastering-kali-linux-advanced-penetration-testing-third-edition.

**Berde, P.; Gerola, M.; Hart, J.; Higuchi, Y.; Kobayashi, M. et al.** (2014): ONOS: towards an open, distributed SDN OS. *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, pp. 1-6.

**Cox, J. H.; Clark, R. J.; Owen, H. L.** (2016): Leveraging SDN for ARP security. *SoutheastCon*, pp. 1-8.

**Dargahi, T.; Caponi, A.; Ambrosin, M.; Bianchi, G.; Conti, M.** (2017): A survey on the security of stateful SDN data planes. *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1701-1725.

**De Oliveira, R. L. S.; Schweitzer, C. M.; Shinoda, A. A.; Prete, L. R.** (2014): Using mininet for emulation and prototyping software-defined networks. *IEEE Colombian Conference on Communications and Computing*, pp. 1-6.

**Deng, S.; Gao, X.; Lu, Z.; Gao, X.** (2017): Packet injection attack and its defense in software-defined networks. *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 3, pp. 695-705.

**Fan, Z.; Xiao, Y.; Nayak, A.; Tan, C.** (2019): An improved network security situation assessment approach in software defined networks. *Peer-to-Peer Networking and Applications*, vol. 12, no. 2, pp. 295-309.

**Hameed, S.; Ahmed Khan, H.** (2018): SDN based collaborative scheme for mitigation of DDoS attacks. *Future Internet*, vol. 10, no. 3, pp. 23.

**Huang, S.; Griffioen, J.; Calvert, K. L.** (2014): Network hypervisors: enhancing SDN infrastructure. *Computer Communications*, vol. 46, pp. 87-96.

**Hussain, M.; Shah, N.** (2018): Automatic rule installation in case of policy change in software defined networks. *Telecommunication Systems*, vol. 68, no. 3, pp. 461-477.

**Hussain, M.; Shah, N.; Tahir, A.** (2019): Graph-based policy change detection and implementation in SDN. *Electronics*, vol. 8, no. 10, pp. 1136.

**Kalkan, K.; Gur, G.; Alagoz, F.** (2017): Defense mechanisms against DDoS attacks in SDN environment. *IEEE Communications Magazine*, vol. 55, no. 9, pp. 175-179.

**Khan, S.; Gani, A.; Wahab, A. W. A.; Guizani, M.; Khan, M. K.** (2016): Topology discovery in software defined networks: threats, taxonomy, and state-of-the-art. *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 303-324.

**Ma, H.; Ding, H.; Yang, Y.; Mi, Z.; Yang, J. Y. et al.** (2016): Bayes-based ARP attack detection algorithm for cloud centers. *Tsinghua Science and Technology*, vol. 21, no. 1, pp. 17-28.

**Masoud, M. Z.; Jaradat, Y.; Jannoud, I.** (2015): On preventing ARP poisoning attack utilizing Software Defined Network (SDN) paradigm. *IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies*, pp. 1-5.

**Medved, J.; Varga, R.; Tkacik, A.; Gray, K.** (2014): Opendaylight: towards a model-driven SDN controller architecture. *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pp. 1-6.

**Modi, C.; Patel, D.; Borisaniya, B.; Patel, H.; Patel, A. et al.** (2013): A survey of intrusion detection techniques in cloud. *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 42-57.

**Moyano, R. F.; Cambronero, D. F.; Triana, L. B.** (2017): A user-centric SDN management architecture for NFV-based residential networks. *Computer Standards & Interfaces*, vol. 54, pp. 279-292.

**Nunes, B. A. A.; Mendonca, M.; Nguyen, X. N.; Obraczka, K.; Turletti, T.** (2014): A survey of software-defined networking: past, present, and future of programmable networks. *IEEE Communications Surveys & Tutorials*, vol. 16, no. 3, pp. 1617-1634.

**Rengaraju, P.; Ramanan, V. R.; Lung, C. H.** (2017): Detection and prevention of DoS attacks in Software-Defined Cloud networks. *IEEE Conference on Dependable and Secure Computing*, pp. 217-223.

**Schneider, F.; Bifulco, R.; Matsiuk, A.** (2016): Better ARP handling with InSPired SDN switches. *IEEE International Symposium on Local and Metropolitan Area Networks*, pp. 1-6.

**Scott-Hayward, S.; O'Callaghan, G.; Sezer, S.** (2013): SDN security: a survey. *IEEE SDN For Future Networks and Services*, pp. 1-7.

**Sezer, S.; Scott-Hayward, S.; Chouhan, P. K.; Fraser, B.; Lake, D. et al.** (2013): Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE Communications Magazine*, vol. 51, no. 7, pp. 36-43.

**Shalimov, A.; Zuikov, D.; Zimarina, D.; Pashkov, V.; Smeliansky, R.** (2013): Advanced study of SDN/OpenFlow controllers. *Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia*, pp. 1-6.

**Sharafaldin, I.; Gharib, A.; Lashkari, A. H.; Ghorbani, A. A.** (2018): Towards a reliable intrusion detection benchmark dataset. *Software Networking*, vol. 2018, no. 1, pp. 177-200.

**Shin, S.; Gu, G.** (2013): Attacking software-defined networks: a first feasibility study. *Proceedings of the Second ACM SIGCOMM workshop on Hot Topics in Software Defined Networking*, pp. 165-166.

**Siddiqui, M. S.; Escalona, E.; Trouva, E.; Kourtis, M. A.; Kritharidis, D. et al.** (2016): Policy based virtualised security architecture for SDN/NFV enabled 5G access networks. *IEEE Conference on Network Function Virtualization and Software Defined Networks*, pp. 44-49.

**Wallner, R.; Cannistra, R.** (2013): An SDN approach: quality of service using big switch's floodlight open-source controller. *Proceedings of the Asia-Pacific Advanced Network*, vol. 35, pp. 14-19.

**Wang, L.; Li, Q.; Jiang, Y.; Wu, J.** (2016): Towards mitigating link flooding attack via incremental SDN deployment. *IEEE Symposium on Computers and Communication*, pp. 397-402.

**Xing, W.; Zhao, Y.; Li, T.** (2010): Research on the defense against ARP spoofing attacks based on Winpcap. *Second International Workshop on Education Technology and Computer Science*, pp. 762-765.

**Xu, Y.; Liu, Y.** (2016): DDoS attack detection under SDN context. *IEEE INFOCOM the 35th annual IEEE International Conference on Computer Communications*, pp. 1-9.