Computer Science and Information Systems 15(1):163–185 https://doi.org/10.2298/CSIS170328044M

CSDSM: Cognitive Switch-based DDoS Sensing and Mitigation in SDN-driven CDNi Word

Nishat I Mowla, Inshil Doh, and Kijoon Chae

Department of Computer Science and Engineering, Ewha Womans University,
51, Ewhayeodaegil, Seodaemungu, Seoul, 03760, Korea
nishat.i.mowla@gmail.com, {isdoh1, kjchae}@ewha.ac.kr

Abstract. Content Delivery Networks (CDNs) are increasingly deployed for their efficient content delivery and are often integrated with Software Defined Networks (SDNs) to achieve centrality and programmability of the network. However, these networks are also an attractive target for network attackers whose main goal is to exhaust network resources. One attack approach is to over-flood the OpenFlow switch tables containing routing information. Due to the increasing number of different flooding attacks such as DDoS, it becomes difficult to distinguish these attacks from normal traffic when evaluated with traditional attack detection methods. This paper proposes an architectural method that classifies and defends all possible forms of DDoS attack and legitimate Flash Crowd traffic using a segregated dimension functioning cognitive process based in a controller module. Our results illustrate that the proposed model yields significantly enhanced performance with minimal false positives and false negatives when classified with optimal Support Vector Machine and Logistic Regression algorithms. The traffic classifications initiate deployment of security rules to the OpenFlow switches, preventing new forms of flooding attacks. To the best of our knowledge, this is the first work conducted on SDN-driven CDNi used to detect and defend against all possible DDoS attacks through traffic segregated dimension functioning coupled with cognitive classification.

Keywords: SDN, CDN, CDNi, DDoS, Flash Crowd, Machine Learning, Support Vector Machine, Logistic Regression.

1. Introduction

Content Delivery Networks (CDN) were originally proposed for content providers to meet clients' performance requirements by decreasing web latency during content delivery by distributing content via proximity or edge servers spanning the internet. There are many kinds of Content Delivery Networks, which can be generally classified as either industrial or academic. Popular among industrial CDNs are Akamai, Limelight, and EdgeStream, while CoDeen, Coral, and Globule, are popular academic CDNs. Though all CDNs differ in the way they serve content, they all have an origin server that eventually communicates with some other edge servers. However, the customers of one CDN could not access service from another CDN, which gave rise to various scalability issues [1]. To solve this issue, the interconnection of the CDNs was proposed and referred to as the Internetworking of CDNs, or CDNi [2].

Soon after CDNs became popular, the concept of Software Defined Network (SDN) came into practice, as it promised a better monitoring and controlling scenario for the entire network and could be programmed from a single point controller. Per the Open Network Foundation (ONF), the key to an SDN lies in the physical separation of the control plane from the data plane, while the control plane controls several data plane components. An application plane is also housed on top of the control plane, where the network programming applications are stored as modules. The data plane mainly consists of OpenFlow-enabled switches, which save rules in a flow table by taking instructions from the central controller in the control plane. These planes are connected to the Operations System Support (OSS), which constantly extracts basic management information for the SDN to run [3]. Figure 1 shows an example of an interconnected Content Delivery Network, or CDNi, consisting of two Content Delivery Networks (CDN1 and CDN2) communicating with each other to provide contents to a user.

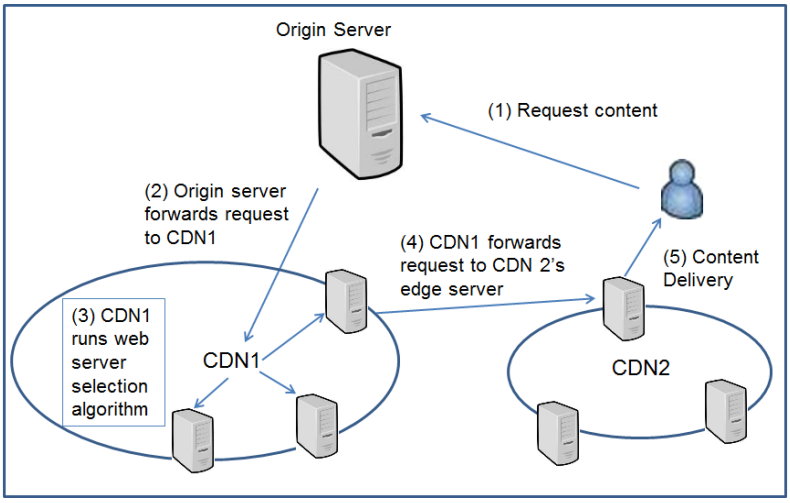

Fig. 1. Interconnect Content Delivery Network (CDNi)

Figure 2 shows the Open Networking Foundation-defined SDN architectural planes and their basic contents. The figure shows the three main planes (Application, Control, and Data plane) which interoperate over northbound and southbound API to manage packet request received at the Data plane switches.

Recently, the combination of SDNs and CDN has received enormous attention due to improved and synchronous CDN request routing by utilizing the centralization and programmability of SDN [4] [5]. However, as these networks become increasingly centralized, they are also becoming potential targets for attacks where a single point of failure can cause damage to multiple components in the centralized network [6]. One of the most widely discussed attacks for these networks is a Distributed Denial of Service (DDoS) attack, which makes the system effectively unavailable [7] [8].

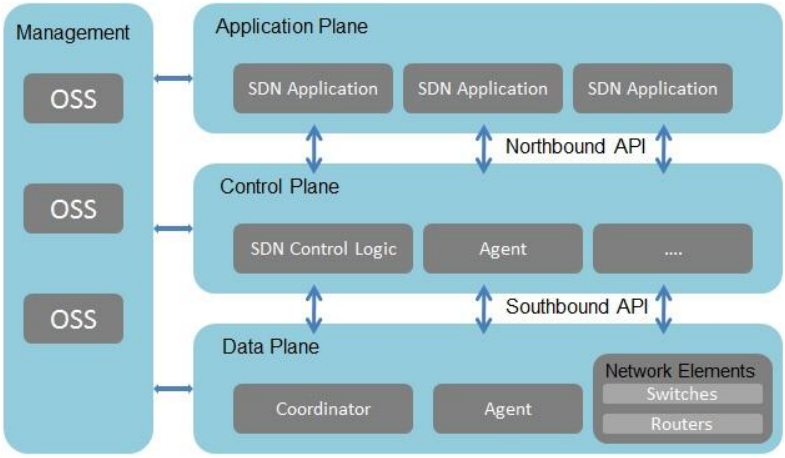

Fig. 2. Software Defined Network (SDN)

Distributed Denial of Service attacks typically occur when a large number of internet packets from compromised hosts (zombies) flood the bandwidth or resources of a single target (victim). The flood of incoming messages to the victim essentially forces it to respond so slowly as to be rendered effectively unavailable and even to shut down, thereby causing denial of service for legitimate users of the targeted system [9]. The number of different types of DDoS attacks is increasing daily, ranging from TCP flooding, UDP flooding, ICMP flooding, SYN flooding, and other source-based and destination-based bandwidth and scanning attacks [10]. Along with the evolution of new next-generation networks, new next-generation attacks are also evolving. These attacks are often hybrid in nature and difficult to identify when we try to match them against previously established signature profiles. In the case of SDN-driven CDNi, the centralized architecture becomes a bottleneck and therefore can be exploited by the attackers who aim to bring down the backbone network. There are three challenges faced by this network: 1) detecting close, 2) detecting soon, and 3) differentiating an attack from a Flash Crowd. Detecting close is critical since attacks can be highly distributed and attack traffic from each source can be made smaller as camouflage. On the other hand, detecting soon is also crucial because alarms need to be activated sooner, especially in big networks, such as SDN/CDNi, on which many devices depend. The last and the most important issue is to be able to identify the attack traffic correctly without false alarms and distinguish it from legitimate Flash Crowds. In this research work, we mainly focus on this last challenge.

By definition, Flash Crowds are large surges of legitimate traffic directed toward some specific sites on the internet over a relatively short period, which can cause a website or target system to slow its service for users or even temporarily close due to the significantly increased traffic. Flash Crowds are quite similar to DDoS attacks in terms of traffic volume [11]. Motivated by these findings, our attack study approach aims to approach attacks differently from the conventional ways of categorizing attack types. Our mechanism considers the occurrence of all possible DDoS attacks and compares them with real network traffic samples. For example, a modern attacker can use tricks to

mimic a Flash Crowd so that its traffic looks legitimate. Therefore, the goal of this paper is to introduce a unique architectural technique to distinguish all possible DDoS traffic from Flash Crowd/Normal traffic in an SDN-driven CDNi using the extraction of OpenFlow switch traffic features, supported by a segregated dimension functioning cognitive approach. For doing so, we propose a stretch model to provide insight into real DDoS and Flash Crowd traffic by utilizing the dimensionalities of traffic features verified against machine learning classification techniques. In essence, the main contributions of this paper are as follows:

- We propose a cognitive detection and defense mechanism to distinguish all possible DDoS attacks and Flash Crowd traffic apart in an SDN based CDNi architecture. Therefore, we provide a robust framework to distribute the complex detection and defense problem into various parts of the SDN based CDNi architecture to detect and defend DDoS attacks more accurately in the presence of legitimate Flash Crowd traffic.
- We formulate a dimension segregation and functioning approach to simplify the processing computation of robust machine learning algorithms further. A systematic stretch model implements the approach to attain higher accuracy rate. Furthermore, the segregation and functioning based stretch model contributes to significantly enhance the evaluation time besides increasing the accuracy of the mechanism.
- We also implement a security module in the SDN Floodlight controller to insert security rules based on the intelligent decision made by segregated dimension functioned Support Vector Machine, and Logistic Regression. We simulate a DDoS attack in an emulated SDN based CDNi architecture and defend the attack traffic by utilizing our security module. In addition, we also evaluate the processing delay caused by our security module and show that the overall effect is minimal for a set of standard values of bandwidth provided to the network.
- We perform extensive experimental analysis to evaluate the performance of the proposed approach. The results show that the dimension segregation and functioning approach achieves higher accuracy for all the four-evaluated state-of-the-art machine learning algorithm. In addition, the accuracy of the two state-of-the-art machine learning algorithms, Support Vector Machine and Logistic Regression outperforms the other state-of-the-art machine learning algorithm for the given scenario.

This paper is, thus, organized as follows. Section II presents some related works in DDoS detection and discrimination from Flash Crowd and classification techniques. Section III introduces our proposed model and architectural mechanism followed by the defense algorithm. The implementation and performance analysis are presented in Section IV. Section V concludes our paper, summarizing the contributions and outcomes.

2. Related Works

There has already been significant research on the detection of DDoS attacks [12]. In [9], DDoS attacks in MANET was considered and proposed a defense using protection nodes forming a tree topology, where the low-level nodes inform the high-level nodes

about a possible attack. However, the paper mainly focuses on attack defense, rather than on establishing a concrete way of detecting an attack and differentiating it from non-attack traffic. In [13], the concept of separating the identifier and locator to detect a DDoS attack was proposed, which distributes the work of detection over the network. The conventional way of detecting attacks has been to create profiles of different attacks and then match them to the sample of concern. However, along with the evolution of new next-generation networks, the number, and type of attack are also evolving daily as attackers tweak their attack patterns in order to bypass these simple network intrusion detection systems.

In [14], the NOX controller was used with OpenFlow for flow analysis using Self-Organizing Maps (SOM) to detect DDoS attacks. The paper focuses on proposing a lightweight mechanism for DDoS detection using the platform provided by NOX. In [15] a clustered neural network was proposed and in [16] a random neural network was proposed for enhancing the separating boundaries of normal and attack. However, neural network techniques can also sometimes classify Flash Crowds as DDoS traffic and vice versa, particularly if the attacker tweaks the traffic features to mimic a Flash Crowd. Besides, in [17] the combination of Bayesian networks and Regression Trees was proposed for optimized feature deduction. In [18], combining classification trees for increasing accuracy in intrusion detection was proposed. A flow-based detection mechanism was proposed in [19], where they tried to differentiate normal traffic with false alarms and DDoS attack traffic using functions for each attack type. Their mechanism consisted of creating a chart of characteristics of the different flooding attacks and matching the sample functions to the flooding attack characteristics. However, the parameters that they used as the standard to calculate the functions could also lead to false alarms in the case of a Flash Crowd.

In [20], [21], [22], and [23] the use of machine learning classifiers was considered to distinguish DDoS from normal traffic, and their experimental results suggest that SVM can be useful for such classifications, detecting fewer false positives and leading to higher accuracy. However, with pure SVM-based classification, an attacker tweaking traffic features can still be successfully misclassified as a Flash Crowd. In, [24] the difference of using machine learning in network intrusion detection was discussed and in other domains and they pointed out the need of illuminating the problem space and binding careful decisions with the problem. In [6] and [25], some of the attack defense capabilities of SDN architecture were discussed as a futuristic next-generation network. However, while they talked about the many security solutions that SDN architecture offers, they did not focus on specifically what sort of attacks the SDN architecture itself can face and therefore the sort of detection measures required for those attacks.

In, [26], [10], [27], and [11] specific discriminating techniques were proposed to differentiate attack traffic from Flash Crowds. Most of these techniques rely on creating packet arrival patterns or measuring the flow distances (based on flow distance calculation methods such as Sibson's distance) to distinguish attack traffic. As discussed earlier, Flash Crowds can also lead to traffic flows with similar distances, and attackers can tweak attack profile information to bypass intrusion detection. In [11], the idea of a correlation coefficient between flows was proposed. This technique considers all packets incoming to a certain destination as one flow, and while it differentiates traffic quite well, it does not help to categorize traffic based on the incoming traffic's unique header combination. Because our approach uses OpenFlow-enabled traffic extraction, we can

uniquely identify and categorize the packets with the same 5-tuple headers as one flow and thus can better categorize flows for our attack analysis.

As can be seen, some of the detection mechanisms do not consider the similarities of a DDoS attack and a Flash Crowd, while some do not realize a universally suitable method to detect them. Although machine learning techniques are known to effectively classify with very low false positive rates, most of the proposed mechanisms that feed the traffic into the classifier directly and test the performance of different machine learning classifiers can still lead to false alarms when we feed in traffic that mimics Flash Crowd traffic. To the best of our knowledge, the art of feeding more useful input in order to output more efficient classification has not yet been considered. The goal of this paper is to address these aspects while proposing a unique approach for DDoS discrimination from normal or Flash Crowd traffic for next-generation networks.

3. Proposed Detection and Defense Mechanism

Our proposed detection and defense mechanism has two parts. The first part includes the proposed architecture in which the mechanism will be implemented, and the second part elaborates on the detection mechanism itself.

3.1. Proposed Architecture

Our proposed SDN-based CDNi architecture consists of interconnected Content Delivery Network servers connected to OpenFlow-enabled switches, which are in turn connected to a central controller for SDN services. In this paper, we are considering the DDoS attack aimed at the OpenFlow switch of the SDN which is also responsible for implementing the defense mechanism to protect the rest of the network comprising of CDNs. Therefore, if the OpenFlow switch is protected and defending the DDoS attack properly, the CDN servers and the rest of the network will also eventually be protected. Hence, due to the limitation of our emulator and to the fact that, for our paper, we are considering the DDoS attack aimed at the OpenFlow switch, of the SDN and not at the CDNs which will eventually be protected by the OpenFlow switch, we do not define the type of CDN. Instead, we simply consider the CDN until the origin servers. The application modules, housed on top of the SDN controller, feed rules into the OpenFlow-enabled switches using the southbound OpenFlow API. When a client makes a request, it is forwarded to the best-suited CDN/CDNi by the OpenFlow switch in the SDN network, where a suitable edge server provides services to the requesting client. Figure 3 shows our proposed architecture with its three main sectors.

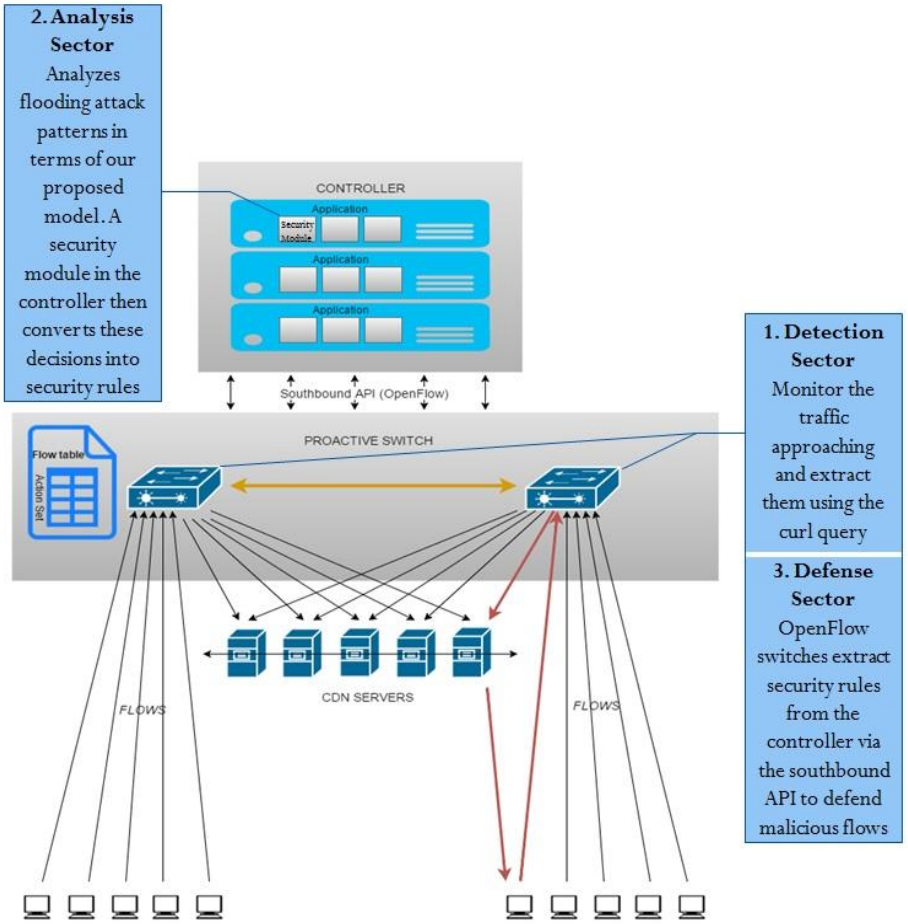

Fig. 3. Proposed SDN-driven CDNi Architecture for DDoS sensing and mitigation

Typically, in a Software Defined Network, all requests should pass through an OpenFlow switch, which determines its route based on the rules saved in the switch's table. Thus, the switch plays an important role and is therefore quite vulnerable to certain attack types. In our proposed scenario, when a customer wants to access services from a CDN server, its request is forwarded first to the SDN controller by a proactive OpenFlow switch. The controller then analyzes the request, before being forwarded by the OpenFlow Switches in the data plane. The size of the switch table is thus controlled by a proactive OpenFlow switch which inserts a rule only after being approved by the SDN controller and not instantly inserting a rule as it happens in a reactive OpenFlow switch scenario. In the process, the requests remain in a default queue of the SDN architecture and, therefore, does not immediately overflood the OpenFlow switch until the controller confirms a rule to be inserted for a certain set of flows. Besides, any incoming traffic is dropped based on the type of flow and not the type of packets which allows to categorizing and concise the incoming traffic further. Here, a flow is

considered to be any traffic with the same five tuples consisting of source IP, destination IP, source port, destination port and protocol. The detection and defense mechanism consist of three sectors as also shown in Figure 3: the detection sector, the analysis sector, and the defense sector.

Detection Sector. The detection sector in our proposed mechanism is in the OpenFlow Switches, where approaching traffic is identified and extracted using curl query API for the OpenFlow Switches. Through curl queries, we can extract specific feature information about the ingress flows and packets entering the OpenFlow Switch. The detection sector thus provides the raw data for analysis from which the proposed detection technique can operate.

Analysis Sector. The analysis sector is the controller, where a security module converts the analyzed decisions into security rules to defend against the attack and protect the attacked switch. The proposed mechanism leverages machine learning classification techniques for analysis to find traces of an attack in the ingress flow information extracted at the detection sector.

Defense Sector. In the defense sector, the results from the detection mechanism in the controller side are leveraged to enact security rules in the OpenFlow Switch via the southbound API. The classification from the machine learning technique of the analysis sector is used to set rules of the following form:

Rule *R*: If $x_i \in C_j$
then action = *drop* with *IP==attackFlowIP*

where *R* is the rule for the *i*-th *n*-dimensional pattern vector, $x_i = (x_{i1}, x_{i2}..., x_{in})$ which belongs to *j*-th class, C*j*. The 'then' part defines the action that causes the OpenFlow switch to drop the flow with the IP address of x_i. This security rule is translated into the SDN OpenFlow rule format and applied to the OpenFlow switch, which then uses this rule to drop malicious flows and thus act as the defense sector.

However, it is important to note that Flash Crowds can also cause flooding of flows like attack traffic, and attackers may try to mimic Flash Crowds. However, since Flash Crowds are legitimate packets, they require normal service. This necessitates an effective analysis mechanism to distinguish Flash Crowd traffic from DDoS traffic. In the next section, we propose a model based on a dimension segregation and functioning technique that aims to efficiently differentiate Flash Crowd traffic from DDoS traffic to aid in machine learning classification. Here we discuss a comparison of our proposed detection and defense mechanism with other previously proposed approaches as shown in Table 1.

Table 1. Comparison Chart

	Previous Approaches	Proposed Approach
Proposal	Typical detection and defense techniques [9,12,18]	Proposed an intelligent SDN controller module
Architecture	Traditional architectures [9,12]	SDN based CDNi
Feature deduction	Direct feature use, feature selection [19,20,21,22,23]	Feature segregation, feature functioning
Technique and goal	Correlation coefficient [10, 11], Sibson's distance [27], Bhattacharyya coefficient [26], Machine learning [20,21,22.23] etc. to detect DDoS attacks in general	Machine learning with segregated feature functioning to detect all possible forms of DDoS attacks including those trying to mimic Flash Crowd
Hardware requirement	Sometimes extra hardware required [9, 10,11, 13,26,27]	No extra hardware required
Tool used	For machine learning, commonly, Python (Scikit) requiring python environment and needs to check dependency Matlab considered to be heavy weight [28]	Weka tool is easy to implement with a better interface and can easily be compiled into native code with cross-platform tool [29]

3.2. Dimension Segregation and Functioning Model

Network traffic information can be represented as flows and packets or can be divided into more dimensions as meaningful information is derived when we segregate basic features. Flows can be represented in two dimensions by flow count and flow size, and packets can be represented in two dimensions by packet count and packet rate. The raw data can be used directly or can be represented with functions that extract more characteristics from them. Different kinds of flooding attacks give rise to different kinds of peaks. For example, a TCP SYN flood will have a large flow count, whereas a ping of death attack might have a smaller flow count but larger flow size. However, these properties can also be tweaked by an attacker to change the expected profile for a certain attack. What remains true is that, when the attacker has an immense volume of attack traffic to manage, the attacker is not able to assign unique sets of properties to each traffic instance, creating patterns in traffic properties. We can visualize these patterns when we segregate traffic into useful dimensions. In this model, we segregate based on four features: packet count, flow size, packet rate, and flow count. The patterns of these features are most prominent when we look at their standard deviation function. This is because, it is not easy for the attacker to create high deviations for each packet while generating a huge number of attack packets; therefore, standard deviations of DDoS traffic features are smaller than those of Flash Crowd traffic [5]. Classification based on

this dimension segregation and functioning approach can then be verified against optimized machine learning classification techniques to visualize improvements. Figure 4 shows our dimension segregation and functioning for traffic instance vectors in a four-dimensional space.

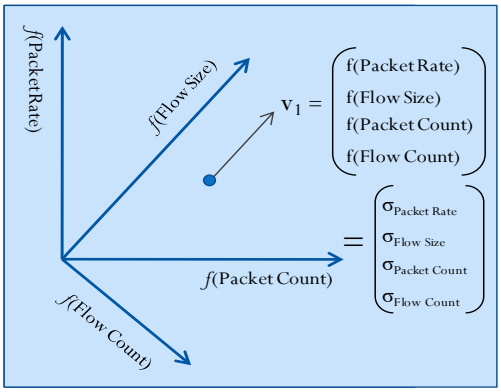

Fig. 4. Proposed Segregation and Functioning Process. A vector v represents a traffic instance comprised of four traffic features or dimensions represented as a function of themselves

It should be noted that our classification is based on flows. Therefore, if the packet flow is continuous and simultaneous, we can separate them as flows. This also allows to categorize and manage the traffic better. Thus, our mechanism can distinguish if the incoming flows from one switch are a mix of DDoS and Flash Crowd traffic. When we consider a Flash Crowd flow, the individual standard deviation of all four features, packet count (*pc*), packet rate (*pr*), flow size (*fs*), and flow count (*fc*), of that specific flow during times t_1 to t_2 and t_2 to t_3 will tend to be high. On the other hand, if we consider a DDoS flow, the individual standard deviations should be low for the same time frames. A challenge could be created by DDoS attacks that try to vary the values of the features and try to make them random. However, when there is a huge volume of traffic to manage and send, it is difficult for an attacker to assign individualistic values to all instances of each flow the attacker sends. Therefore, when we evaluate the standard deviations of the features in the classification test, the division between the Flash Crowd and DDoS flows will become more distinct and easily separable. The easy separability of this technique can also support in reducing the processing time of the machine learning algorithms as less computation will be required to form the separating margin

3.2.1 Stretch Model

We propose a stretch model based on dimension segregation and functioning as described above. In the stretch model, the provided data is divided into vector

dimensions, analyzed, and then divided further until an optimal point is reached. We test our stretch model using two machine learning techniques, the Support Vector Machine, and Logistic Regression, which are each highly optimized for a two-class problem. Each time the number of dimensions increase, we test the performance of our data input against these two machine learning classification techniques. At a suitable model dimension, we apply a function of the standard deviation and apply the Machine Learning Algorithms, as described in the next subsection. Figure 5 shows the workflow of our stretch model.

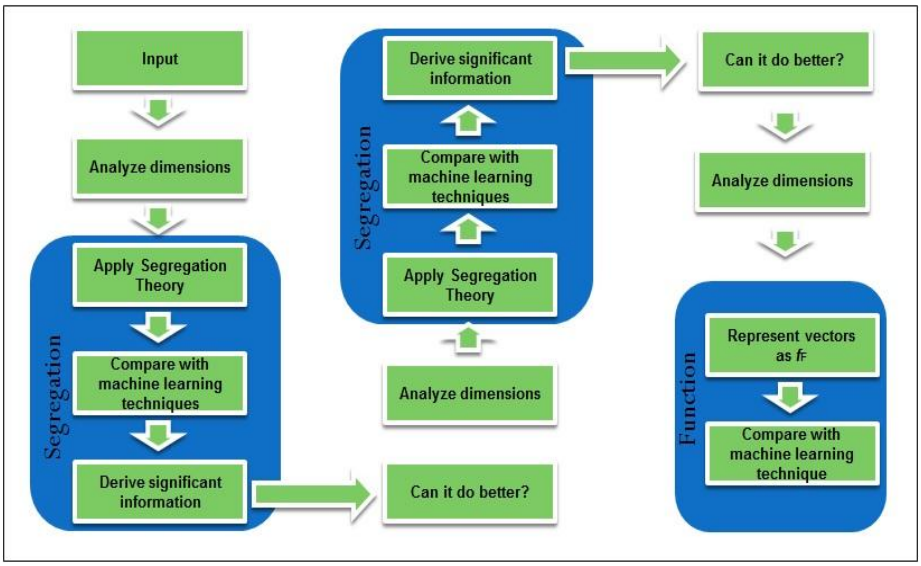

Fig. 5. The Stretch model created by using dimension segregation and functioning subjected to machine learning classification

Algorithm 1 Flooding Attack Detection and Defense Algorithm

for i=1 *to* n
  f_{Fi} is standard deviation function of ith feature F
run classification test;
if *sample! = DDoS class* **then**
  forward flash crowd traffic to free OFSwitch;
  forward normal packet to CDN;
else
  active security rules;

Fig. 6. Proposed DDoS Flooding Attack Detection and Defense Algorithm

Figure 6 shows our detection and defense algorithm inspired by the above mechanism. If the instance is found to be DDoS class, then the security rules are implemented. If the instance is not a DDoS class, then a further classification is run on the Flash Crowd class, since Flash Crowd traffic is more prone to false alarms due to its randomness. This also allows us to double-check the traffic that we tentatively accept as normal. The complexity of the algorithm itself is linear assuming 1 unit of time cost and a single processor of the SDN controller conducting sequential execution over a single loop. Further evaluation of the delay caused by our algorithm will be discussed in section IV. In the next section, we discuss our mathematical model that is used to validify our segregated dimension functioning approach.

3.2.2 Mathematical Model

Dimension segregation is used to increase dimensions and to test the vector's standard deviation function of feature *i*, f_i, against machine learning classification techniques. For this, we use two optimal classification techniques, namely the SVM and Logistic Regression. Machine learning classification techniques are optimal for discrimination processes in almost all domains. One of the most efficient machine learning technique is the Support Vector Machine (SVM). Support Vector Machines work by creating a classification plane in order to separate vectors from two classes onto either side of this classification plane. If it is a two-dimensional space, the separating plane is called a linear classification line. However, if it is in three dimensions, the separating plane is simply called a plane. Apart from that, there are multiple dimensions, the separating plane is called a hyperplane [30]. Besides Support Vector Machine, Logistic Regression is also quite optimal with the two-class problem in a multi-dimensional environment. In Logistic Regression, when there is a two-class problem, the probability of a vector belonging to a certain class can be quite useful and more important than the value of the class itself. Therefore, Logistic Regression aims to find the probabilities of vectors belonging to a certain class [31]. Due to these unique features of Support Vector Machine and Logistic Regression, we use these two algorithms as our baseline algorithm, where we apply a proposed feature dimension functioning.

In segregation model, we consider every instance vector, x, described by particular dimensions. Instead of using the raw dimensional value, we calculate the standard deviation value for the dimensions at specific intervals, denoted as a function of the feature *i* defined as

$$f_i = \sigma \text{ of feature } i. \quad (1)$$

where feature i can be one of our selected features: packet count, packet rate, flow size, or flow count. Accordingly, in terms of the SVM, we can consider $x = f_i$ in the function $G(x)$ as

$$G(f_i) = \omega^T f_i + b = 0. \quad (2)$$

where ω^T is the orientation of the separating hyperplane, and b is the position. The values of ω^T and b come out of the training and optimization process, which maximizes

the separation of the support vectors from the classifying hyperplane, G(f_i). The default optimization mechanism for SVM is used to extract the optimal values for w and b for the hyperplane, G(f_i), where we then input the vectors in the form of our vector function f_i to produce the optimal classification.

In the case of a dichotomous output, which in our case is a DDoS or a Flash Crowd, it is also useful to classify based on the probability of a sample belonging to either of these two classes. Here we used Logistic Regression to calculate the probability p by equating our function-based features in the probability equation, denoted as

$$p = e^{b_0 + b_1 f_1 + b_2 f_2 \ldots b_n f_n} / (1 + e^{b_0 + b_1 f_1 + b_2 f_2 \ldots b_n f_n}). \quad (3)$$

where b_i is the associated coefficient for feature i, denoted as a feature function f_i. Using this probability, we derive the Logistic Regression equation as

$$\text{Logit}(p) = b_0 + b_1 f_1 + b_2 f_2 \ldots b_n f_n \quad (4)$$

which presents the probability of a vector belonging to a particular class. In the next section, we discuss the performance analysis of our mechanism based on the above described methodologies.

4. Experimental Result and Performance Analysis

4.1. DDoS Detection Mechanisms

For implementing our proposed mechanism, we verified our model with real datasets, namely the World Cup 98 dataset [32] that caused one of the biggest Flash Crowd events in history, and the CAIDA DDoS 2007 dataset [33] for real DDoS attack traffic. Both the datasets are standardized and publicly available State-of-the-art datasets. Besides, the World Cup 98 dataset for Flash Crowd is one of the few comprehensive datasets available for Flash Crowd evaluation. Like other networks, the SDN based CDNi is also used as a basic network which can be prone to DDoS attacks in a similar manner. Therefore, we believe the two selected datasets to be well suited for this network evaluation. We also used attack tools to generate DDoS attack traffic in the Mininet emulator, which was later defended by the security rules inserted into the OpenFlow Switch.

We used two main tools for our experimentation. The Mininet 2.2.0 emulator [34] was used to build the architecture of the SDN-driven CDNi, where security rules were also implemented. Weka 3.8 was used to analyze the performance of our proposed classification techniques against machine learning algorithms. Results of the performance analysis were then plotted for graph generation.

4.2. System Architecture

The proposed architecture, comprised of the three sectors of detection, analysis, and defense, was implemented via the SDN-driven CDNi architecture emulated in Mininet. Ten hosts were connected to the OpenFlow Switches, which were in turn connected to four CDN origin servers. A Floodlight controller was implemented on a remote server machine, which was connected to the OpenFlow switches. When a request approaches the OpenFlow Switch to access a CDN server, the rules saved in the OpenFlow switch are used to decide whether to redirect the request to the CDN server or to drop it. The rules come from the controller, which is responsible for delivering security rules to the OpenFlow switch via a security module housed in the application plane of the controller. Figure 7 shows the simulated architecture in Mininet.

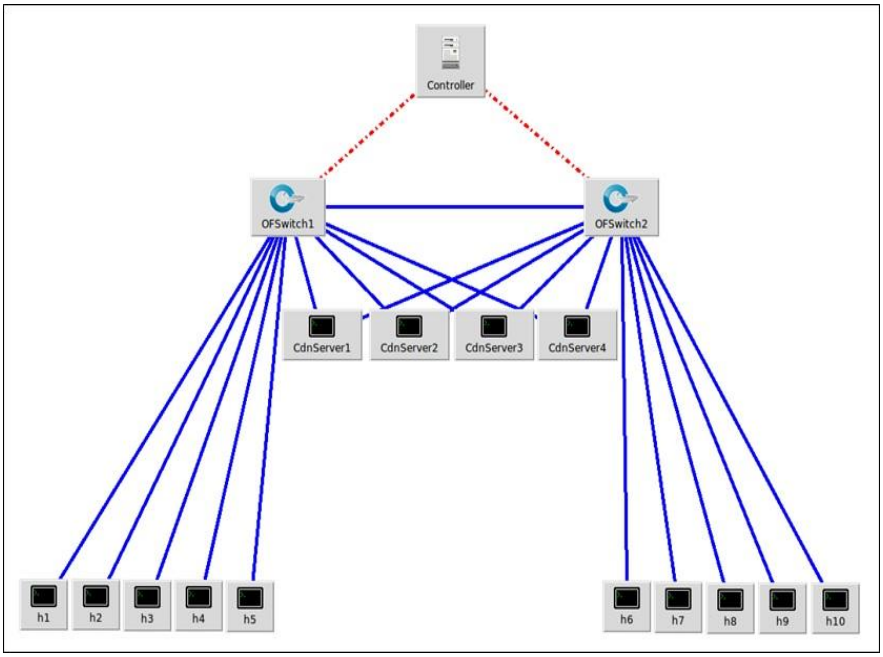

Fig. 7. Mininet Simulation Environment

Our proposed mechanism requires monitoring the traffic in terms of packet count, packet rate, flow size, and flow count, each extracted through curl queries used for the classification test. The classification test in Weka was performed using the World Cup 98 Flash Crowd and CAIDA DDoS datasets. The datasets are used to validate our proposed mechanism via experimentation, following which traffic collected from the OpenFlow Switch can be similarly treated to defend flooding attacks.

The simulation has two major parts. The first part includes conducting the classification tests while managing dimensions and applying the proposed function. The second part includes emulating an attack environment in Mininet and implementing security rules to defend against the malicious flows.

Our experimental setup contains a lab environment where we set up a controller in one Linux machine (Ubuntu 14.04, 2.94 GHz, and 4 GB RAM). We set up the emulated SDN-based CDN servers in another Linux machine also running Ubuntu 14.04 in Virtual Box with hosts operating with the controller over the Mininet 2.2.0 emulator. The joining point is the OpenFlow Switch, which collects information about the flows, which are then segregated into useful dimensions by converting them to their standard deviation functions. These vectors are then fed into the Weka tool as ARFF (Attribute-Relation File Format) files, where classification tests are run using the machine learning techniques. The results of the classification test are discussed in the next section.

4.3. Implementation and Performance Evaluation

Machine Learning Classification Performance. We used a total of 38,484,863 Flash Crowd flows and 22,569,183 DDoS flows for the classification test. Due to the dataset format and processing limitations, we rearranged the data into 14 different sets. To verify our results, we tested the results using cross-validation in 10-fold, the full training set as well as percentage split tests of 50:50, 60:40, 70:30, 80:20, and 90:10 training/test datasets, for both the SVM and Logistic Regression classifiers. The traffic data was first divided into two dimensions, flows and packets. The two dimensions' classification was then tested against SVM and Logistic Regression classifiers. In the next experiment, we increased the number of dimensions to four where we used the four dimensions of packet count, packet rate, flow size, and flow count. The same seven tests in both SVM and Logistic Regression were repeated for the 4-dimension case. Next, we processed our dimensions with the standard deviation functions and re-ran the classification tests. Figure 8 shows the classification performance for the three experiments.

With four dimensions, more packets were correctly identified than the two-dimensional case, and almost all tests in both SVM and Logistic Regression resulted in higher values. Logistic Regression achieved 100% correct classification in the training set test and 90% in the percentage split test. The performance in all experiments improved when we applied the proposed function, with nearly every test resulting in 100% correct classification. Logistic Regression achieved 100% correct classification in six tests and over 90% correct classification in the seventh test performed. SVM achieved 100% correct classification in five tests and over 90% in the remaining two tests, still much higher compared to the results for the four-dimensional case. To verify the performance further, we ran the classification test for two other state-of-the-art classifiers in a four-dimensional case and a functioned four-dimensional case. Figure 9 shows the average classification performance of all four classification techniques with the proposed function. As can be seen, all the four classifiers' performance is higher with functioned four-dimensional case than that in four-dimensional case. This is because the functioning of the features allows making the features of DDoS and Flash Crowd more differentiable. The functioning is based on standard deviation value which will eventually be low for DDoS when we combine all the DDoS data features while it will be sparser for Flash Crowd. The functioned four-dimensional case is further elaborated for all the classifiers through the same seven tests as shown in Figure 10.

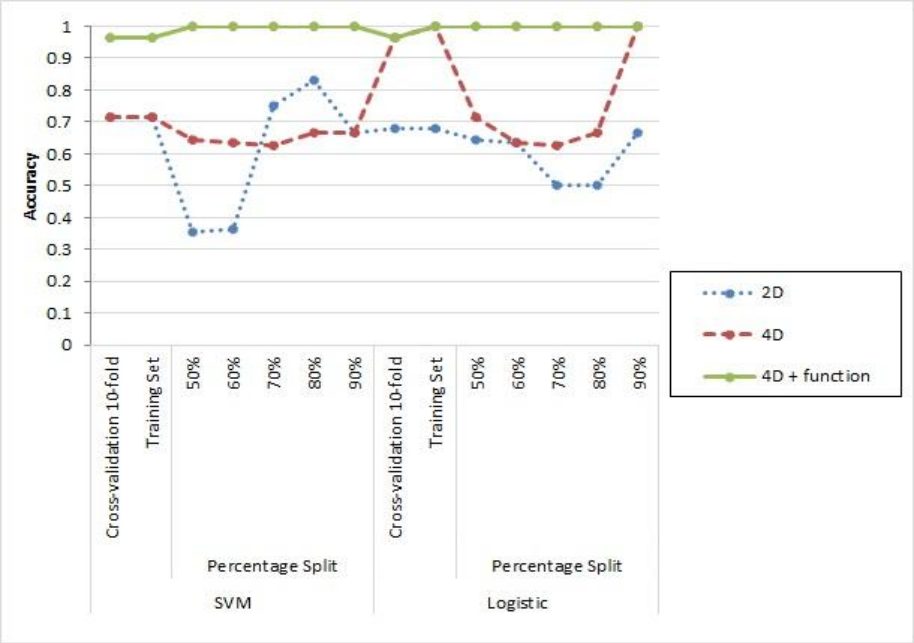

Fig. 8. Classification performance with dimensional segregation and dimensional segregated functioning

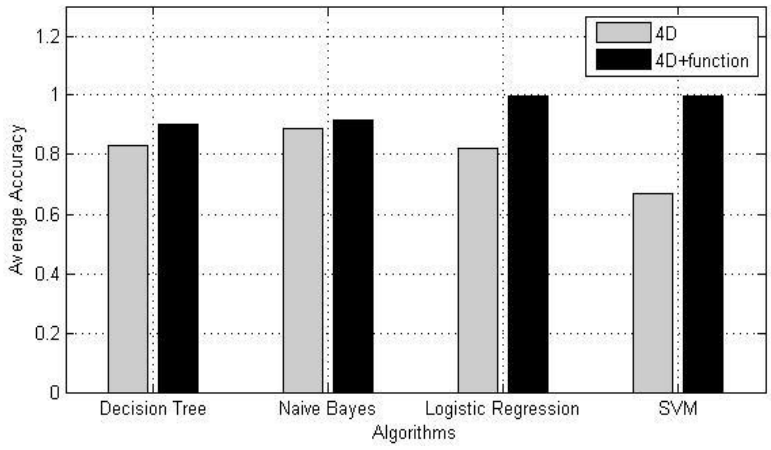

Fig. 9. Average performance of four classification technique with and without dimension functioning

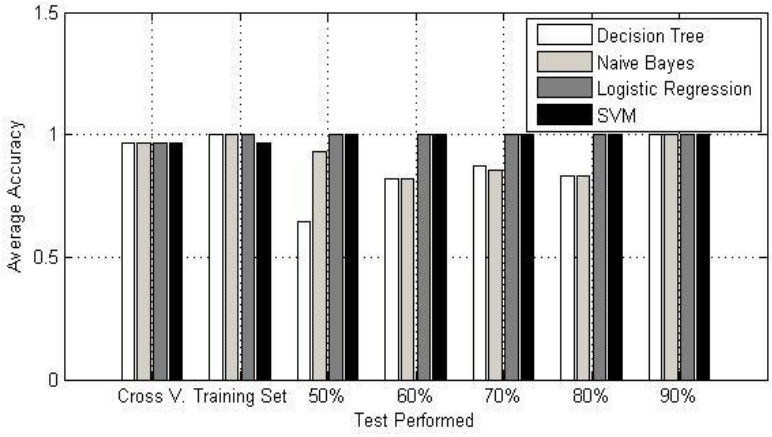

Fig. 10. Classification performance of four classification techniques with proposed function

Though the average performance is higher with the functioned four-dimensional case for all classifiers, Logistic Regression and SVM significantly benefit from the functioning in all seven tests. This is because both SVM and Logistic Regression are robust in detecting noisy and sparse data that is introduced here by the Flash Crowd data instances. SVM and Logistic Regression perform comparably in practice. However, the performance of SVM was little lower than Logistic Regression in the pure training set test. This phenomenon is expected, as SVM is trying to simplify a problem that is already simplified by the feature functioning process while Logistic Regression utilized the feature functioning and solves it better with a more probabilistic approach. The performance of our proposed segregation function for SVM is further discussed with precision and recall in the next subsection.

Precision and Recall. In practice, there is always a possibility of false positive or false negative results. In our set of tests, the possibility of false positive is more likely for a Flash Crowd, as it can accept any randomness that is introduced. However, our mechanism should be able to correctly classify all DDoS traffic, as the standard deviation will have a similarity that is difficult for attackers to avoid. The difficulty lies in giving unique features to each instance of thousands and thousands of requests. The above-discussed scenario is also illustrated through our TPR, FPR precision and recall tests subjected to SVM shown in Table 2 and 3.

Table 2. True Positive Rate and False Positive Rate

	DDoS	Flash Crowd
True Positive Rate	0.929	1
False Positive Rate	0	0.071

Table 3. Precision and Recall

	DDoS	Flash Crowd
Precision	1	0.933
Recall	0.929	1

As per the precision and recall tests, all Flash Crowd traffic is predicted correctly, which also results in a True Positive Rate (TPR) of 1; however, some DDoS is identified as a Flash Crowd, which results in a False Positive Rate (FPR) of 0.071. However, no Flash Crowd traffic is identified as DDoS, for which the False Positive Rate (FPR) is 0, but not all DDoS is predicted correctly, resulting in a True Positive Rate (TPR) of 0.929. This means that, in cases of DDoS, there were some False Negatives, causing the False Negative Rate (FNR) for it to be 1-TPR = 0.071. In this regard, our approach improves the State-of-the-art architectures in a significant way as shown by the above results. Also, we consider the issue of detecting all Flash Crowd correctly could be dealt with further learning by a second level of security. Hence, we consider this issue as a future work of our mechanism.

In terms of nodal delay, which is a summation of transmission delay, propagation delay, processing delay and queuing delay, our architecture is subject to queuing delay and processing delay [35]. This is because the transmission delay is dependent on the channel capacity while the propagation delay depends on the environment. Therefore, considering the above discussed two delays remain the standard values, we evaluate the processing delay and queuing delay of our proposed algorithm. Figure 11 shows the average processing delay for the machine learning algorithm with and without dimension functioning.

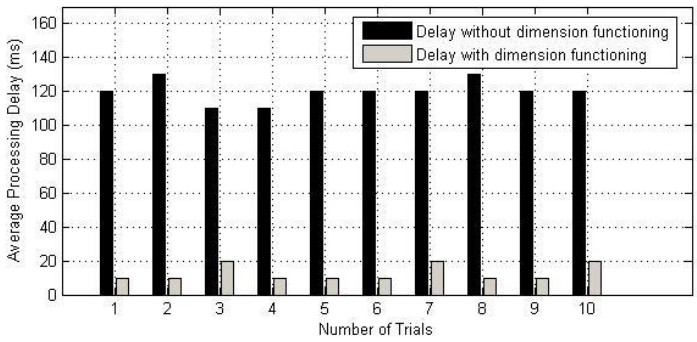

Fig. 11. Average processing delay with and without proposed dimension functioning

Our segregation and dimension functioning method significantly reduces the processing delay of the machine learning algorithm since the segregated dimension functioning causes easy separability of the two classes. The delay with no dimension functioning, however, remains very high as it takes a lot of computation for the machine learning algorithm to create a separating plane for raw DDoS and Flash Crowd instances. In case of the machine learning algorithm with dimension functioning, the functioned features easily separate for DDoS and Flash Crowd with lower values for

DDoS and higher values for Flash Crowd. This aids in a quicker generation of a separation plane for the machine learning algorithm resulting in much lower processing delay.

Security Rule Implementation. Once the decision-making process is complete, the next step is to feed the decision from the controller application plane security module into the defense sector, which is in the OpenFlow switch. To do so, we write our own API in the application plane module that can be used to insert security rules based on our command into the OFSwitch and drop malicious flows. For our security rule insertion, we manipulated a Floodlight built-in module that helps to drop packets. Here, we wrote our own API to drop flows in order to proactively control the OpenFlow Switches from being overflooded with rules.

For experimentation, we generated an ICMP flood-based DDoS attack in a Mininet emulator against the OpenFlow Switch associated with the controller. We tested our experiment over 10 hosts, where 6 hosts were creating a DDoS attack on the OpenFlow Switch. We monitored the other 4 hosts and their service as the OpenFlow Switch experienced a DDoS attack.

DDoS traffic was generated from an IP of 10.0.0.2 toward another IP of 10.0.0.5 (one of our CDN origin servers), causing an ICMP flood and was viewed from OpenFlow switch S1. Another 5 hosts were also used to make the flooding attack stronger. The simulated DDoS attack caused a 16% packet loss from one of the legitimate hosts. Similar packet losses were observed in another 4 hosts. After the flooding attack, we implemented a curl POST to insert the rule with the following commands:

```
curl -X POST -d {switched: 00:00:00:00:00:00:00:01}
http://localhost:8080/wm/firewall/rules/json

curl -X POST -d {{"src-ip": <attack host ip>, "flood":
"takeaction"}
http://localhost:8080/wm/firewall/rules/json
```

in which the first command selects the switch facing the attack, and the second command causes the flows of the host creating the flooding attack to be dropped.

After we installed the security rules with our API definition, we ensured that the incoming flooding flows were dropped by our security rule while ongoing flooding flows were down-linked. After restoring the table utilization of the OpenFlow switches, we found that the 4 hosts in the network experienced 100% packet reception, and normal communication was restored. Figure 12 illustrates the cumulative distribution of the average queuing delay caused by our security rule implementation.

From Figure 12, we observe that the average queuing delay increases slightly more in the proposed security rule based architecture than the architecture without any security rule. The reason behind this slight increase in queuing delay is due to the added overhead caused by the security rule implementation. However, the average delay distribution with security rule module running and not running concentrates between 0.4 to 0.8 ms. Since the processing delay, on the other side, is significantly reduced by our proposed approach, the slightly increased amount of queuing delay does not significantly affect the overall tolerable delay. Additionally, the proposed security rule

based system enhances the security of the architecture for increased performance gain at the cost of a minor increase in queuing delay.

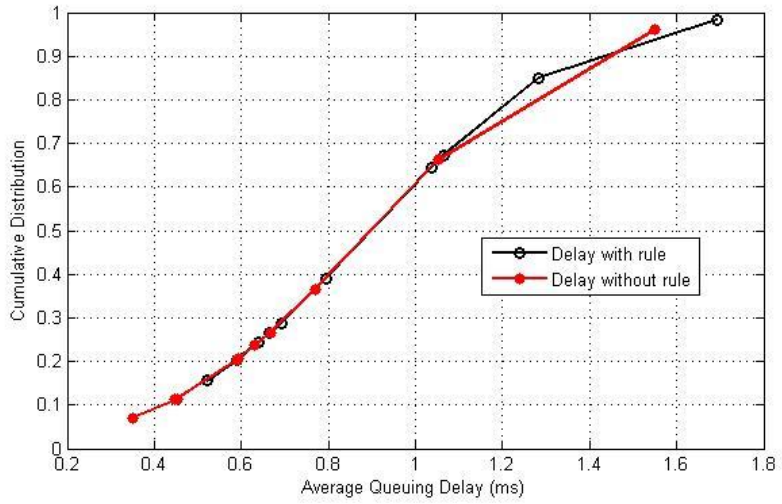

Fig. 12. Average queuing delay comparison between delay with security rule and without security rule

5. Conclusion

In this paper, we presented a detection and defense architecture of an SDN-based CDNi network environment where we utilized network traffic features to detect attack traffic, such as DDoS, and efficiently differentiated it from other high-volume normal network traffic, such as a Flash Crowd. For doing so, we proposed a theoretically, mathematically, and experimentally-supported cognitive classification mechanism based on the concept of dimensional segregation and functioning. The proposed classification mechanism can efficiently insert rules in the SDN OpenFlow-enabled switches to prevent the OpenFlow switch from being over-flooded. We also introduced a deep-inspection mechanism for DDoS detection in a SDN-driven CDNi network environment along with a stretch model to enhance performance. Furthermore, Flash Crowd traffic was used as the normal class instances to rigorously validate the DDoS classification. The experimental results showed the high-performance gain of our proposed mechanism with two optimized machine learning classification techniques, SVM and Logistic Regression. We also contrasted the results with two other state-of-the-art classifiers, Decision Tree, and Naïve Bayes. Our results suggest that other traditional classifiers can also benefit from segregated dimensional functioning. However, SVM and Logistic Regression significantly enhance the overall classification compared to other classifiers and benefit the most from our proposed model. We believe that our mechanism can be a useful technique for malicious traffic detection and defense for next-generation networks

such as SDN-based CDNi, which also tend to experience normal Flash Crowd traffic. To the best of our knowledge, this is the first work using SDN-driven CDNi to sense and mitigate DDoS attacks by leveraging the concepts of segregated dimension functioning to achieve high classification performance.

Our mechanism, however, still does not take the decisions from the machine learning techniques and dynamically insert security rules, which we consider being a possible future work. In the future, we will also consider other dimensional utilization processes that can be effectively used to increase the detection of next-generation attacks and flash crowds.

Acknowledgements. The work was supported by the National Research Foundation of Korea (NRF) funded by the Korea government (MSIP) (2016R1A2B4015899). Kijoon Chae is corresponding author.

References

1. Buyya R., Pathan M., and Vakali A.: Content Delivery Networks," Vol. 9. (2008)
2. Niven-Jenkins B., Le Faucheur F., and Bitar N..: Content Distribution Network Interconnection (CDNI) Problem Statement. Internet Engineering Task Force (IETF), Request for Comments: 6707. (2012)
3. Open Networking Foundation: SDN Architecture," Open Networking Foundation Issue 1. (2014)
4. Mowla N., Doh I., and Chae K.: An Efficient Defense Mechanism against Spoofed IP attack in SDN based CDNi. The 29th International Conference on Information Networking (ICOIN), 92-97. (2015)
5. Shin M-K., Lee S., Chang D., and Kwon T.: CDNI Request Routing with SDN," Network Working Group, 1-9. (2013)
6. Di Maio, A., Palattella, M. R., Soua, R., Lamorte, L., Vilajosana, X., Alonso-Zarate, J., & Engel, T.: Enabling SDN in VANETs: What is the Impact on Security?". Sensors, Vo. 16, Issue 12, pp. 2077. (2016)
7. Luo, S., Dong, M., Ota, K., Wu, J., & Li, J.: A Security Assessment Mechanism for Software-Defined Networking-Based Mobile Networks". Sensors, Vol. 15, No. 12, 31843-31858. (2015)
8. Yan Q. and Yu F.: Distributed denial of service attacks in software-defined networking with cloud computing," IEEE Communications Magazine, Vol. 53, Issue. 4, 52-59. (2015)
9. Xiang M., Chen Y., Ku W., and Su Z.: Mitigating DDoS Attacks Using Protection Nodes in Mobile Ad Hoc Networks. IEEE Global Telecommunications Conference (GLOBECOM), 1-6. (2011)
10. Thapngam T., Liu S., Zhou W., and Beliakov G.: Discriminating DDoS Attack Traffic from Flash Crowd through Packet Arrival Patterns. IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 952-957. (2011)
11. Yu S., Zhou W., Jia W., Guo S., Xiang Y., and Tang F. Discriminating DDoS Attacks from Flash Crowds Using Flow Correlation Coefficient. IEEE transactions on Parallel and Distributed Systems, Vol. 23, No. 6, 1073-1080. (2012)
12. Zargar S. T., Joshi J., and Tipper D.: A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks," IEEE communications surveys & tutorials, vol. 15, no. 4, 2046-2069. (2013)
13. Luo H., Lin Y., Zhang H., and Zukerman M.: Preventing DDoS Attacks by Identifier/Locator Separation. IEEE network, 60-65. (2013)

14. Braga R., Mota E., and Passito A.: Lightweight DDoS flooding attack detection using NOX/OpenFlow. IEEE 35th Conference on Local Computer Networks (LCN 2010), 408-415. (2010)
15. Wang, G., Hao, J., Ma, J. and Huang, L.: A new approach to intrusion detection using Artificial Neural Networks and fuzzy clustering. Journal of Expert Systems with Applications, Vol. 37, No. 9, 6225-6232. (2010)
16. Saeed, A., Ahmadinia, A., Javed, A. and Larijani, H.: Intelligent intrusion detection in low-power IoTs. ACM Transactions on Internet Technology (TOIT), Vol. 16, No. 4, 27. (2016)
17. Chebrolu, S., Abraham, A. and Thomas, J.P.: Feature deduction and ensemble design of intrusion detection systems. Journal of Computers & security. Vol. 24, No. 4, 295-307. (2005)
18. Kevric, J., Jukic, S. and Subasi, A.: An effective combining classifier approach using tree algorithms for network intrusion detection. Journal of Neural Computing and Applications, 1-8. (2016)
19. Kim M., Kong H., Hong S., Chung S., and Hong J.: A flow based method for abnormal network traffic detection. Network Operations and Management Symposium (NOMS), 599-612. (2004)
20. Kokila R.T, Thamarai S., and Govindarajan K.: DDoS detection and analysis in SDN based environment using Support Vector Machine classifier. 2014 Sixth International Conference on Advanced Computing (ICoAC), 205-210. (2014)
21. Peddabachigari, S., Abraham, A., Grosan, C. and Thomas, J.: Modeling intrusion detection system using hybrid intelligent systems. Journal of Network and Computer Applications, Vol. 30, No. 1, 114-132. (2007)
22. Shon T., Kim Y., Lee C., and Moon J.: A machine learning framework for network anomaly detection using SVM and GA," Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop (IAW), 176-183. (2005)
23. Subbulakshmi T., Shalinie S. M., GanapathiSubramanian V., and BalaKrishnan K.: Detection of DDoS attacks using Enhanced Support Vector Machines with real time generated dataset. Third International Conference on Advanced Computing (ICoAC 2011), 17-22. (2011)
24. Sommer, R. and Paxson, V.: Outside the closed world: On using machine learning for network intrusion detection. IEEE Symposium on Security and Privacy, 305-316. IEEE. (2010)
25. 25. Zhou H., Wu C., Jiang M., Zhou B., Gao W., Pan T., and Huang M.: Evolving defense mechanism for future network security," IEEE Communications Magazine, Vol. 53, Issue. 4, 45-51. (2015)
26. Li K., Zhou W., Li P., Hai J., and Liu J.: Distinguishing DDoS Attacks from Flash Crowds Using Probability Metrics. Third International Conference on Network and System Security, 9-17. (2009)
27. Yu S., Thapngam T., Liu J., Wei S., and Zhou W.: Discriminating DDoS Flows from Flash Crowds Using Information Distance. Third International Conference on Network and System Security, 351-356. (2009)
28. Landset S, Taghi M.K., Aaron N.R., and Tawfiq H.: A survey of open source tools for machine learning with big data in the Hadoop ecosystem." Journal of Big Data, Vol. 2, Issue. 1, 24 (2015)
29. Weka The University of Waikato. Available at https://weka.wikispaces.com/
30. Biswas P. K, Indian Institute of Technology, Lec-29 Support Vector Machine Mod 1. Available at https://www.youtube.com/watch?v=SRVswRH5Q7E
31. Statistics 101: Logistic Regression. Available at https://www.youtube.com/watch?v=zAULhNrnuL4
32. Arlitt M. and Jin T.: 1998 World Cup Web Site Access Logs. Available at http://www.acm.org/sigcomm/ITA/. (1998)

33. The CAIDA: DDoS Attack 2007. Dataset, Center for Applied Internet Data Analysis. Available at https://www.caida.org/data/passive/ddos-20070804_dataset.xml.
34. Mininet: Rapid Prototyping for Software Defined Network. Available at https://github.com/mininet/mininet
35. Zhang, B., Ng, T. S., Nandi, A., Riedi, R., Druschel, P., and Wang, G.: Measurement based analysis, modeling, and synthesis of the internet delay space. Sixth ACM SIGCOMM conference on Internet measurement, 85-98. (2006)

Nishat Mowla received the B.S degree in Computer Science from Asian University for Women, Chittagong, Bangladesh in 2013, an M.S. degree in Computer Science and Engineering from Ewha Womans University, Seoul, Korea in 2016. She is currently a PhD student at Ewha Womans University, Seoul, Korea. Her research interests include next generation network security, IoT network security and network traffic analysis.

Inshil Doh received the B.S. and M.S. degrees in Computer Science at Ewha Womans University, Korea, in 1993 and 1995, respectively, and received the Ph.D. degree in Computer Science and Engineering from Ewha Womans University in 2007. From 1995-1998, she worked in Samsung SDS of Korea to develop a marketing system. She was a research professor of Ewha Womans University in 2009~2010 and of Sungkyunkwan University in 2011. She is currently an assistant professor of Computer Science and Engineering and Cyber Security at Ewha Womans University, Seoul, Korea. Her research interests include wireless network, sensor network security, and M2M network security.

Kijoon Chae received the B.S. degree in mathematics from Yonsei University in 1982, an M.S. degree in computer science from Syracuse University in 1984, and a Ph.D. degree in electrical and computer engineering from North Carolina State University in 1990. He is currently a professor in Department of Computer Science and Engineering at Ewha Womans University, Seoul, Korea. His research interests include sensor network, smart grid, CDN, SDN and IoT, network protocol design and performance evaluation.

Received: March 28, 2017; Accepted: October 20, 2017.