Enhancing Load Balancing by Intrusion Detection System Chain on SDN Data Plane

1st Nadia Niknami
Temple University
Philadelphia, USA
nadia.niknami@temple.edu

2nd Jie Wu
Temple University
Philadelphia, USA
jiewu@temple.edu

2022 IEEE Conference on Communications and Network Security (CNS) | 978-1-6654-6255-6/22/$31.00 ©2022 IEEE | DOI: 10.1109/CNS56114.2022.9947270

***Abstract*—The software-defined network (SDN) allows us to control network flows easily and dynamically. Intrusion detection systems (IDS) are among the controller's applications. The IDS can become overloaded when analyzing a large amount of traffic. Multiple instances of IDSs are recommended across a network to increase processing power. SDN centralized control facilitates the deployment of multiple IDSs on the data plane. This paper proposes a method to deploy some IDS chains, helping the controller increase the detection rate. By grouping flows in a balanced manner and assigning each group to one IDS chain, transmission delay can be reduced. In this study, we formulate an optimization problem to minimize the cost of grouping flows using a modified version of K-means and assigning an IDS chain. We implement our method on a test bed and a trace-based simulation. In various traffic scenarios, our proposed method can satisfy different measurements, such as detection rate and dropping rate, and only increases the delay by a small amount over one IDS scheme.**

***Index Terms*—Attack detection, IDS, grouping traffic, K-means, load balancing, SDN, matching problem.**

I. Introduction

Software-defined networking (SDN) is an emerging architecture that enables a computer network to be intelligently and centrally controlled via software applications. It separates the control plane and data plane of the network [1]. On top of the control plane, there are many network applications running on the application layer. Due to the large scale of traffic as well as different applications that the controller has to handle, an overhead is possible for the controller One of the security applications is the intrusion detection system (IDS). IDS helps the controller to detect anomaly flows and install new rules to block abnormal traffic in the flow tables on the switches. The controller's overhead can be reduced by considering security applications in the data plane. While assigning IDS to all switches can increase detection rates, installing IDS on switches has some costs, and due to a limited budget, it is not practical. In addition to the cost, the process of IDS takes time and makes delays in the transmission process. Providing some chains of IDSs across the data plane would be helpful. This provides a higher detection rate on the data plane and lower overhead on the controller. Fig. 1 illustrates the SDN including the application layer, control plane, and data plane. There are some applications including firewall, load balancer, and routing in the application layer. The yellow boxes in the data plane display IDS components installed on the given switches. There are some flows such as f_1 and f_2 in this network. Each of these flows will be redirected through some IDSs in order to perform intrusion detection.

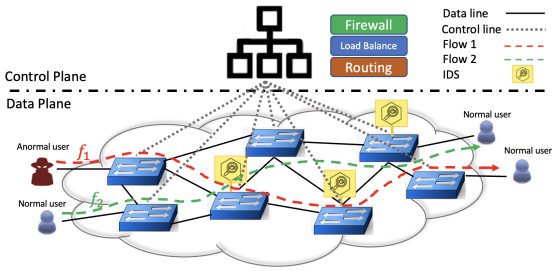


Fig. 1: Redirecting packets through IDSs on the data plane.

One of the measurements to evaluate the performance of the IDS is detection rate. The rate of blocked malicious packets can be displayed by the detection rate which is determined by AB/AR, where AB denotes the number of blocked malicious packets and AR presents the number of received malicious packets. Fig. 2 (a) shows that the more the number of IDSs across the network, the higher the detection rate. Fig. 2 (b) illustrates the impact of a large quantity of incoming traffic on the overhead of controller. Low performance leads to reduced detection rate as well as increased dropping rate. Flow rules are installed by the controller to detect attack flows by redirecting incoming traffic to alternative paths, rather than the shortest path, that go through specific intrusion detection systems. Finding the best path and selecting the best IDS chain for each flow is a difficult task. The following questions need to be addressed:

- IDS has limited hardware resources in terms of CPU power, memory access speed, and storage capacity. Due to these limitations, IDS applications are unable to achieve an acceptable detection rate. Chains of IDSs may provide a solution to this problem. How can multiple IDSs be implemented on the SDN?
- Implementing an IDS chain can improve detection rates. Due to installation costs and flow table capacity limitations, IDS cannot be installed on all switches. Therefore, there are a limited number of IDSs. As incoming traffic is

This research was supported in part by NSF grants CNS 2214940, CPS 2128378, CNS 2107014, CNS 2150152, CNS 1824440, and CNS 1828363.

TABLE I: Main notations

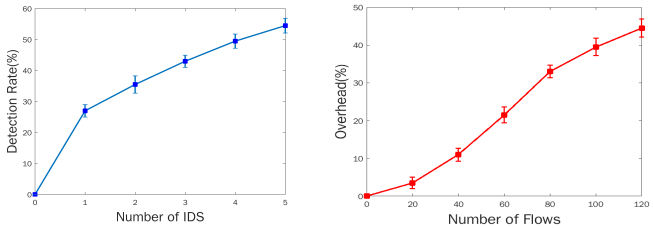
Symbol	Meaning
f, F	Single flow, Set of flows
s_j/d_j	Flow j's source/destination
c_j	Centroid of cluster j
F_j	j^{th} flow group
$\overline{s}_j/\overline{d}_j$	Source/Destination centroid of group j
I/I_i	Set of IDS chain/IDS chain i
h_i/t_i	IDS chain i's head/tail
$dist(.)$	Defined distance
r_j	Traffic rate of flow j
K	Total number of clusters
$G(V, E)$	Network including node set V and link set E

grouped, there is no need for many IDSs. Which method is the best for grouping flows?

- Grouping flows and IDS assigning techniques can have a significant impact on performance measurements, such as dropping rates under high load and transmission delays caused by non-shortest path routing. How can we maintain balanced flow groups? How can flow groups be matched with IDS chains?

Our work involves designing and developing an SDN deploying IDS on some switches in the data plane. In this way, we will reduce the dropping rate, increase the detection rate, and prevent the controller from being overloaded. Forwarding flows from a source to a destination on a network can be problematic when all traffic passes through several specific switches. There is also the issue of flow grouping and matching of IDSs. A trade-off must be considered between detection rate, dropping rate, and delay. Chains of IDSs deployed across the data plane and redirecting flows through these chains satisfy detection rate and overhead. IDSs can be chained in a fixed or dynamic sequence. We call these methods as *total matching* and *partial matching* in this paper. Since there is an IDS on some switches, rather than on all switches, all traffic should take additional hops that deviate from the shortest path to the destination. This will increase latency. Due to the limitations of keeping a large flow table and transmission delay, we can group incoming traffic. This reduces the total number of rules needed in flow tables. All flow in a group follows the same path. Due to the fact that each flow has a source and a destination, K-means clustering is not suitable for this problem. We proposed a modified version of K-means clustering, with a new distance measurement in a 2-D space. The main contributions are as follows:

- We propose a method to provid chains of IDSs on the data plane to increase the rate of intrusion detection and reduce the dropping rate.
- We introduce a creative centroid-based (modified K-means clustering method) to group the incoming flows in order to reduce transmission delay.
- To solve the joint optimization problem, we propose a two-phase algorithm to achieve the optimization goals.
- We provide an in-depth investigation on grouping flows and matching flow groups to IDS chains under different scenarios. We introduce two models for matching flow groups to IDS chain: minimum cost 2-D matching and minimum cost 3-D matching.
- Finally, we evaluate the performance of our approach on a real test bed under different measurements.



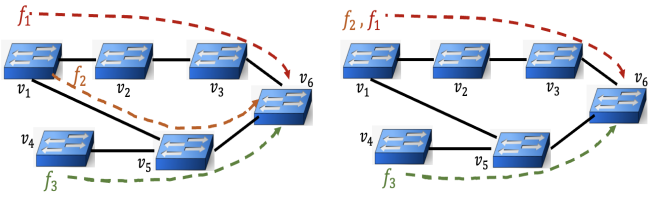
(a) Detection rate with IDS (b) Overhead for controller

Fig. 2: Data plane extension.

II. Related Work

During recent years, there has been a considerable number of researches on SDN network architecture while they integrate IDS into SDN. Studies regarding SDN have been actively conducted as it is considered the next promising networking platform [2]. Authors in [3] introduce a lightweight flow-based IDS that periodically gathers statistical information about flows from SDN switches to detect multiple types of attacks and separate these attacks from real traffic. Manos *et al*. in [4] propose an IDS that automatically detects several DDoS attacks, and then as an attack is detected, it notifies a SDN controller. Authors in [5] presents a new method for identifying attacks on SDNs based on the similarity to existing attacks, as well as a packet aggregation technique designed for the purpose of creating attack signatures and using them to anticipate attacks on SDNs.

It is an important limitation of distributed controllers that the switch-controller mapping is statically configured, resulting in uneven load distribution among controllers [6]. Neghabi *et al*. in [7] review the load-balancing mechanisms which have been used in SDN systematically based on two categories: deterministic and non-deterministic. Authors in [8] propose a new approach which is an SDN-based architecture for attack detection in the data plane. They designed a zone-based architecture in the KIDS to provide scalability and anomaly detection. Cui *et al*. in [9] propose a load-balancing strategy based on the changing characteristics of real-time response times versus controller loads when using multiple SDN controllers. Authors in [10] try to deal with the challenge that how to achieve load balancing without additional device and software on commodity switches while dealing with network/traffic uncertainties. Goo *et al*. in [11] propose a traffic grouping method using the correlation model. They consider the similarity index between two flows by calculating the Euclidean distance value. In our approach, we introduce a novel extension regarding the provision of IDS in the data plane in order to reduce the load on the controller and enhance the rate of attack detection. The data plane is equipped with a chain of IDSs that are connected to several switches. By using



(a) Original route configuration (b) New path for the group

Fig. 3: Original and grouped routes.

a modified version of K-means, incoming flows are assigned to certain groups based on the new measurement for the distance, and then redirect them to IDS chains.

III. Background and Motivation

In SDN, the controller monitors all the switches in the network and sets rules in the flow tables on each switch. The centralized controller communicates with switches through a protocol like OpenFlow [12], and abstracts the data plane's routing and forwarding with a match-action table. The switch sends a packet-in message to the controller whenever a new packet arrives that does not match any of the flow table entries. Also, SDN-based intrusion detection systems (IDSs) identify and report malicious behavior or attacks to network administrators as intrusion events. Therefore, any malicious activity is reported to the network controller. The IDS of SDN are designed currently with a machine learning approach [13].

Clustering algorithms try to decompose the set of nodes into a number of disjoint clusters. The problem is how to select the cluster head or Centroid and how to manage the clusters. A K-means clustering tries to group similar items in the form of clusters [14]. Initially, it starts with a group of randomly chosen centroids, which are used as the starting points for every cluster, and then iteratively optimizes the positions of the centroids. The central point of a cluster is called a centroid. The K-means algorithm identifies K a number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible. Most strategies involve running K-means with different K values and finding the best value using some criterion. The two most popular criteria used are the elbow [15] and the silhouette [16] methods. Balanced clustering is a special case of clustering where, in the strictest sense, cluster sizes are constrained to $\lceil \frac{N}{K} \rceil$ or $\lfloor \frac{N}{K} \rfloor$, where N is the number of points and K is the number of clusters. In our problem, nodes includes servers and switches. Links are the connections between two nodes or the total number of hops between nodes. Also, $|F| = N$ where F is the set of flows and K is the number of IDS chains. The balanced K-means algorithm can be implemented using the Munkres [17] or Hungarian algorithm [18] for solving minimum-cost assignment problems.

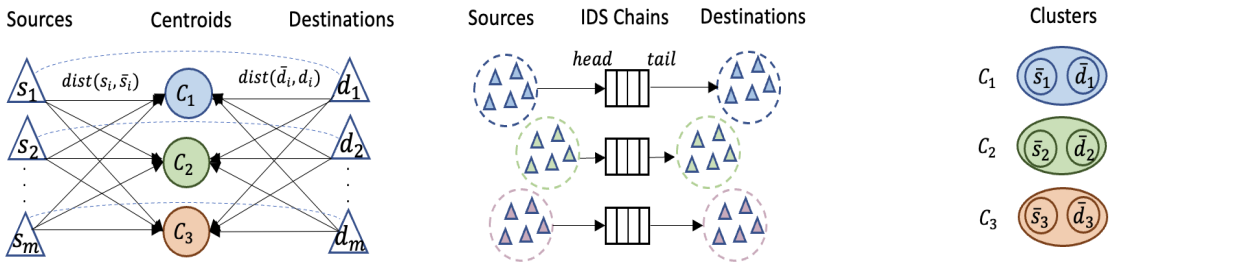
Generally, matching problems seek to find a set of edges such that each vertex belongs to at most one of the edges in the chosen set. Consider a network $G = (V, E)$, where V is the set of nodes and its cardinality. E is the set of links between nodes and its cardinality. The purpose of weighted matching is to assign edges weights in order to identify a set of disjoint edges, edges that do not share a vertex, that have the greatest weight sum. Perfect weighted matching is designed to produce an edge set on a bipartite graph consisting of two equally sized vertex sets [19]. Weighted 3-D Matching utilizes 3-D hypergraphs as an extension of Weighted Matching. The Iterative Round Search approach approximates the general weight 3-D matching problem [20]. There are several possible solutions to the matching problem, including two-round matching, local ratios, Genetic Hungarian Search, and Hungarian Search.

IV. Proposed Method for Deploying IDS Chains

We propose a novel data plane extension for SDNs that addresses the challenges discussed in this paper. It is a routing mechanism that guarantees automatic traffic detection. It is common to find an IDS application in the network application on top of the control plane. Although it has limited capabilities, it is widely used. Furthermore, there is limited capacity for communication between the data plane and control plane. A large volume of traffic can overload the SDN controller and cause the entire network to go down, as SDN is entirely control-based. In this paper, we propose a new approach that allows data plane switches to provide security functions as part of packet processing logic. This is because the controller usually handles a number of applications. The installation of IDS on some switches in the data plane can reduce the burden on the controller significantly. Furthermore, for a given traffic flow, a greater number of IDSs will increase the likelihood that an attack will be detected. Directing flows through some specific path including IDS leads to increased transmission delay. Grouping incoming flows and using the same path for the flows in the same group can reduce this delay. Upon entry into the network, the classifier first categorizes the traffic pattern into suitable categories, following which it assigns the IDS chain that is most appropriate to that traffic pattern. New arrival flows can be determined immediately, allowing the controller to deal with traffic dynamics.

A. The Flow Grouping Solution

Using the proposed approach, incoming flows are grouped based on their source and destination. Then, assign each group to an IDS chain using the matching problem. This problem is NP-hard, and we propose a modified version of K-means as an approximation solution. Lloyd's algorithm is the most popular way of solving K-means. As a basic concept, we will group the flows and add a set of rules that apply to the group of flows. First, we classify the flows based on the distance of source and destination hosts from the centroid of the clusters. Next, we assign one IDS chain to each flow group. The controller installs flow rules based on the redirecting of packets of flows in each group to go through a specific number of IDSs in the assigned chain. Security actions that use the same cluster ID are considered as belonging to the same cluster. The clustered action operates as a single integrated action across a variety of flow rules. Fig. 3 illustrates an example of grouping flows. For this toy example, f_1 and f_2 are combined as one group.



(a) Grouping flows (b) Assigning flow groups to IDS chain (c) Source and destination of centroids

Fig. 4: Grouping flows, matching, and assigning IDS chains.

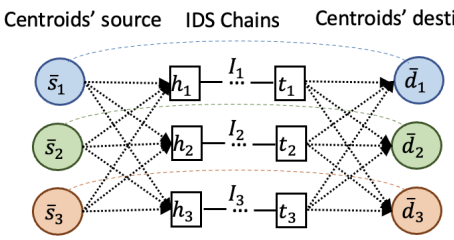


Fig. 5: Total matching.

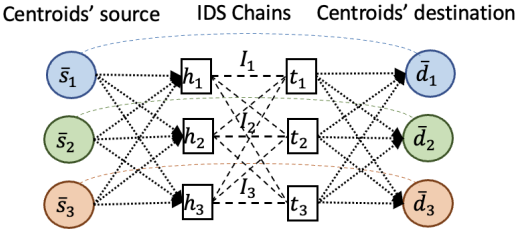


Fig. 6: Partial matching.

By doing so, these flows are redirected along a single path rather than multiple paths. As a result of grouping flows, f_2 will pass through a longer path (3 hops instead of 2 hops) from its source to its destination.

Theorem 1. The flow grouping problem is NP-hard, which implies that we should not expect efficient algorithms that find optimal solutions.

Proof: The proof has been shown in [21] and [22] with an Exact Cover by 3-Sets and a reduction from Planar 3-SAT. ■

Definition 1. (Grouping strategy) A grouping strategy Δ, defines a partition $\{F_1, ..., F_K\}$ of F, where $F_j \subseteq F$ is referred to as the j^{th} group. Each grouping strategy partitions the inputs based on the similarity of specific features. The total number of partitions or groups can be a predefined parameter.

Definition 2. (GroupFlow) A GroupFlow includes multiple flows that have been forwarded through the same path. The path of a group flow depends on the location of IDS chains which are deployed in the network to detect anomalies.

With these two definitions, we need to define grouping strategy and similarity measurement. K-means can be used as a grouping strategy for flow set F. Each data point in the K-means clustering is a flow $f \in F$. K-means is based on measuring the distance between data points and their centroids. Here, we have a different problem and a different similarity measurement in comparison with the general version of K-means. In our problem, data points are flows which have source s_j and destination d_j. The modified K-means clustering should be done for pair (s_j, d_j), where distance measurement is $dist(s_j, \bar{s}_k) + dist(d_j, \bar{d}_k)$ for a cluster with center c. The steps of the proposed approach are as follows:

- Run modified K-means clustering for pair (s_j, d_j), where distance measurement is $dist(s_j, \bar{s}_k) + dist(d_j, \bar{d}_k)$ for a cluster with center c_k. The distance between each host and the center of cluster is the total number of hops.
- Run balanced K-means algorithms for each cluster.
- Use the standard perfect matching to pair cluster centers with chain of IDSs.

Balanced clustering involves an equal number of points in each cluster. Our approach to balanced K-means clustering differs from those commonly used. To determine if groups are balanced, we use the total data rate of the groups rather than the number of group members. Once the cluster is formed from modified K-means clustering, the cluster center becomes the representative for bipartite matching.

Definition 3. (Distance) The distance measurement is the summation of the distance of each source s_j to the cluster centroid c_k's source and distance of each destination d_j to the cluster centroid c_k's destination. Distance value $dist(s_j, \bar{s}_k) + dist(d_j, \bar{d}_k)$ is used to finding the nearest cluster centroid for each flow group.

For any clustering method, the important question is whether the particular estimations converge in an appropriate sense. In order to answer this question, we need to establish a related optimization problem and make the notion of convergence precise . The word convergence means the algorithm has successfully completed this clustering or grouping of data points in K clusters. The algorithm will make sure it has completely grouped the data points into correct clusters, if the difference in the values of the last two iterations is less than a particular threshold. Classical K-means is designed for Euclidean distance, which happens to satisfy the triangle inequality. Using the triangle inequality is required in order to find the bounds to avoid redundant distance calculations. Based on the fact that most distance calculations using standard K-means are redundant, the optimized algorithm uses

Algorithm 1 Balanced-Modified-K-means

Require: Flow set F
Initialize the K cluster centroids
repeat
  for each (s_f, d_f) of flow $f \in F$ **do**
    for each centroid c_k **do**
      $\overline{s}_k \leftarrow$ source of c_k, $\overline{d}_k \leftarrow$ destination of c_k
      $j \leftarrow \arg\min_k(dist(s_f, \overline{s}_k) + dist(d_f, \overline{d}_k))$
      $F_j \leftarrow F_j \cup f$
    end for
  end for
  for each cluster F_j **do**
    Calculate $R_{F_i} = \sum_{f \in F_j} r_f$
    Find c'_j as new centroid based on R_{F_i}
  end for
  Update centroids $c'_1, c'_2, ..., c'_K$
until Convergence
return List of clusters and their centroids

Algorithm 2 Matching IDS chains (Total matching)

Require: IDS chains I and set of K clusters
for each unmatched centroid c_k **do**
  $\overline{s}_k \leftarrow$ source of c_k, $\overline{d}_k \leftarrow$ destination of c_k
  for each IDS chain $i \in I$ **do**
    $h_i \leftarrow$ head of i
    $i^* \leftarrow \arg\min_i\{dist(s_k, h_i)\}$
  end for
  Assign $\overline{s}_k$ to h_{i^*}, Assign $\overline{d}_k$ to t_{i^*}
end for
return List of matched IDS chains and clusters

a more efficient calculation method. If a point is far from a center, it is not necessary to calculate the exact distance between that point and the center to know that it should not be assigned to that center. On the other hand, if a point is substantially closer to one center than to any other, calculating the exact distance is not necessary to determine that the point should be assigned to that first center. There is usually a latency added to network traffic when using an IDS, and the more complex the inspection of any packet is, the longer the packet is delayed on its forwarding. We propose balanced modified K-means and matching problems in 2-D and 3-D. We take the traffic rate of flows into account to have balanced groups in the process of flow grouping. Otherwise, large traffic leads to dropping packets due to overloading on the IDS chain. The proposed method of assigning IDS chains to the flow groups is based on the number of hops between the centroid and IDS chain. We introduce total matching and partial matching models.

B. Matching Flow Groups with IDS Chain

Once the incoming flows are grouped by modified K-means clustering, all flow groups will be assigned to the IDS chains deployed in the data plane.

Definition 4. (Matching) For a given group F_j, there is a matched IDS chain based on the matching of centroid c_k with the specific IDS chain. The head of each cluster of flows should be assigned to one IDS chain. This matching is based on the weight of each link between the head of the cluster and the head or tail of the IDS chain.

Algorithm 3 Matching IDS chains (Partial Matching)

Require: IDS chains I and set of K clusters
for each unmatched centroid c_k **do**
  $\overline{s}_k \leftarrow$ source of c_k, $\overline{d}_k \leftarrow$ destination of c_k
  for each IDS chain $i \in I$ **do**
    $i^* \leftarrow \arg\min_i dist(\overline{s}_k, h_i)$
    $j^* \leftarrow \arg\min_i dist(t_i, \overline{d}_k)$
  end for
  Assign $\overline{s}_k$ to h_{i^*}, Assign $\overline{d}_k$ to t_{j^*}
end for
return List of matched IDS chains and clusters

Fig. 4 (a) shows the first step, which is calculating the distance between sources and destinations of flows and initial centroids. We have the distance measurement $dist(s_j, \overline{s}_k) + dist(d_j, \overline{d}_k)$ and flows would be divided into three clusters with centroids $\{c_1, c_2, c_3\}$. Fig. 4 (b) illustrates details of our defined centroid. It includes source $\overline{s}_k$ and destination $\overline{d}_k$. After grouping the flows, each group will be assigned to one IDS chain. Fig. 4 (c) shows the assigning of the heads and tails of the IDS chains to the source and destination of the centroids. h_i is the head of IDS chain i and t_i is the tail of this IDS chain. All flows in a cluster k get a virtual center, including source $\overline{s}_k$ and destination $\overline{d}_k$. We introduce two models for matching the centroid of flow groups to IDS chains: 1)Total matching which is assigning the $\overline{s}_k$ and $\overline{d}_k$ to the h_i and t_i of IDS chain i respectively. This is a 2-D minimum cost perfect matching. 2)Partial matching which is assigning $\overline{s}_k$ to h_i and assigning $\overline{d}_k$ to t_j where h_i and t_j are the head and tail of two different IDS chains. This is a 3-D minimum cost perfect matching. Figs. 5 and 6 illustrate the total matching and partial matching models, respectively. Fig. 5 shows a fixed IDS chain for each flow group. Fig. 6 shows IDS chains which are not fixed (in the case of sequential order) for each flow group. In this type of IDS chain, there are cross connections between IDS chains. There are two problems that can be classified as grouping incoming traffic and IDS assignment issues. These two important problems can be formulated as follows:

Problem 1. Grouping incoming traffic to reduce transmission delay in a balanced way. The distance of flows to the cluster's centroid and the total amount of traffic in each cluster are important factors that should be taken into consideration. This problem is NP-hard, and we provide an approximation based on the grouping of the incoming flows with the help of the modified version of K-means clustering. We formulate the grouping incoming traffic problem as an optimization problem with an objective of minimizing overhead/cost.

$$\begin{aligned} \min \quad & \sum\nolimits_{F_j \in F} cost(F_j) \\ \text{subject to} \quad & cost(F_j) = |F_j| \cdot \sum\nolimits_{f \in F_j} r_f \end{aligned} \tag{1}$$

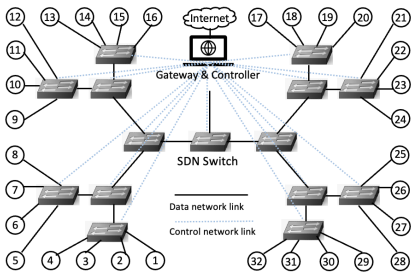
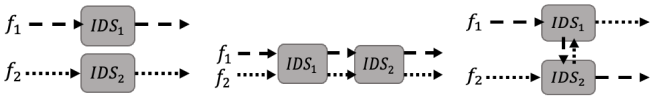


Fig. 7: Datacenter topology.



(a) One IDS (b) Same IDS (c) Mixed IDS

Fig. 8: Different combination of IDS.

Here, $cost(F_j)$ represents the cost of clustering incoming traffic f. The cost represents the overhead of the controller due to any extra work for grouping the incoming traffic. This is based on the total number of flows and total traffic rate r_f in each cluster F_j. For simplicity, we assume that $r_f = 1$.

Problem 2: Find an IDS chain assignment for each flow group so that the total number of malicious packets is minimized by ensuring that all the traffic is forwarded to an IDS chain before reaching the destination. We assume that the locations of IDS chains are predetermined. The problem can be expressed as the following:

$$\begin{aligned} \min \quad & \sum\nolimits_{i \in I} cost(I) \\ \text{subject to} \quad & cost(I) = \sum\nolimits_{M_{j,i}=1} R_j * \min dist(F_j, I_i) \\ & R_j = \sum\nolimits_{f \in F_j} r_f \\ & 1 \leq |I_i| \end{aligned} \tag{2}$$

Here, $cost(I)$ represents the cost of assigning a flow group to an IDS chain I_i. This is based on the total traffic rate of each flow group and the distance between the centroid of the flow group and the IDS chain. R_j denotes the total traffic rate of flow group j. r_f denotes the data rate of flow f. The distance between IDS chain I and flow group F_j is shown by $dist(F_j, I_i)$, which is the number of hops between the h_i and $\overline{s}_j$. $M_{j,i}$ is a matrix that shows each flow group F_j is assigned to which IDS chain I_i. Grouping the flows is done by a modified version of K-means clustering. Algorithm 1 presents the steps of grouping the incoming traffic into K clusters. The first step is initializing the K cluster centroids randomly. The second step is finding the distance of each pair (s_j, d_j) to centroid c_k and assigning flows to each cluster. After clustering all the flows, it is necessary to update the centroids and find $\{c'_1, c'_2, ..., c'_K\}$ such that the sum of the shortest path distance from all flows in the cluster j to the new centroid c'_j is minimized. This process continues until it converges to a stable state. Since we consider traffic rate r_f for updating the centroids, the convergence result applies to balanced K-means. This algorithm returns a list of clusters and their centroids. We introduce two models for matching step. Algorithm 2 presents the details of the total matching method for matching clusters with fixed IDS chains. The first step is the matching process based on the distance of each cluster centroid c_k and the head and tail of the IDS chain i. Each flow group is assigned to the IDS chain i^* with the minimum distance. $\overline{s}_j$ will be assigned to h_{i^*} and $\overline{d}_j$ will be assigned to t_{i^*}. This is the 2-D minimum cost perfect matching. The output of this algorithm is the list of matched IDS chains and clusters.

Algorithm 3 presents the details of the partial matching method for matching clusters with dynamic sequence of IDS chains. Each flow group is assigned to an IDS chain after finding the best head of chains i^* and the best tail of chains j^*. $\overline{s}_j$ will be assigned to h_{i^*} and $\overline{d}_j$ will be assigned to t_{j^*}. This is the 3-D minimum cost perfect matching. The output of this algorithm is the list of matched IDS chains and clusters. We assume that the total number of groups is the same as the total number of IDS chains. For generalization, this is solvable as long as the total number of groups is smaller than the number of IDS chains [23]. In this case, the assigning flow groups to IDS chains should be a type of partial assignment.

Theorem 2. The complexity of the Algorithm 1 is $O(KEL)$. The complexity of the Algorithm 2 is $O(K^2(V+E))$.

Proof: The most time-consuming part of this algorithm is the K-means clustering, which takes $O(KEL)$, where E is the number of links in the flow groups and L is the number of iterations until convergence. For Algorithm 2, the complexity of finding a path is $O(V+E)$. The complexity of finding all the paths of K group flows is $O(K(V+E))$. The most time-consuming part of this algorithm is the matching which takes $O(Klog(K))$, Therefore, the algorithm takes $O(K^2(V+E))$. ■

V. Evaluation

We conduct real experiments on our test bed to validate our approach. In our network, we have gateway nodes, SDN switches, and some servers serving as sources and destinations. It is a perfect tree topology with four layers. There are 32 servers, 15 SDN switches, and some regular L2 switches, shown in Fig. 7. We have a Dell 3248 PowerEdge server as our controller, which is running an ONOS platform. IDSs are installed on the associated server for each switch. After describing the network and flow settings, we analyze the results from various perspectives to provide an insightful interpretation. To demonstrate the feasibility and efficiency of the proposed algorithm, we conduct the experiment evaluation on a real test bed. Fig. 8 displays different combinations of IDSs. Fig. 8 (a) illustrates a scenario in which each flow passes through one IDS. Fig. 8 (b) shows the scenario in which multiple flows go through the same IDS sequence. Fig. 8 (c) presents the mixed IDS, when for each of IDSs, there is one full flow as well as a partial amount of another flow. In other words, there is full flow f_1 for IDS_1 and then just a partial flow f_1 will pass through IDS_2. In fact, the partial flow of f_1

TABLE II: Comparison of one IDS vs multiple IDSs

Traffic	Attack Rate	Detection Rate(%)			Dropping Rate(%)			Delay(msec)		
		Single	2 IDS	Mixed	Single	2 IDS	Mixed	Single	2 IDS	Mixed
Small	20%	36.6	48	52	24.9	26.3	25	1.8	3.45	3.3
	50%	47.5	55	60	25.5	26.9	26.2	3.6	6.9	6.45
	80%	52	69	72	24.8	26.7	25.1	6.1	11.31	10.8
Medium	20%	49.3	64.5	74.5	28.7	30.5	29.9	5.55	9.99	9.57
	50%	60.3	71	73	28	29.5	28.9	7.1	15	14.1
	80%	72	81	83	28.9	32	31.5	13.5	24.9	24.51
Large	20%	61.8	80.3	85	31.2	34	32.7	9.6	17.4	16.5
	50%	74.1	86	91	34.5	36.3	35.18	17.1	33.3	32.82
	80%	81	92	94.3	35	37.5	38.7	30	54.6	54

TABLE III: Effects of clustering methods on overhead and detection rate for an IDS chain with one IDS

Clustering Method	Overhead (%)	Detection Rate (%)	Delay (ms)
K-means and random assigning	21%	45%	2.7
K-means and total matching	27.5%	64.5%	3.33
K-means++ and total matching	31.2%	64.7%	4.1
Balanced K-means and total matching	35.7%	74%	6.32
Balanced K-means and partial matching	36%	81%	6.4

TABLE IV: Effects of IDS in control plane and data plane under different amounts of incoming traffic

Anomaly Detection	Ctr-Overhead(%)			Dropping Rate(%)			Detection Rate(%)			Delay (ms)		
	S	M	L	S	M	L	S	M	L	S	M	L
Centralized IDS	7	12	27	32.6	37	43.2	39.4	53.3	68.3	2.7	5.3	19.2
Chain with one IDS	10.2	12.3	17.8	31	28.5	33.8	38.5	60.3	74.1	3.6	7.1	23.1
Chain with two IDS	10.3	12.3	18	32.9	31.5	35.6	55	71	86	9.6	15	30.3

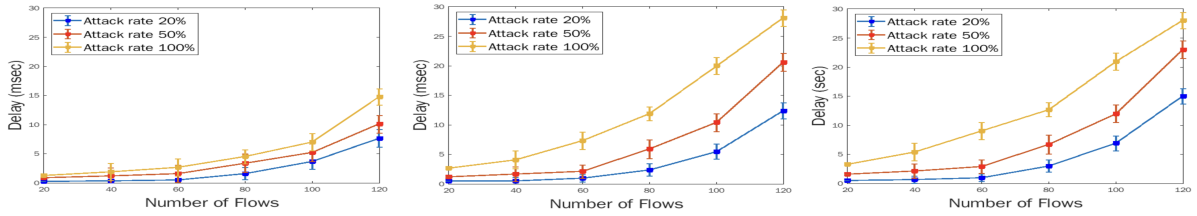
consists of the packets, including the normal ones and the abnormal missing ones. We install Snort on some Ubuntu servers to perform as an IDS in our network. The legitimate traffic was generated by using Ostinato traffic generator. The Ostinatoas a network traffic generator can be used in normal mode and burst mode to generate legitimate traffic. For the purpose of generating malicious traffic, Kali Linux was used.

A. Experimental Results

We evaluate our approach using measurements of the percentage of detected packets, the percentage of dropped packets, missing rate, and delay that are crucial to intrusion detection and packet forwarding. It is possible for legitimate packets and malicious packets to be dropped if the server receives more traffic than it can handle. We calculate the dropping rate to find the number of dropped packets. If IDS processed the packets but did not detect any anomalies in them, then those packets are forwarded to the next switch.

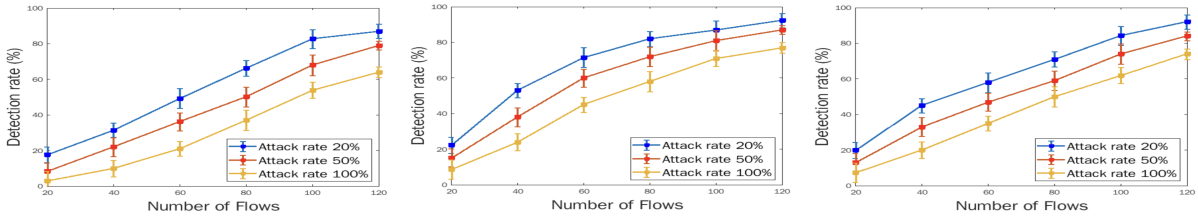
1) Performance under One IDS vs multiple IDSs: Table II presents the comparison between single IDS versus multiple IDSs regarding detection rates, dropping rates, and delays. Different amounts of traffic were classified as small, medium, and large traffic. The size of traffic is based on the total amount of incoming traffic. We consider 500 flows as small traffic, 2000 flows as medium traffic, and 8000 flows as large traffic. Additionally, we tested different measurements under different attack rates, namely $20\%, 50\%$, and 80%. According to the results, assigning more IDSs has a positive impact on the detection rate for malicious packets. There would be a lower missing rate, but a higher dropping rate. It is evident that considering multiple IDSs increases the delay time as compared to one IDS. However, the increase in the delay is not substantial. The reason for this is that some portions of traffic were blocked or dropped by the previous IDS. This would result in a smaller amount of incoming data for the later IDSs. The increasing of the attack rate increases the detection rate and decreases the missing rate. This is due to the fact that IDS is based on a machine learning algorithm, and if there are more samples for detecting packets of attack, the likelihood of detection would be higher. The results indicate that attack rate does not influence dropping rate. The dropping rate is a result of the limited capacity of switches and is not determined by the ratio of malicious packets or legitimate packets. The delay is increased by increasing the attack rate, since switches need to send alarms to the controller in order to take appropriate action. Large traffic results in increasing the detection rate, missing rate, and dropping rate due to having more attacks.

Fig. 9 illustrates the delay time as a measure of the performance of one IDS, multiple IDSs, and a mixed IDS under different amounts of incoming traffic. The results of delay time for multiple IDSs and mixed IDS are similar. With a single IDS, there is less delay time than with multiple IDSs. Fig. 10 displays the detection rate of one intrusion detection system, multiple intrusion detection systems, and a mixed intrusion detection system under changing amounts of incoming traffic. We have a higher detection rate when we use mixed IDS, but it is still close to the detection rate when we deploy multiple IDSs. The probability of detecting anomalous flows increases when multiple IDSs are used instead of a single IDS. Fig. 11



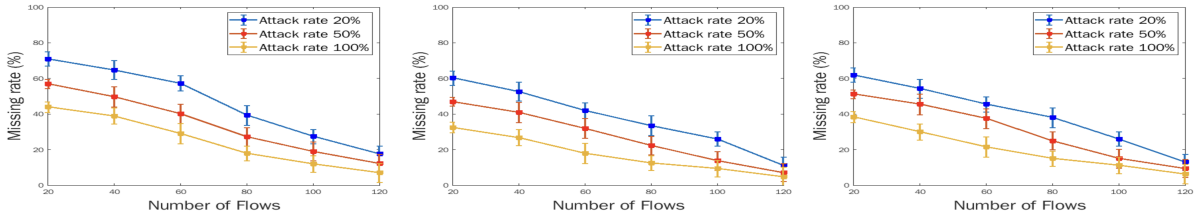
(a) One IDS (b) multiple IDSs (c) Mixed IDS

Fig. 9: Delay time.



(a) One IDS (b) multiple IDSs (c) Mixed IDS

Fig. 10: Detection rate.



(a) One IDS (b) multiple IDSs (c) Mixed IDS

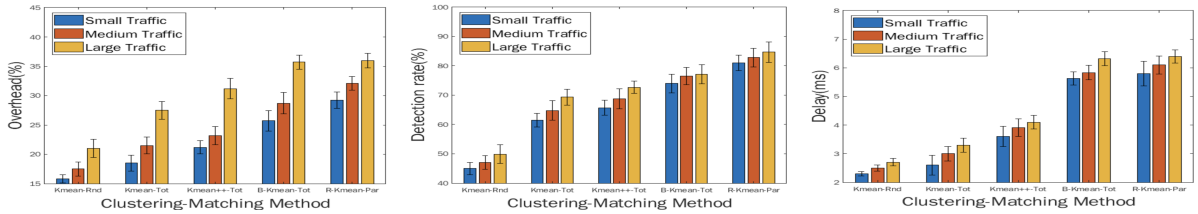
Fig. 11: Missing rate.

shows missing rate for different amounts of traffic. There are similar results for all scenarios.

2) Performance under different grouping and matching methods: One practice for reducing the controller overhead is to install a rule for minimal flow entry in network switches. Table III shows the effect of different clustering methods on overhead of controllers in comparison with general anomaly detection, dropping rate, and the detection rate on the switches in the data plane. The traditional K-means method groups the flows in K clusters without considering any limitation on the size or number of the members in each cluster. K-means is dependent on initial values. K-means++ removes the drawback of K-means. The difference between the two methods is in the process of picking the initial centroids. The original version of balanced K-means provides K clusters, each of which has the same number of members. Our balanced K-means divides the flows in K clusters. Clusters have the same amount of total data rate, rather than the total number of members. Table III shows that balancing the clusters based on the data rate has a positive impact on the dropping rate of IDS. Fig. 12 illustrates the performance of different clustering methods in terms of overhead, detection rate, and delay. In terms of detection rate, our proposed clustering approaches produce the most promising results, but have a high degree of delay and overhead compared to other clustering approaches.

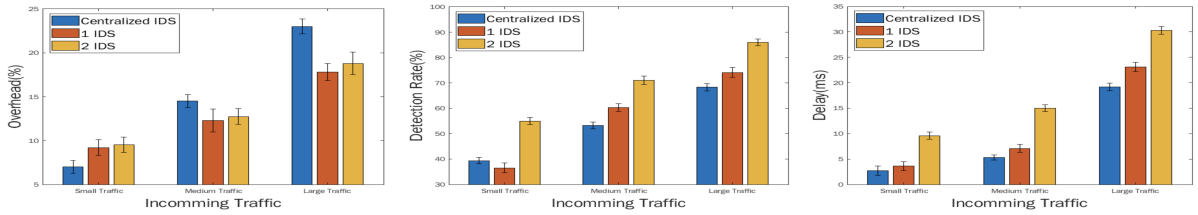
Table IV shows the effect of providing IDS in the control plane and data plane with different measurements of overhead, missing rate, dropping rate, detection rate, and delay. It appears that the amount of incoming traffic is an important measure to evaluate IDS deployment in the data plane. Moreover, the number of IDSs in each chain affects all measurements. Fig. 13 shows the outcome of IDS deployment in the data plane as a centralized or chained arrangement. Therefore, two IDSs chained together provide a higher detection rate.

According to the results, more IDSs increase detection rates for malicious packets. In addition, the missing rate would be lower, but the dropping rate would be higher. When using multiple IDSs, delay time increases compared to using a single one. This is because there is an alternative path that involves several servers, including the IDS chain, rather than the shortest path. Nevertheless, it is not significant. It has been shown that increasing attack rates increases detection rates and decreases missing rates in general. As IDS is based on a machine learning algorithm, if there are more samples for detecting packets of attack, there is a higher probability of detection. Based on the results, dropping rates are not affected by attack rates. Dropping rates are a result of the limited capacity of switches and are not determined by the ratio of malicious packets to legitimate packets. As the attack rate increases, the delay increases. This is due to the fact that switches need to send alarms to the controller before any appropriate action can be taken. With more attack samples,



(a) Overhead (b) Attack detection rate (c) Delay

Fig. 12: Detection rate, overhead, and delay for different clustering methods.



(a) Overhead (b) Attack detection rate (c) Delay

Fig. 13: Comparison between centralized IDS, 1 IDS, and 2 IDS for detection rate, overhead, and delay.

the detection rate, missing rate, and dropping rate increase.

VI. Conclusion

If a network is large and busy, a single IDS may be vulnerable to packet loss, so an IDS chain setup is essential. We proposed a novel solution for applying attack detection to incoming traffic in the data plane by creating an approximation model. We applied a modified version of the K-means method to group the incoming flows with intuitive insights. Flows in each group are processed into an IDS chain as a new flow. Assigning of flows to IDS chains leads to grouped flows detouring on a longer path and being processed by the assigned IDS chain. We discussed several factors, such as detection rate, missing rate, dropping rate, and delay time to evaluate our approach.

References

[1] J. H. Cox, J. Chung, S. Donovan, J. Ivey, R. J. Clark, G. Riley, and H. L. Owen, "Advancing software-defined networks: A survey," *IEEE Access*, vol. 5, pp. 25 487–25 526, 2017.

[2] K. Shipulin, "We need to talk about ids signatures," *Network Security*, vol. 2018, no. 3, pp. 8–13, 2018.

[3] G. A. Ajaeiya, N. Adalian, I. H. Elhajj, A. Kayssi, and A. Chehab, "Flow-based intrusion detection system for sdn," in *Proc. of IEEE Symposium on Computers and Communications (ISCC)*, 2017, pp. 787–793.

[4] P. Manso, J. Moura, and C. Serrão, "Sdn-based intrusion detection system for early detection and mitigation of ddos attacks," *Information*, vol. 10, no. 3, p. 106, 2019.

[5] A. AlEroud and I. Alsmadi, "Identifying cyber-attacks on software defined networks: An inference-based intrusion detection approach," *Journal of Network and Computer Applications*, vol. 80, pp. 152–164, 2017.

[6] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, A. Vahdat *et al.*, "Hedera: dynamic flow scheduling for data center networks." in *Nsdi*, vol. 10, no. 8, 2010, pp. 89–92.

[7] A. A. Neghabi, N. J. Navimipour, M. Hosseinzadeh, and A. Rezaee, "Load balancing mechanisms in the software defined networks: a systematic and comprehensive review of the literature," *IEEE Access*, vol. 6, pp. 14 159–14 178, 2018.

[8] A. Yazdinejadna, R. M. Parizi, A. Dehghantanha, and M. S. Khan, "A kangaroo-based intrusion detection system on software-defined networks," *Computer Networks*, vol. 184, p. 107688, 2021.

[9] J. Cui, Q. Lu, H. Zhong, M. Tian, and L. Liu, "A load-balancing mechanism for distributed sdn control plane using response time," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1197–1206, 2018.

[10] H. Wang, H. Xu, C. Qian, J. Ge, J. Liu, and H. Huang, "Prepass: Load balancing with data plane resource constraints using commodity sdn switches," *Computer Networks*, vol. 178, p. 107339, 2020.

[11] Y.-H. Goo, S.-H. Lee, S. Choi, M.-J. Choi, and M.-S. Kim, "A traffic grouping method using the correlation model of network flow," in *Proc. of IEEE 19th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, 2017, pp. 386–390.

[12] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[13] Y. Hande and A. Muddana, "A survey on intrusion detection system for software defined networks (sdn)," in *Research Anthology on Artificial Intelligence Applications in Security*, 2021, pp. 467–489.

[14] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations," in *Proc. of 5th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, no. 14, 1967, pp. 281–297.

[15] E. Umargono, J. E. Suseno, and S. V.-c. Gunawan, "K-means clustering optimization using the elbow method and early centroid determination based on mean and median formula," in *Proc. of 2nd Intl. Seminar on Science and Technology (ISSTEC)*, 2020, pp. 121–129.

[16] K. R. Shahapure and C. Nicholas, "Cluster quality analysis using silhouette score," in *Proc. of IEEE 7th Intl. Conf. on Data Science and Advanced Analytics (DSAA)*, 2020, pp. 747–748.

[17] X. Chang, F. Nie, Z. Ma, and Y. Yang, "Balanced k-means and min-cut clustering," *arXiv preprint arXiv:1411.6235*, 2014.

[18] R. Burkard, M. Dell'Amico, and S. Martello, *Assignment problems: revised reprint*. SIAM, 2012.

[19] R. M. Karp, "An algorithm to solve the m× n assignment problem in expected time o (mn log n)," *Networks*, vol. 10, no. 2, pp. 143–152, 1980.

[20] T. Johnson and J. Wu, "Improvements to worker assignment in bike sharing systems," in *Proc. of IEEE 18th Intl. Conf. on Mobile Ad Hoc and Smart Systems (MASS)*, 2021, pp. 639–644.

[21] A. Vattani, "The hardness of k-means clustering in the plane," *Manuscript*, vol. 617, 2009.

[22] M. Mahajan, P. Nimbhorkar, and K. Varadarajan, "The planar k-means problem is np-hard," in *Proc. of Intl. Workshop on Algorithms and Computation*, 2009, pp. 274–285.

[23] D. G. McVitie and L. B. Wilson, "Stable marriage assignment for unequal sets," *BIT Numerical Mathematics*, vol. 10, no. 3, pp. 295–309, 1970.