This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2021.3054522, IEEE Transactions on Information Forensics and Security

Sampling Rate Distribution for Flow Monitoring and DDoS Detection in Datacenter

Rajorshi Biswas, Sungji Kim, and Jie Wu
Department of Computer and Information Sciences
Temple University, Philadelphia, PA, USA
{rajorshi, sungji.sj.kim, jiewu}@temple.edu

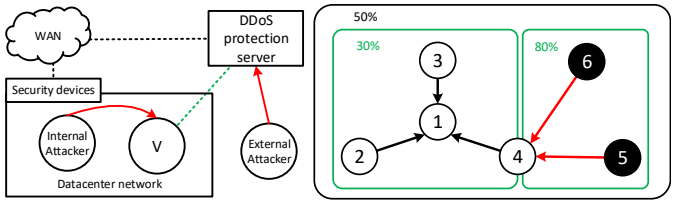
Abstract—Monitoring all the internal flows in a datacenter is important to protect a victim against internal distributed denial-of-service (DDoS) attacks. Unused virtual machines (VMs) in a datacenter are used as monitors and flows are copied to the monitors from software defined networking (SDN) switches by adding some special rules. In such a system, a VM runs a machine learning method to detect DDoS behavior but it can only process a limited number/amount of flows. When the amount of flows is beyond the capacities of all monitor VMs, the system sub-samples each flow probabilistically. The sampling rate affects the DDoS detection rate of the monitors. Besides, the DDoS detection rates of different types of flows are different for the same sampling rate. A uniform sampling rate might not produce a good overall DDoS detection rate. Assigning different sampling rates to different flows may produce the best result. In this paper, we propose a flow grouping approach based on behavioral similarity among the VMs followed by hierarchical clustering of VMs. The sampling rate is uniform among all the flows in a group. We investigate the relationship between the sampling rate and the DDoS detection rate. Then, we formulate an optimization problem for finding an optimal sampling rate distribution and solve it using mix-integer linear programming. We conduct extensive experiments with Hadoop and Spark and present results that support the feasibility of our model.

Index Terms—*botnet, DDoS defense, DDoS, flooding attack, network security, sampling rate distribution.*

1 Introduction

A denial-of-service (DoS) attack is a cyber attack that aims to make a machine or service temporarily unavailable to its users. Attackers send a huge number of requests to the victim machine. There are several types of DoS attacks, including SYN flood and UDP flood [1]. The goal of these attacks is to fill-out the limited slots for the half-open connection so that the server cannot accept a new TCP connection. The purpose of the UDP flood attack is to consume all of the available network bandwidth of the victim. In this attack, the payloads of the attack packets are kept large. In a distributed denial-of-service (DDoS) attack, many attackers - controlled by a master - simultaneously send requests to the victim.

Based on the locations of attackers, we can divide DDoS attacks in a datacenter into two types: internal and external, as shown in Fig. 1(a). In external DDoS attacks, the attackers reside outside the datacenter. A commercial DDoS protection service cannot defend against the internal DDoS attacks. The filter-based DDoS attack defense mechanisms such as [2] also cannot protect against internal DDoS attacks. The clients of a commercial DDoS protection point their domain to the commercial DDoS protection server. The protection server creates a private connection with the client's server and passes all the requests and responses between the client's server and users. When the attackers know the internal or the public IP address and send attack packets to that IP address, the packets do not travel through the commercial DDoS protection server. The security modules of a datacenter usually remain in the aggregation or core



(a) Internal and external attackers. (b) Distribution of sampling rate.

Fig. 1: Attacker and sampling rate distribution.

layer of the network. Therefore, internal flows usually do not go through the security modules. As a result, internal flows are not monitored.

One way to protect the server against internal DDoS attacks is to block all internal flows, but there are some applications, including Hadoop and Spark, that need internal communication. We conducted an experiment to show how an internal DDoS attack affects Hadoop. Table 1 shows the effect of a SYN flood attack on the Hadoop master from fourteen bots. The bots produce a SYN attack using hping3 tools. The bots fire SYN packets with an interval of 1 ms. Each packet is 66 bytes long. With this low attack rate, the mapper time increases to almost double that of without an attack. With the attack, the MapReduce job failed 50% of the time. Details will be presented in Section 5.

To protect against internal DDoS attacks, we need to monitor all the internal flows. To monitor all the internal flows, unused virtual machines (VMs) are used as monitors. The monitors run some machine learning approaches to detect attack behavior in the flows. Obviously, there is a

TABLE 1: Effect of SYN flood attack.

Measurements	Normal	Attacked
Killed map tasks	1.18	3.66
MapReduce success rate	100%	50%
Total time for map (ms)	1122082	2116012
Total time for reduce (ms)	360243	438762

limited number of VMs and capacity of a VM is limited. We assume a VM can monitor a limited number of flows. If the number of flows is larger than the total capacity, then each flow is partially copied to the monitors. The partial copy is done probabilistically. Flow copying is done by the software defined networking switches. When a flow is sub-sampled, the DDoS attack detecting rate is also reduced. Sub-sampling a flow is probabilistically taking a portion of the packets in that flow. The rate of detection also depends on the type of attack.

Let us assume that we have a DDoS detection mechanism that works based on the arrival time intervals of packets. If a flow has low arrival time intervals, then it is classified as a DDoS flow. In the datacenter, we have five flows, as shown in Fig. 1(b). We assume all the flows have the same data rate. Flows $2 \rightarrow 1$, $3 \rightarrow 1$, and $4 \rightarrow 1$ are normal flows that have a high interval of packets. Other flows, $6 \rightarrow 4$ and $5 \rightarrow 4$, are DDoS flows that have a small interval of packets. The monitoring system can handle 50% of the total packets. If we sample 50% from each flow and send them to the monitor, the packet arrival interval will increase for the DDoS flows. Therefore, flows $6 \rightarrow 4$ and $5 \rightarrow 4$ might be classified as normal flows. Let us divide the flows into two groups such as flows $2 \rightarrow 1$, $3 \rightarrow 1$, and $4 \rightarrow 1$ in group 1 and flows $6 \rightarrow 4$ and $5 \rightarrow 4$ in group 2. If we sample 30% from group 1 and 80% from group 2, then the overall sampling rate remains 50%. A higher sampling rate for group 2 increases the probability of DDoS flow detection. On the other hand, 30% samples from group 1 slightly reduce the DDoS detection rate. So, the overall rate is higher than the previous approach.

It is challenging to find a perfect grouping of flows. The relationship between the detection rate and the sampling rate is important. By using the relationship between the sampling rate and the detection rate of each group, we can get an optimal sampling rate distribution with the help of a traditional optimization solver in the given groups of flows. There is also a trade-off between the number of groups and computation complexity of finding an optimal sampling rate distribution. Therefore, a grouping of flows based on their characteristics and a uniform sampling rate for each group might be a good solution. In this paper, we focus on finding an optimal sampling rate distribution among groups of flows. To the best of our knowledge, we propose a group-based sampling rate distribution for the first time. Our main contributions in this paper are the following:

1) We propose a flow grouping approach based on the behavioral similarity of VMs.
2) We study the relationship between the sampling rate and DDoS detection rate using an artificial neural network second order regression.
3) Using the relationship between the sampling rate and DDoS detection we find the sampling rate distribution among groups that produces the maximum detection rate.
4) We conduct experiments in a datacenter with 15 SDN switches and 36 servers with Hadoop and Spark.

The remainder of this paper is arranged as follows. In Section 2, we discuss some related works and their limitations. In Section 3, we present the network model and an overview of the monitoring system. Section 4 contains our proposed sampling rate distribution approach and a formal definition of the problem. In Section 5, we present some experimental results that strengthen our proposed solutions. Finally, Section 6 concludes our paper.

2 Related Work

There exist many works on DDoS detection methods and flow monitoring. Firstly, there are many statistical and machine learning methods, including correlation, entropy, divergence, cross-correlation, co-variance, and information gain that detect anomalous DDoS requests [3]. In [4], a DDoS detection mechanism is proposed based on Artificial Neural Networks. There are several other methods to detect DDoS attacks, including [5–8].

There are also many works on flow monitoring and packet sampling. In [9], an adaptive non-linear sampling method for passive measurement is proposed. The system dynamically sets the sampling rate for a flow depending on the number of packets. The sampling rate diminishes with the increase of packet count in a flow. Their approach samples a small flow with a large sampling rate and samples a large flow with a small sampling rate, both of which provide good accuracy. Several counter-based sampling techniques are proposed in [10]. The systematic count-based technique selects packets through a deterministic and invariable function based on the packet position. The systematic time-based technique selects packets based on arrival time with a deterministic interval. The random count-based technique selects the starting points of the sampling intervals randomly. Adaptive linear prediction [11] increases (or decreases) sampling rate in order to identify the new traffic pattern when the network activity increases (or decreases).

In [12], a multiadaptive sampling technique is proposed. In addition to adjusting the sampling rate, it adjusts the sample size. This approach avoids overload of the measurement points. When there is less activity, the multiadaptive technique decreases sampling rate and increases sample size to get more information about the network. In [13], the authors propose a distributed and collaborative monitoring system called DCM. DCM enables flow monitoring tasks and balances the measurement load at the switch. DCM can monitor different groups of flows using different actions. The system uses bloom filters to represent rules which use a small amount of memory in switch, but monitoring is very limited and cannot produce good accuracy. In [14], the authors use a deep neural network model to build an intrusion detection system. They use six basic features for SDN network without any sampling. In [15], the authors propose a flow monitoring algorithm to record some features of the DDoS attack traffic on the data plane.

We discussed three types of existing systems: (1) statistical approaches that analyze packets to detect and block

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2021.3054522, IEEE Transactions on Information Forensics and Security

TABLE 2: Table of notations

r_{ij}	Data rate of flow f_{ij}
s_{ij}	Sampling rate of flow f_{ij}
$dist(a,b)$	Distance (hops) between nodes a and b
$A(f_{ij})$	Node that monitors flow f_{ij}
$T(n)$	Type of node n (receiver, sender, or forwarder)
$r_i(n)$	Incoming data rate of n
$r_o(n)$	Outgoing data rate of n
θ	Node classification threshold
$\sigma[m,n]$	1st level similarity between nodes m and n
$\sigma'[m,n]$	2nd level similarity between nodes m and n
A	Adjacency matrix of flow graph
$D[m,n]$	Dissimilarity between nodes m and n
K	Number of groups
s_k	Sampling rate for $k-$th group
r_k	Total data rate for $k-$th group
r^k	Normalized data rate for $k-$th group
S	Maximum overall sampling rate
S_{min}	Minimum sampling rate for each group

attack traffic (2) dynamic sampling of packets based on traffic volume and other parameters, and (3) machine learning based systems for detecting attack packets without sampling the packets. Since, there are some differences among the attackers, victims, and other nodes in the way they communicate with each other or the master, we can use that for grouping their communications/flows. None of the existing works discussed above utilize grouping the flows based on characteristics of their source and destination. In addition, the detection rate of DDoS behavior depends on the sampling rate. The effect of the sampling rate on the detection rate also depends on the characteristics of the flows. Therefore, a group-based sampling rate distribution is necessary to increase the detection rate.

3 System Model

In this section, we describe the datacenter structure, attack model, and monitoring system. The overall system works in three steps. In the first step, the controller divides the flows into several groups. In the second step, packets in all flows in each group are analyzed to find the relationship between sampling rate and accuracy. Then, an optimization problem is solved to find an optimal sampling rate in each group. In the third step, the monitoring system optimally assigns each flow to a monitor. We summarize all the notations used in this papers in Table 2.

3.1 Datacenter Network Model

Our datacenter is composed of virtual machines (VM), physical machines (PM), top of rack (TOR) switches, SDN switches, users, attackers, and a victim. The victim uses DDoS protection service from a commercial DDoS protection service provider. The datacenter also has some security modules that are installed at the top layer of the network. Therefore, all the external DDoS attack traffics are caught by DDoS protection server or security modules.

3.2 Attack Model

We assume that the attackers reside in the same datacenter as the victim. Attackers are controlled by a master who may reside outside of the datacenter. The attacker is capable of launching SYN flood and UDP flood. The attackers cannot spoof IP addresses and launch an amplification attack. This is because, in most of the datacenters, the VMs are not allowed to spoof others' IP addresses and spoofed packets are blocked at the hypervisor level. Amplification attacks are not possible nowadays because, new versions of the operating systems do not reply to an ICMP/ping packet which is destined to a broadcast address. The attackers target the master of Hadoop or Spark system and launch a mix of SYN flood and UDP flood. The goal of the attacker is to increase the computation time or hamper the operation of the Hadoop or Spark.

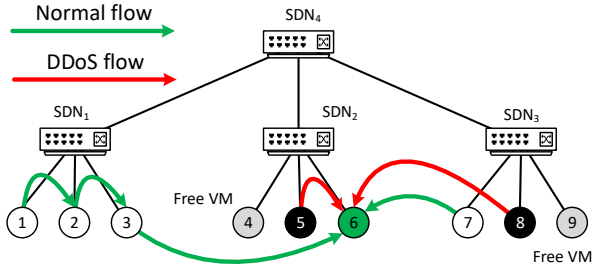


Fig. 2: Monitoring system.

3.3 Monitoring System Overview

Fig. 2 depicts the overall monitoring system. VM 6 is the victim and master. VMs 3 and 7 are the workers (e.g. data nodes). Attackers 5 and 8 send DDoS packets to the master 6. VMs 4 and 9 are unused VMs and they will be used as monitors. Let the capabilities of all VMs be the same and all flows have equal data rates. A VM can monitor only one flow. There are 6 flows in the datacenter. To monitor all the flows we need at least 6 monitors. We have only 2 monitors, so the system sub-samples the flows and copies 33% of the total packets probabilistically to the monitors. 33% of the packets can be copied in many different ways from the flows. However, the copied packets in each flow increase the network overhead. The network overhead for copying a flow is the data rate of the copied flow multiplied by the number of hops it travels. For example, if the flow $1 \rightarrow 2$ is monitored in VM 4, then network overhead is $r_{12} \times 0.33 \times 3$, where r_{12} is the data rate of the flow $1 \rightarrow 2$. $r_{12} \times 0.33$ is the data rate of the copied flow, and 3 is the number of hops that the copied packets travel. Flow $1 \rightarrow 2$ can be copied from SDN_1 and copied flow travels path $SDN_1 \rightarrow SDN_4 \rightarrow SDN_2 \rightarrow 4$. Therefore, the total increased network overhead for copying all flows is:

$$C = \sum_{f_{ij} \in F} r_{ij} s_{ij} \times \min_{p \in i \rightarrow j} dist(p, A(f_{ij})). \quad (1)$$

Here, F is the set of all flows, s_{ij} is the sampling rate of flow f_{ij}, and A is an assignment function that maps a flow to a monitor VM. The goal of the monitoring system is to find an assignment that produces the minimum network overhead.

The assignment process of the monitoring system consists of two steps. In the first step, from the flow graph $G = (V, E)$ it constructs a bipartite graph $G' = (V', E')$, where $V' = V_1 \cup V_2$, V_1 is the set of flows, and V_2 is the set of monitor VMs. From each flow, edges are added to all

flow VMs. The weight of an edge is the increased network overhead for assigning that VM to the monitor.

In the second step, it transforms the G' to a flow graph $G'' = (V'', E'')$. V'' contains all the vertices of V_1 and V_2, including a source S and destination D. Then, edges from S to all nodes in V_1 are added with capacities equal to the data rates of the corresponding flows. Edges are added from all nodes in V_2 to D having a capacity of the data rate the monitor can handle. The costs of new edges are zeros. The cost of other edges are set to the increased network overheads. Then it finds a max-flow min-cost paths and the edges with flow indicate the flow assignment to the monitors [16].

4 SAMPLING RATE DISTRIBUTION

In this section, we present the sampling rate distribution approach among the flows. We first divide the VMs into several groups. Then, we group the flows based on the VM groups. Finally, we assign sampling rates to each group.

4.1 Flow Grouping

We first group the nodes/VMs in the datacenter and then group the flows by the groups of nodes. We use second level behavioral similarity to classify the nodes. The uniqueness of second level behavioral similarity is that the behavior of neighbors determines the behavior of a node along with its own behavior. Therefore, two nodes that have similar type of nodes in their neighborhood are considered similar. When a group of bots attack, they send a lot of traffic to the victim. The victims also gets a lot of traffic from users. Therefore, if there are multiple victims in a network, they should be similar according to second level behavioral similarity because they will have a lot of traffic from separate groups of bots and users. On the other hand, the master will receive traffic from the bots but not from the users. Therefore, all of the masters in a network might be similar according to the second level behavioral similarity. Thus, traffics related to similar nodes are treated similarly which is not possible in any other type of grouping approaches including machine learning and other attribute based approaches [17].

We classify the nodes into three types: sender, receiver, and forwarder. A sender (or receiver) node's incoming data rate is much lower (or higher) than the outgoing data rates. A forwarder node has similar incoming and outgoing data rate. We classify the nodes into these three categories based on general behaviors of attackers and victims. Usually, attackers send a lot more traffic than they receive. A victim usually is overwhelmed by traffic and thus its incoming traffic is much higher than the other nodes. Forwarder nodes are usually neutral nodes which are neither attackers or victims. The types are considered as first level similarity and similarity among the neighbors is considered second level similarity. We use the second level similarity to group nodes because nodes that have similar types of neighbors are subject to suffer similarly by their neighbors. The type of a node n, $T(n)$ is defined as the following:

$$T(n) = \begin{cases} \text{receiver}, & \text{if } r_i(n)/r_o(n) > \theta, \\ \text{sender}, & \text{if } r_i(n)/r_o(n) < \frac{1}{\theta}, \\ \text{forwarder}, & \text{otherwise.} \end{cases} \quad (2)$$

Here, $r_i(n)$ and $r_o(n)$ are the incoming and outgoing data rates of node n. θ is a system-defined threshold. For example, if $\theta = 2$, then a node will be classified as a sender (or a receiver) if its outgoing data rate is higher than double (or less than half) of its incoming data rate. If the incoming data rate is less than double and higher than half of the outgoing data rate, then it is a forwarder node. The value of θ can be set based on statistics of the nodes in the network. It should not be close to 1 because a slight change in flow rates might change the type of the nodes. On the other hand, it should not be a large number. In that case, most of the nodes will be classifed as forwarder nodes.

Next, we define the first level similarity matrix σ. σ is a matrix containing similarity between every pair of nodes based on the type of nodes. $\sigma[m, n]$ contains the similarity between node m and n. If the types of nodes n and m are the same, then $\sigma[m, n]$ is equal to 1. If the types of nodes n and m are different, then $\sigma[m, n]$ is equal to 0 . σ can be expressed as the following:

$$\sigma[m, n] = \begin{cases} 1, & \text{if } T(m) = T(n), \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

Then, we calculate the second level similarity matrix σ'. We use regular equivalence to calculate σ'. The regular equivalence is widely used in social network analysis. In regular equivalence, two nodes are similar if they have many neighbors who are themselves similar. $\sigma[m, n]$ can be expressed as the following [18]:

$$\sigma'[m, n] = \alpha \sum_{kl} A[m, k]A[n, k]\sigma[k, l] + \delta[m, n]. \quad (4)$$

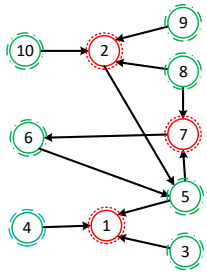
Here, A is the adjacency matrix of the flow graph. α is an eigenvalue of σ. $\delta[m, n]$ is 1 (or 0) if m and n are equal (or different). Equation 4 can be expressed as the following matrix multiplication form:

$$\sigma' = \alpha A\sigma A + I. \quad (5)$$

Fig. 3(a) shows an example of a flow graph with 10 nodes. A directed edge indicates a flow direction. Inspecting visually, we can say that nodes 1 and 2 look similar because they have incoming/outgoing flows from a similar type of node. They could be masters of a distributed application or victims of a DDoS attack. So, we should treat the incoming flows of these nodes the same way.

In this example, we assume the data rates are equal for all flows and θ is 2. For node 1, the numbers of incoming flows ($r_i(1)$) and outgoing ($r_0(1)$) flows are 3 and 0, respectively. $r_i(1)/r_o(1) = 3/0 = \infty$, which is greater than 2. Therefore, node 1 ($T(i)$) is a receiver. Similarly, node 2 is also a receiver. According to the first level similarity, nodes 1 and 2 are similar. The first level similarity cannot produce what we expect. If we look at node 10, it has 1 incoming flow and 0 outgoing flows. Therefore, node 10 is also similar to nodes 1 and 2, which is unexpected. Nodes $\{1, 2, 7\}$, $\{3, 4, 8, 9, 10\}$, and $\{5, 6\}$ are receivers, senders, and forwarders, respectively. Fig. 3(b) shows the similarity matrix σ. Therefore, the similarity among nodes 1, 2, and 7 is 1. Next, we calculate second level similarity among nodes 1, 2, and 7. The maximum eigenvalue of the Q is $\alpha = 2.75$. Using Equation 4, we calculate $\sigma'[1, 2]$. Sender neighbors of nodes 1 and 2 are $\{3, 4\}$ and $\{8, 9, 10\}$. Forwarder

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2021.3054522, IEEE Transactions on Information Forensics and Security



(a) Flow graph G.

i	1	2	3	4	5	6	7	8	9	10
1	1	1	0	0	0	0	0	0	0	0
2	1	1	0	0	0	0	0	0	0	0
3	0	0	1	1	0	0	1	0	1	1
4	0	0	1	1	0	0	1	0	1	1
5	0	0	0	0	1	1	0	1	0	0
6	0	0	0	0	1	1	0	1	0	0
7	0	0	1	1	0	0	1	0	1	1
8	0	0	0	0	1	1	0	1	0	0
9	0	0	1	1	0	0	1	0	1	1
10	0	0	1	1	0	0	1	0	1	1

(b) 1st level similarity matrix σ.

i	1	2	3	4	5	6	7	8	9	10
1	0	0	1	1	1	0	0	0	0	0
2	0	0	0	0	1	0	0	1	1	1
3	1	0	0	0	0	0	0	0	0	0
4	1	0	0	0	0	0	0	0	0	0
5	1	1	0	0	0	1	1	0	0	0
6	0	0	0	0	1	0	1	0	0	0
7	0	0	0	0	1	1	0	1	0	0
8	0	1	0	0	0	0	1	0	0	0
9	0	1	0	0	0	0	0	0	0	0
10	0	1	0	0	0	0	0	0	0	0

(c) Adjacency matrix A.

i	1	2	3	4	5	6	7	8	9	10
1	15	19	0	0	3	3	11	0	0	0
2	19	29	0	0	3	3	14	0	0	0
3	0	0	4	3	8	3	0	6	3	3
4	0	0	3	4	8	3	0	6	3	3
5	3	3	8	8	29	11	6	17	8	8
6	3	3	3	3	11	7	6	6	3	3
7	11	14	0	0	6	6	15	0	0	0
8	0	0	6	6	17	6	0	12	6	6
9	0	0	3	3	8	3	0	6	4	3
10	0	0	3	3	8	3	0	6	3	4

(d) 2nd level similarity matrix σ'.

Fig. 3: An example of flow grouping using second level behavioral.

neighbors of nodes 1 and 2 are $\{5\}$ and $\{5\}$. Therefore, $\sigma'[1,2] = 2.75 \times \{(2 \times 3) + (1 \times 1) + 0\} = 19.25$.

Similarly, we calculate that $\sigma'[1,7] = 11$ and $\sigma'[2,7] = 13.75$. So, we can see that nodes 1 and 2 are more similar than nodes 1 and 7 or 2 and 7. The similarity score between 1 and 3 is 0. This is because they are different in type and they do not have any similar neighbors. Using the matrix multiplication form, we calculate σ' (shown in Fig. 3(c), rounded to the nearest integer).

Then, we group the nodes using the hierarchical clustering algorithm. We formulate the distance matrix from the second level similarity matrix. The similarity values between a pair of nodes are subtracted from the maximum of the similarity values (except the similarity value of a node with itself) to get dissimilarities. The dissimilarities are used as distances for hierarchical clustering. The distance matrix is represented as follows:

$$D[m,n] = \begin{cases} \max\limits_{i \neq j} \{\sigma'[i,j]\} - \sigma'[m,n], & \text{if } m \neq n, \\ 0, & \text{otherwise.} \end{cases} \quad (6)$$

Next, we use D to cluster the nodes into K groups. There is trade off between the number of groups and the execution time. When the number of groups is higher, the system takes a long time to find the optimal sampling rate. Therefore, the system needs to adjust the value of K based on several parameters including the topology size, machine configuration, and performance of detection algorithm. Firstly, we use the hierarchical clustering algorithm. We set the maximum class to be the number of nodes. This clustering will group the most similar nodes in a cluster which will be the largest cluster. We pick that cluster and remove all the nodes in that cluster. Then, we repeat the process $K-1$ to find $K-1$ clusters. The rest of the nodes are grouped and treated as another cluster. The complete approach is shown in Algorithm 1. The procedure IN-FLOWS(c) returns the incoming flows of the nodes in set c. Details of the procedure are not shown due to limited space.

If we use $K = 2$, then we get groups $\{1,2,7\}$ and $\{3,4,5,6,8,9,10\}$. Therefore, the groups of flows are $\{3 \to 1, 4 \to 1, 5 \to 1, 8 \to 2, 9 \to 2, 10 \to 2, 5 \to 7, 8 \to 7\}$ and $\{2 \to 5, 7 \to 6, 6 \to 5\}$. If we use $K = 3$, then we get groups $\{1,2,7\}$, $\{4\}$, and $\{3,5,6,8,9,10\}$. Therefore, the groups of flows are still the same because there are no incoming flows to node 4.

After grouping flows, we need to find the relationship between the DDoS detection rate and the sampling rate using an artificial neural network (ANN).

Algorithm 1 Flow grouping

Input: The adjacency matrix A,set of flows F, and number of clusters K.
Output: A group of flows.

Procedure: FLOW-GROUPING(A, F, k)
  Calculate types $T[n]$ from flows for each node n.
  Calculate σ from the types T.
  $\alpha \leftarrow$ the maximum Eigen value of σ.
  $\sigma' = \alpha A \times \sigma \times A + I$.
  Calculate D from σ' using Equation 6.
  return GROUP(D, k)
Procedure: GROUP(D, k)
  $g \leftarrow \emptyset$
  for $i = 1$ to $K - 1$ **do**
    $C \leftarrow$ clusters using hierarchical clustering
    $c \leftarrow$ largest cluster in C
    Remove all nodes in c from D.
    $g \leftarrow g \cup$ IN-FLOWS(c)
  $r \leftarrow$ set of nodes that are not in any cluster
  $g \leftarrow g \cup$ IN-FLOWS(r)
  return g.

4.2 Relationship between Detection Rate and Sampling Rate

To find the relationship between the sampling rate and the detection rate, we need to vary the sampling rate and record the detection rate for each flow group. We use a pre-trained model of ANN. The input of the ANN consists of the features of a certain number of packets. The packets are grouped based on the order they are received. The ANN uses two types of features: features of an individual packet and features of packets in a group. Features of an individual packet include the following:

- Protocol: The protocol of the packet. A protocol of a packet can be TCP, UDP, HTTP, or ICMP.
- Packet Size: The total size of the packet in bytes.
- Entropy: The entropy is calculated over all bytes in a packet using the Tsallis method.

Features of a group of packets are the following:

- Variance in packet size: The variance of the packet size (in milliseconds) difference between two consecutive packets.
- Entropy: The entropy is calculated over all bytes in all packets in a group using the Tsallis method.
- Variance in arrival time: The variance of the time (in milliseconds) difference between two consecutive packets.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2021.3054522, IEEE Transactions on Information Forensics and Security

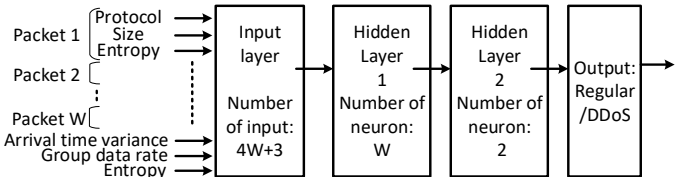


Fig. 4: Structure of the ANN.

Though there is a protocol number for each protocol, we use an indexing method to provide a numeric value for each protocol. In our approach, DNS, TCP, UDP, ICMP, HTTP, DHCP, and SSH protocols are considered. All other protocols are considered to be the same. To calculate the entropy of a packet we use the Tsallis entropy formula [19]. Let there be N bytes in a received packet p. Then, the entropy of the packet p is:

$$E(p) = \frac{1}{q-1}\left\{1 - \sum_{n=0}^{N-1} P(p[n])^q\right\}. \tag{7}$$

Here, P is the probability density function of the bytes of packet p. $p[n]$ denotes the n th byte of p. q is a real parameter which is called the entropic-index.

We split the packets in each flow by a 60-40 ratio. We keep the first 60% of the packets in each flow as the training packets and the rest as the test packets. Then from each flow, we divide the packets into non-overlapping groups based on arrival time. Each group contains w consecutive packets. Then the four features of all packets are organized in a specific order. Then, the three group features are appended. For example, if we have $w = 5$, then the total number of inputs of the ANN is $5 \times 4 + 3 = 23$. The class of each input is either regular or DDoS. The packets in the flows from the attackers to the victim are considered as DDoS packets. Other packets are considered as regular packets.

From the test packets, we pick a portion of the packets from a group. From each group, packets are picked with different probabilities. For example, if we have two $(w = 2)$ groups of flows and from each group, we pick a packet with 10 different probabilities, then there will be $10 \times 2 = 20$ different test packet groups. From each test group, we formulate a test dataset. Using the retrained model, we calculate the DDoS detection rate of each test dataset.

The details of the ANN structure are shown in Fig. 4. We have $4w + 3$ inputs in the input layer. We have two hidden layers with w and 2 neurons. We use the most commonly used activation function, ReLU, as the activation function in our ANN. This structure works well to differentiate between DDoS and regular flows. This two-hidden layered ANN works much better than a one-hidden layered ANN. The relationship depends greatly on the machine learning model. If the model is very good and can detect the attack with even 1% of the sampling rate, then all of the relationships will be straight lines parallel to the X-axis. In that case, any sampling rate is good. In practical it is impossible to find a perfect model because of the dynamic attack behavior. If the model is changed, then it is required to retrain using the train dataset. In this paper, we are not focusing on finding the ANN structure that performs best. We assume the machine learning model is not perfect. We are focusing on finding out how the sampling rate affects the DDoS detection rate.

After getting the DDoS detection rates of each test dataset, we apply second-order polynomial regression to find the relationship between the sampling rate and the DDoS detection rate for each group.

Let there be K number of groups. We denote the relationship between the sampling rate and DDoS detection rate for a group k as $f_k(s)$. Here, $k = 0, 1, 2, ...K-1$ is the group number and s is the sampling rate. The total data rate of all flows in group k is r_k. The problem can be expressed as the following:

$$\begin{aligned} \text{maximize:} \quad & \sum_{k=0}^{K-1} f_k(s_k) \times r_k \\ \text{subject to:} \quad & \sum_{k=0}^{K-1} s_k \times \hat{r_k} \leq S, \quad \forall_k s_k > S_{min} \end{aligned} \tag{8}$$

Here, $\hat{r_k}$ is the normalized data rate of group k. S is the maximum overall sampling rate that the system can process. S_{min} is the minimum sampling rate that produces good accuracy. We consider the sampling rates as integer-valued and the problem as a mixed-integer program. The problem can be solved using any optimization problem solver.

5 Experiments

5.1 Experimental settings

We first present the datacenter structure. Fig. 5 shows the datacenter structure. The datacenter is composed of 35 servers, 15 SDN switches, and some regular L2 switches. All the servers except the gateway are Dell PowerEdge 210 (2 cores 2.4 GHz processor, 4 GB RAM, 500 GB HDD) servers. Each server has at least two gigabit Ethernet ports. The SDN switches are Pica8 p-3922.

We set up two networks: a control network and a data network. In the control network, all management ports of SDN switches and the SDN controller (gateway) are connected through an L2 switch. SDN switches are configured as the out-of-band controller, which means the control plane and the data plane are separated. The dotted lines in Fig. 5 show the control network. Therefore, our control network is a star topology. In the data network, the data ports of the SDN switch and the gateway are connected. The topology is a three-level complete binary tree topology. The gateway is connected with the root SDN switch and other servers are connected to leaf SDN switches. We use OpenDaylight [20] as the SDN controller, which is installed in the Gateway. For flow rule generation, we use the L2Switch plugin with OpenDaylight. Next, we set up a Hadoop cluster on server 34 and 16 other servers (all even numbers). Server 34 is configured as the namenode and even-numbered servers are configured as data nodes. We configure the Hadoop file system to have three replications of files and set yarn as the resource manager.

Then, we generate some text files for input to a MapReduce program. The text files are generated by taking a word randomly from a dictionary. In our experiments, we generated 100 text files each having a size of about 10 MB. Therefore, the total input is 1 GB of text files that are uploaded to the Hadoop file system. We run the WordCount or WordRank program on the input files to generate normal

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2021.3054522, IEEE Transactions on Information Forensics and Security

TABLE 3: Groups of nodes for Flows I using Algorithm 1

	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6
K=2	12, 33, 34	rest of the nodes				
K=4	12, 33, 34	16, 22, 26	2, 4, 20	rest of the nodes		
K=6	12, 33, 34	16, 22, 26	2, 4, 20	6, 8, 10, 14, 18, 24, 28, 30, 32	0, 1, 3	rest of the nodes
# of packets	793,340	476,068, 2,139,044 (for K=2)	314,351	1,008,455, 1,348,625 (for K=4)	60,142	280,028

TABLE 4: Groups of nodes for Flows II using Algorithm 1

	Group 1	Group 2	Group 3	Group 4	Group 5	Group 6
K=2	26, 33, 34	rest of the nodes				
K=4	26, 33, 34	12, 14, 20	2, 20, 26	rest of the nodes		
K=6	26, 33, 34	12, 14, 20	2, 20, 26	10, 16, 18, 22	6, 8, 24, 28	rest of the nodes
# of packets	649,829	384,961, 2,282,555 (for K=2)	597,192	495,014, 1,300,402(for K=4)	465,218	340,170

traffic. The MapReduce framework first splits the files by a new line and sends them to worker nodes for processing. This step creates much internal traffic. After processing parts of the data, the workers send their results to multiple workers depending on the keys of the result. This process also creates a lot of internal traffic. The final output from the reducer is also stored in the Hadoop file system. Because of three replication factor, writing to a file also produces much network traffic.

Then, we install the attacker programs in the 16 (odd numbered) servers. The attack master 33 runs a web service and the attacker program gets commands from it. The attacker also replies with the response to the attack master. The commands are usually Linux commands. We install popular attacking applications such as hping3 [21] and packETH [22]. Using hping3, we can launch SYN flood, UDP flood, and malformed packet attacks with different data rates. hping3 is not capable of randomizing the sending interval and packet size. With packETH, we can launch attacks with random sending interval and size. The attacks are targeted against the Hadoop master 34. The Hadoop master program listens on port 9000. Therefore, the attackers send the packets targeting port 9000.

We simultaneously run several WordCount programs and a mixed type of attack and capture incoming packets from the 34 servers (except the gateway) using tshark [23]. The packet capture files contain packets of all flows for 10 mins. The files contains about 30 million packets on average. We classify the flows from odd number servers to server 34 as DDoS flows. All other flows, including the flows due to the communication of attackers with the attack master, are considered as regular flows. Then, we train our ANN with the first 60% of the data (18 million packets). The rest of the data is grouped using our proposed grouping approach and relationship functions with the sampling rate are calculated. Finally, by solving an optimization problem, we find our desired sampling rate. We compare the uniform sampling rate with different numbers of groupings. We denote the flows with WordCount run as Flows I. We change the destination of the attack flows randomly and create another set of flows which is denoted by Flows II.

5.2 Experiment results

Firstly, we show a small scale DDoS attack targeting the Hadoop master (server-34). We start with one attacker and increase the number of attackers to sixteen. We run the

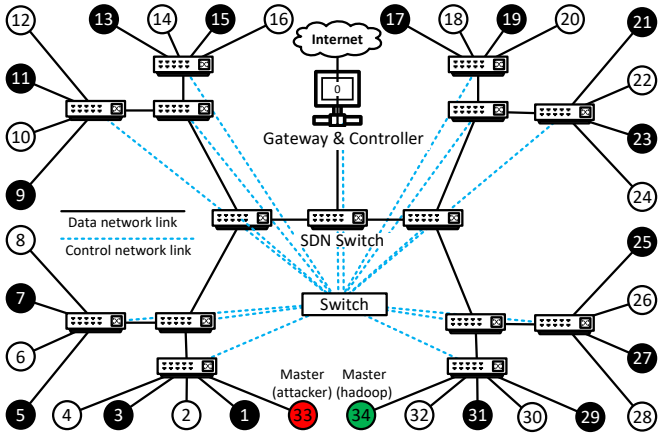
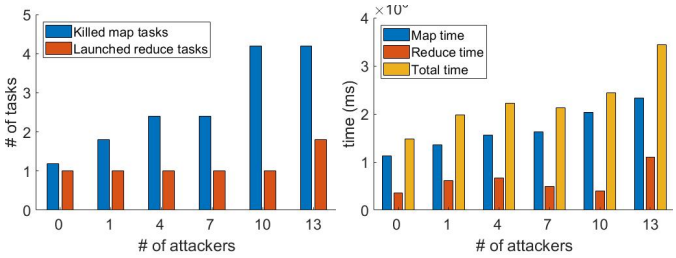


Fig. 5: Datacenter topology.



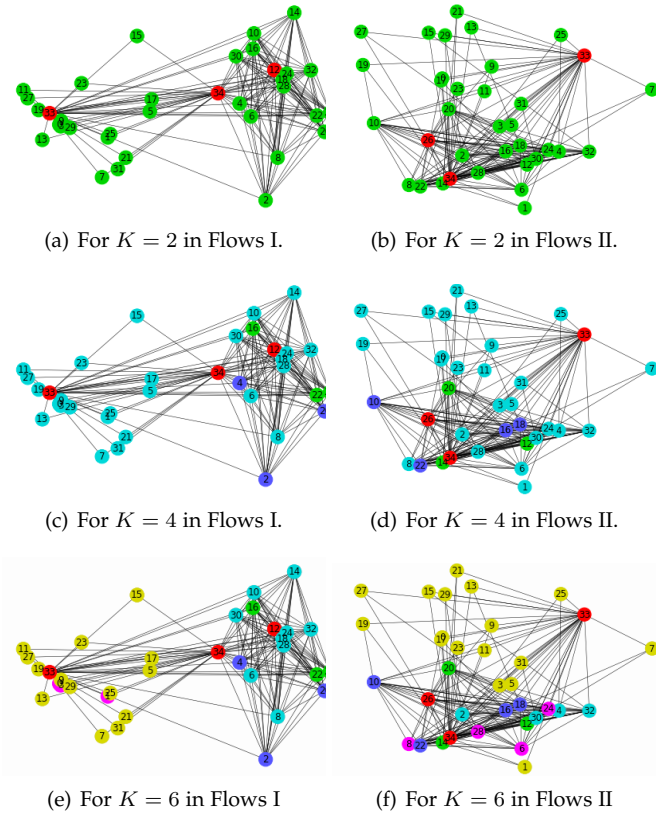
(a) Number of tasks vs. attackers. (b) MapReduce time vs. attackers.

Fig. 6: Effect of small scale SYN flood attack.

MapReduce WordCount program on the generated 1 GB text files. Fig. 6 shows the effect of a SYN flood attack. We vary the number of attackers from 0 to 13 in 5 steps. The attack is produced using the hping3 tool. We keep the interval between two SYN packets as 1 millisecond (ms). Each packet is 64 bytes so that the data rate is 512 Kbps. All the measurements are averaged over 5 runs of the WordCount program.

Fig. 6(a) shows the number of killed map tasks and launched reduce tasks with different numbers of attackers. When the number of attackers increases, the number of killed map tasks increases. When there is no attack, the number of launched map tasks is 101, and among them 1 task is killed. When the number of attackers is between 10 and 13, the number of launched map tasks is between 105 and 108, and among them the number of killed tasks is between 2 and 6. When the number of attackers is between 0 and 10, the number of launched reduce tasks is 1. When the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2021.3054522, IEEE Transactions on Information Forensics and Security

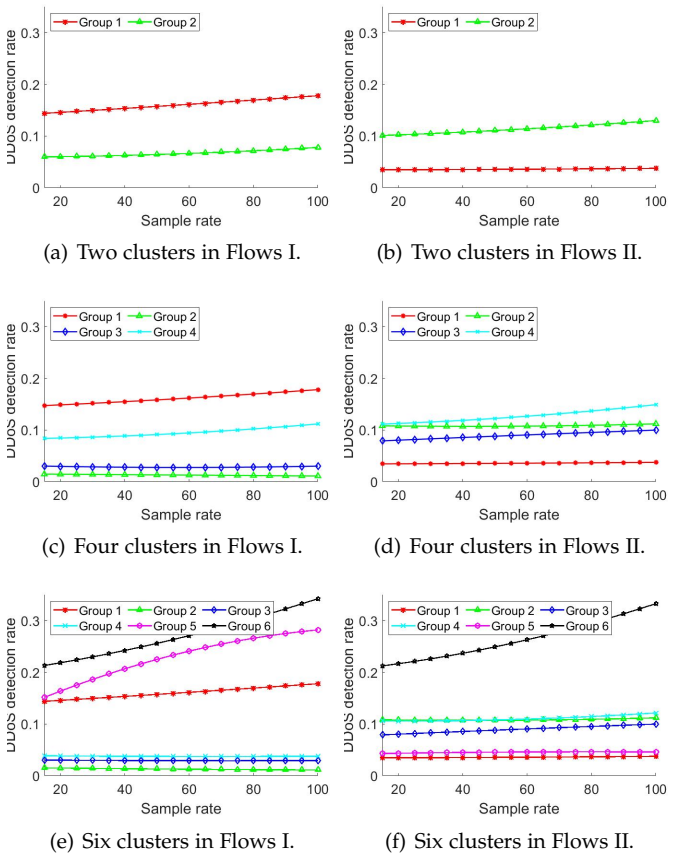


(a) For $K = 2$ in Flows I. (b) For $K = 2$ in Flows II.

(c) For $K = 4$ in Flows I. (d) For $K = 4$ in Flows II.

(e) For $K = 6$ in Flows I (f) For $K = 6$ in Flows II

Fig. 7: Grouping of nodes based on second level similarity.



(a) Two clusters in Flows I. (b) Two clusters in Flows II.

(c) Four clusters in Flows I. (d) Four clusters in Flows II.

(e) Six clusters in Flows I. (f) Six clusters in Flows II.

Fig. 8: Sampling rate vs. DDoS detection rate.

number of attackers is 13, the number of launched reduce tasks is between 1 and 4 (1.8 on average).

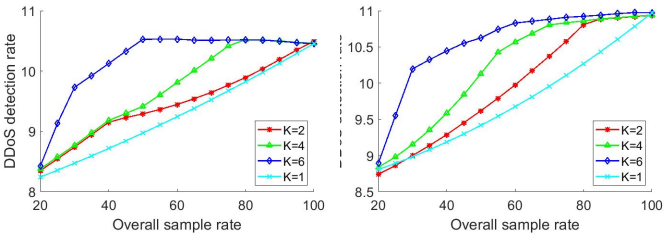
Fig. 6(b) compares the time spent on map and reduce tasks with different numbers of attackers. When there is no attack, the total time spent on map tasks is $1,122,082$ ms. When the number of attackers is 13, the total time spent on map tasks is $2,334,606$ ms. So, the map time is more than double than that of with no attack. The reduce time without attack is $360,243$ ms. Though the reduce time increases and then decreases when the number of attackers is between 1 and 10, it is normal. This is because the number of launched reduce tasks is 1 for 1 to 10 attackers. The variation of reduce time is due to the processing time of the data. When the number of attackers is 13, the number of launched reduce tasks is 2 and the reduce time is $1,106,689$ ms, which is three times greater than that of with no attack. Therefore, due to 512 Kbps SYN flood attack, the map task is affected more than the reduce task. This is because there are more map tasks than reduce tasks. Due to the attack, the probability of communication failure is higher during the map period than during the reduce period. The MapReduce job never fails when the number of attackers is between 0 to 10. When the master is attacked by 13 attackers, the MapReduce job fails almost 50% of the time. When a MapReduce job fails in hadoop hosted by commercial service providers, it is least likely to fail because of back-up VMs which start on a failure. When this type of incident happens, the time to finish the job increases dramatically.

Fig. 7 shows the grouping of the nodes according to Algorithm 1. The groups of nodes are summarized in Tables 3 and 4. In Flows I, we observe that for $K = 2$, the attack master 33 and Hadoop master 34 are in the same group, which is expected. The other worker nodes except 12 and attackers belong to another group. Similar behavior is observed in other values of K. In Flows II, we also observe similar behavior for different values of K. Because of the large number of flows (total of 371 flows in the dataset), groups of flows are not shown here.

Next, we find the relationship between the sampling rate and the DDoS detection rate for each group. We train the ANN that we presented earlier with the first 60% of the data from each flow. Our ANN shows an accuracy of 93.38% with false positive and false negative rates of 0.8% and 5.79%, respectively. We keep $w = 10$ for the ANN. We vary the sampling rate from 20% to 100%. The DDoS detection rate for a group is the number of inputs that are classified as DDoS divided by the total number of inputs in that group. Using the DDoS detection rates and sampling rates, we find the relationship functions by second order polynomial regression.

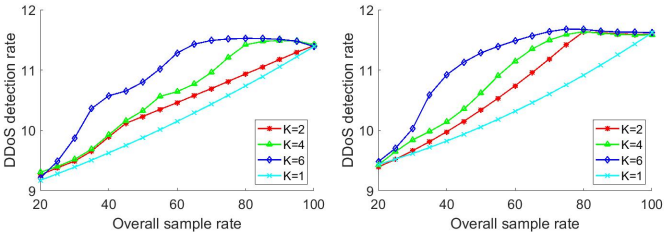
Fig. 8 plots the relationship functions for different groups for different K values. Fig. 8(a) plots the relationship functions for $K = 2$ using Flows I. From the settings we know that all the attack flows belong to Group 1. Therefore, Group 1 shows the highest DDoS behavior. Group 2 also shows some DDoS behavior, which is false positive. Besides, some normal flows can also have similar behavior (entropy, variance of arrival time, etc.) to DDoS flows. It is impossible to build a model that detects DDoS with 100% accuracy. For $K = 2$ and $K = 4$, other groups (except Group 1) show DDoS behavior for the same reason. Most of the groups' (Groups 1 and 2 for $K = 2$, Groups 1 and 4 for $K = 4$, and Groups 1, 5, and 6 for $K = 6$) DDoS detection rates

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2021.3054522, IEEE Transactions on Information Forensics and Security



(a) Using Flows I. (b) Using Flows II.

Fig. 9: Comparison between uniform and grouped distribution.



(a) Attacks to the Spark master. (b) Attacks to random workers.

Fig. 10: Uniform and grouped distribution in Spark.

TABLE 5: sampling rates for different K for Flows I.

Sample rate	Group sampling rates		
	K=2	K=4	K=6
20	23, 20	23, 20, 20, 20	20, 20, 20, 20, 68, 20
40	97, 20	97, 20, 20, 20	63, 20, 20, 20, 100, 100
50	100, 32	100, 20, 20, 40	100, 20, 20, 20, 100, 100
60	100, 46	35, 20, 20, 100	100, 21, 100, 21, 100, 100
80	99, 74	100, 20, 35, 100	100, 20, 35, 100, 100, 100
100	100, 100	100, 100, 100, 100	100, 100, 100, 100, 100, 100

increase with the sampling rate. These flows are regular, having some similar characteristics to DDoS flows. Other groups' DDoS detection rates decrease very slightly with the sampling rate. This is because more packets from a flow help the ANN find more accurate classification. These flows are regular and some packets are classified as DDoS packets by ANN. Fig. 8(b) shows the plot of the relationship functions for $K = 2$ using Flows II. As the destination of attack flows is no longer the server 34 in Flows II, Group 1 in Flows I has less DDoS behavior than Group 1 in Flows II.

Finally, we formulate the problem using the relationship functions in each group. We keep the minimum sampling rate at 20%. This is because a sampling rate less than 20% does not produce good accuracy. Using the CVXPY optimization library in python, we solve the problem. Part of the solution is summarized in Tables 5 and 6. We keep all the sampling rates as integers. Therefore, the overall sampling rate 20 refers to the sampling rate greater than or equal to 20 but less than 21. The groups that have higher DDoS detection rates are assigned higher sampling rates. For example, in Flows I when $K = 2$, to achieve an overall sampling rate of 50%, our approach assigns 100% and 32% to the flows in Group 1 and Group 2, respectively. This kind of assignment produces a better overall detection rate.

We compare the performances of the grouping approach with those of the uniform (no grouping) sampling approach. Fig. 9 shows the comparison between our flow grouping approach and the non-grouping approach. When $K = 1$, all the nodes belong to a group and the sampling rate is

TABLE 6: sampling rates for different K for Flows II.

Sample rate	Group sampling rates		
	K=2	K=4	K=6
20	23, 21	20, 20, 20, 22	20, 20, 20, 20, 68, 20
40	23, 46	20, 20, 20, 22	63, 20, 20, 20, 100, 100
50	22, 59	20, 20, 20, 67	100, 20, 20, 20, 100, 100
60	22, 72	20, 20, 21, 89	100, 21, 100, 21, 100, 100
80	21, 98	20, 90, 100, 100	100, 20, 35, 100, 100, 100
100	100, 100	100, 100, 100, 100	100, 100, 100, 100, 100, 100

distributed uniformly over all flows. When $S = 100\%$, the overall DDoS detection is the same for all K values. In this case, all the groups are assigned to a 100% sampling rate, which is uniform. In other cases, the higher the values of K, the higher the DDoS detection rate. This is because the nature of the flows in each group is different. Detection of DDoS behavior in some of the flows is more sensitive to the sampling rate. At some points, the overall DDoS detection rate is higher than the 100% overall sampling rate. For example, in Flows I for $K = 6$, the overall DDoS detection rate is 10.54%, which is higher than the detection rate at 100% sampling rate (10.45). If we look at the sampling rate distribution, we see that Groups 2, 3, and 4 are assigned the lowest sampling rates, which produce some false positives and the total DDoS detection rate becomes higher. We observe similar behavior in Flows II.

We also conduct experiments by running WordCount in Spark. We set up Spark master on server 34. Workers are set up on all other even number servers (server-2, server-4,.... server-32). We run the WordCount program, which comes with the documentation of Spark. Spark execution is more network intensive and WordCount takes longer than MapReduce WordCount. Fig. 10 shows the comparison between our flow grouping approach and the uniform grouping approach in Spark. We observe similar behavior to Hadoop in Spark.

We compare our proposed system with the uniform sampling distribution and k-means grouping approaches under the same settings. It is hard for our work to have the same settings and parameters as the existing works. If we change some of the parameters or settings, then the originality of them is hampered. Fig. 11 shows the comparison between the proposed flow grouping approach and the *k*-means clustering approach. We use Flows I for this experiment. We use in-degree and out-degree as features of the nodes. By using *k*-means clustering, we divide the nodes into K number of clusters. Then, incoming flows of each cluster are grouped together. We observe that the *k*-means grouping produces better DDoS detection rates than the proposed approach for $K = 2$ and 4. The *k*-means clustering cannot divide nodes into more than 4 clusters. Our proposed approach with $K = 6$ outperforms the *k*-means with $K = 4$.

The proposed group-based approach is always better than the uniform sampling rate distribution for different MapReduce applications and attack strategies. The grouping approach does not produce an optimal grouping of flows. Therefore, we can conclude that our sampling rate distribution approach works better than the uniform sampling rate distribution. The behavioral similarity-based approach can produce better grouping of the flows than the in-degree and out-degree based grouping approach when the number of groups is large.

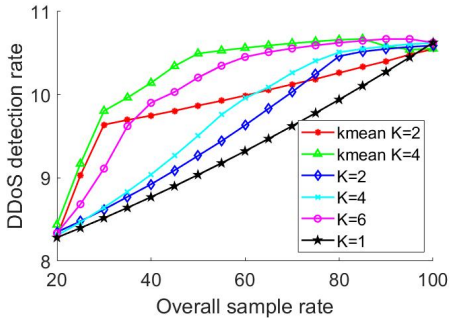


Fig. 11: Comparison with *k*-means clustering.

6 Conclusion

The internal DDoS is less common than the external DDoS attack. Compared to the external DDoS attack, it is also harder to detect and protect against. We can use the functionalities of SDN switches to monitor internal network flows. In this work, we present a framework for flow monitoring in a datacenter. We present a flow grouping approach based on behavior similarities among the virtual machines. We formulate an optimization problem for sampling rate distribution and solve the problem with an optimization solver. We conduct all the experiments in our datacenter and compare performances with different granularity levels including the uniform sampling rate. Our experiments show that increasing the number of groups produces better detection rates than the uniform packet sampling approach.

Acknowledgments

This research was supported in part by NSF grants CNS 1824440, CNS 1828363, CNS 1757533, CNS 1618398, CNS 1651947, and CNS 1564128.

References

[1] V. Corey, C. Peterman, S. Shearin, M. S. Greenberg, and J. Van Bokkelen, "Network forensics analysis," *IEEE Internet Computing*, vol. 6, no. 6, Nov 2002.

[2] R. Biswas and J. Wu, "Filter assignment policy against distributed denial-of-service attack," in *2018 IEEE 24th International Conference on Parallel and Distributed Systems*, Dec 2018.

[3] J. Wang and I. C. Paschalidis, "Statistical Traffic Anomaly Detection in Time-Varying Communication Networks," *IEEE Transactions on Control of Network Systems*, vol. 2, no. 2, Jun 2015.

[4] T. A. Ahanger, "An effective approach of detecting DDoS using Artificial Neural Networks," in *2017 International Conference on Wireless Communications, Signal Processing and Networking*, Mar 2017.

[5] X. Ma and Y. Chen, "DDoS Detection Method Based on Chaos Analysis of Network Traffic Entropy," *IEEE Communications Letters*, vol. 18, no. 1, Jan 2014.

[6] Z. Tan, A. Jamdagni, X. He, P. Nanda, and R. P. Liu, "A System for Denial-of-Service Attack Detection Based on Multivariate Correlation Analysis," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 2, Feb 2014.

[7] B. K. Joshi, N. Joshi, and M. C. Joshi, "Early Detection of Distributed Denial of Service Attack in Era of Software-Defined Network," in *2018 Eleventh International Conference on Contemporary Computing*, Aug 2018.

[8] A. Procopiou, N. Komninos, and C. Douligeris, "Forchaos: Real time application ddos detection using forecasting and chaos theory in smart home iot network," *Wireless Communications and Mobile Computing*, vol. 2019, 2019.

[9] C. Hu, S. Wang, J. Tian, B. Liu, Y. Cheng, and Y. Chen, "Accurate and efficient traffic monitoring using adaptive non-linear sampling method," in *The 27th Conference on Computer Communications*, Apr 2008.

[10] T. Zseby, M. Molina, N. Duffield, S. Niccolini, and F. Raspall, "Sampling and filtering techniques for ip packet selection", rfc 5475," 2009.

[11] E. A. Hernandez, M. C. Chidester, and A. D. George, "Adaptive sampling for network management," *Journal of Network and Systems Management*, vol. 9, no. 4, Dec 2001.

[12] J. M. Silva, P. Carvalho, and S. Rito Lima, "A multiadaptive sampling technique for cost-effective network measurements," *Computer Networks*, vol. 57, p. 3357–3369, 12 2013.

[13] Y. Yu, C. Qian, and X. Li, "Distributed and collaborative traffic monitoring in software defined networks," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*. New York, NY, USA: ACM, Aug 2014.

[14] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *2016 International Conference on Wireless Networks and Mobile Communications*, Oct 2016.

[15] X. Yang, B. Han, Z. Sun, and J. Huang, "Sdn-based ddos attack detection with cross-plane collaboration and lightweight flow monitoring," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec 2017.

[16] R. Biswas, J. Wu, and Y. Chen, "Optimal monitor placement policy against distributed denial-of-service attack in datacenter," in *Resilience Week*, Nov 2019.

[17] M. A. S. Saber, M. Ghorbani, A. Bayati, K. Nguyen, and M. Cheriet, "Online data center traffic classification based on inter-flow correlations," *IEEE Access*, vol. 8, pp. 60 401–60 416, 2020.

[18] M. Newman, *Networks: an introduction*. Oxford university press, 2010.

[19] C. Tsallis, "Possible generalization of boltzmann-gibbs statistics," *Journal of Statistical Physics*, vol. 52, no. 1, Jul 1988.

[20] "OpenDaylight," https://www.opendaylight.org.

[21] "hping3," https://linux.die.net/man/8/hping3.

[22] "packETH," https://github.com/jemcek/packETH.

[23] "tshark," https://www.wireshark.org/docs/man-pages/tshark.html.

Rajorshi Biswas is a PhD student of Computer and Information Sciences at Temple University, Philadelphia. He achieved his bachelor's degree from Bangladesh University of Engineering and Technology, Bangladesh. He is currently doing research at the Center for Networked Computing (CNC) which is focused on network technology and its applications. His research areas include Wireless Networks, Wireless Sensor Networks, Wireless Security, Cryptography, Cognitive Radio Networks etc. He published his research in many conferences and journals including IEEE ICPADS 2018, IEEE GLOBECOM 2019, IEEE ICC 2019, IEEE Sarnoff 2019, and Resilience Week 2019.

Jie Wu is the Director of the Center for Networked Computing and Laura H. Carnell professor at Temple University. He also serves as the Director of International Affairs at College of Science and Technology. He served as Chair of Department of Computer and Information Sciences from the summer of 2009 to the summer of 2016 and Associate Vice Provost for International Affairs from the fall of 2015 to the summer of 2017. Prior to joining Temple University, he was a program director at the National Science Foundation and was a distinguished professor at Florida Atlantic University. His current research interests include mobile computing and wireless networks, cloud and green computing, network trust and security, and social network applications. Dr. Wu regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including IEEE Transactions on Mobile Computing, IEEE Transactions on Service Computing, and Journal of Computer Science and Technology. Dr. Wu was general co-chair for IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, ACM MobiHoc 2014, ICPP 2016, and IEEE CNS 2016, as well as program cochair for IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). Dr. Wu is a Fellow of the AAAS and a Fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.

Sungi Kim is an undergraduate student at Temple University, Philadelphia. She will be receiving her B.S. in Computer and Information Sciences in 2020. Her undergraduate research with Dr. Jie Wu focused on developing a forensics toolkit for an embedded system. Her research interests include Network Security and Forensics. She worked as a STEM tutor at Temple University Student Success Center for two years. As a computer scientist, she strives to be a life-long learner and wants to improve people's quality of life through the use of technology.