# Unsupervised online anomaly detection in Software Defined Network environments

Gustavo Frigo Scaranti [a], Luiz Fernando Carvalho [b], Sylvio Barbon Junior [a], Jaime Lloret [c], Mario Lemes Proença Jr. [a,*]

[a] Computer Science Department, State University of Londrina, Londrina, Paraná 86057-970, Brazil
[b] Computer Engineering Department, Federal University of Technology – Paraná (UTFPR), Apucarana 86812-460, Brazil
[c] Integrated Management Coastal Research Institute, Universitat Politecnica de Valencia, 46022 Valencia, Spain

ARTICLE INFO

*Keywords:*
Anomaly detection
Software Defined Networking (SDN)
Stream mining
DenStream
DDoS
Portscan

ABSTRACT

Software Defined Networking (SDN) simplifies network management and significantly reduces operational costs. SDN removes the control plane from forwarding devices (e.g., routers and switches) and centralizes this plane in a controller, enabling the management of the network forwarding decisions by programming the control plane with a high-level language. However, its centralized architecture may be compromised by flooding attacks, such as Distributed Denial of Service (DDoS) and portscan. Facing this challenge, we propose an Intrusion Detection System (IDS) based on online clustering to detect attacks in an evolving SDN network taking advantage of the entropy of source and destination IP addresses and ports. Our proposal is focused on avoiding the demand for labeling and previous knowledge to provide a practical and accurate method to address real-life online scenarios. Moreover, our proposal paves the way for a comprehensive analysis by projecting the cluster's structure over the feature space, providing insights on intensity, seasonality, and attack type. Our experiments were carried out with the DenStream algorithm in several databases attacked by DDoS and portscan with different intensities, durations, and overlapping patterns. When comparing DenStream performance to Half-Space-Trees, an accurate online one-class classification algorithm for anomaly detection, it was possible to expose the capacity of our unsupervised proposal, overcoming the one-class solution, and reaching f-measure rates above 99.60%.

## 1. Introduction

In the last decade, computer networks have been expanded dramatically in terms of usage and complexity to support emerging technologies. New architectures and devices have become more common, such as cloud computing, virtualization, and the Internet of Things (IoT). Along with the widespread adoption of emerging technologies, new security issues have also emerged, forcing these technologies to tackle new forms of exploitation to provide reliable and resilient network environments (Aldribi, Traoré, Moa, & Nwamuo, 2020; Gamage & Samarabandu, 2020).

Network anomalies can occur from hardware or software issues, a hardware fault may overwhelm critical devices affecting the network functioning, and a misconfiguration of network resources opens a gap for vulnerability exploitation (Proença, Zarpelão, & Mendes, 2005). Intruders can take advantage by exploiting all these possibilities, such as manufacturing backdoors for malware or other penetrative purposes, hardware tampering with invasive operations, and launching zero-day attacks. In this manner, security threats remain a challenge for network managers, even more with the vast amounts of sensitive information transmitted through resource-constrained devices and over the Internet without any encryption, using heterogeneous technologies and communication protocols (Aldweesh, Derhab, & Emam, 2020; Liu, Quan, Cheng, Zhang, & Yu, 2019).

Detecting anomalous behavior in networks with constant changes in their temporal data has been gaining prominence among the industry and the scientific community (Calikus, Nowaczyk, Sant'Anna, & Dikmen, 2020; Kopp, Pevnỳ, & Holeňa, 2020). This kind of detection is a fundamental requirement for providing prompt alert and suitable problem mitigation towards supporting available, reliable, and resilient network services (Sharma, Pilli, Mazumdar, & Gera, 2020; Thakkar & Lohiya, 2020; Yamansavascilar, Baktir, Ozgovde, & Ersoy, 2020). In this context, Software Defined Networking (SDN) has been used to

* Corresponding author.
*E-mail addresses:* gustavo.scaranti@uel.br (G.F. Scaranti), luizfcarvalho@utfpr.edu.br (L.F. Carvalho), barbon@uel.br (S. Barbon Junior), jlloret@dcom.upv.es (J. Lloret), proenca@uel.br (M.L. Proença Jr.).

Accepted 11 November 2021

create a dynamic, flexible, and autonomous environment with secure mechanisms to protect network assets, switches, routers, servers, and other devices (De Assis et al., 2018; Sahay, Meng, & Jensen, 2019).

Due to its intrinsic characteristics, SDN has become compelling for managing Local Area Networks (LANs) infrastructures, as well as converged and hyper-converged data centers (Barakabitze, Ahmad, Mijumbi, & Hines, 2020; Yurekten & Demirci, 2021). SDN centralizes the network management into a programmable controller, decoupling the data and control planes and communicating with the network devices to instruct them on handling the traffic. However, the centralized design provided by the SDN is a convenient target for Distributed Denial of Service (DDoS) attacks (Carvalho, Abrão, de Souza Mendes, & Proença, 2018; Correa Chica, Imbachi, & Botero Vega, 2020; Singh & Bhandari, 2020; Ujjan et al., 2020; Yi, Wang, Huang, & Zhao, 2020).

An Intrusion Detection System (IDS) is a system designed to automate the intrusion detection process and help identify abnormal behavior, leading to the discovery and identification of actual attacks (Abdulqadder, Zhou, Zou, Aziz, & Akber, 2020; Gamage & Samarabandu, 2020; Sovilj, Budnarain, Sanner, Salmon, & Rao, 2020). An IDS can be divided into host-based (HIDS) or network-based (NIDS); the former recognizes any unusual patterns in the hosts, the latter recognizes these patterns in the network, protecting against possible intrusions (Masdari & Khezri, 2020). NIDS can be divided into signature-based and profile-based; the signature-based approaches use the signature of the attacks to detect them in the network traffic. Several new attacks frequently emerge, requiring a continuous updating of these signatures, a drawback of this approach. On the other hand, profile-based is driven by network history data, predicting under the assumption that a typical behavior deviation from the actual one indicates an attack. One of the advantages of this approach is detecting unknown attacks just by analyzing the expected behavior of the network (Gamage & Samarabandu, 2020; Novaes, Carvalho, Lloret, & Proença, 2020).

Different IDS approaches have been proposed to detect anomalies, handling the increasing number of security threats and a massive volume of network data (Jin, Lu, Qin, Cheng, & Mao, 2020; Pena, Barbon, Rodrigues, & Proença, 2014; Sahay et al., 2019; Singh & Behal, 2020). Nonetheless, many approaches demand labeled samples to create a detector module or use costly tuning strategies to achieve desirable outcomes. Both are very prohibitive when dealing with current network speed and complexity (Carvalho et al., 2018).

Dealing with data streams requires algorithms capable of performing fast, and incremental processing of data objects to address time and memory limitations (Kim & Park, 2020; Lopes, Santana, da Costa, Zarpelão, & Barbon, 2020). Stream sensors are numerous and operate at high speed, allowing few opportunities for human intervention, let alone for experts (Ahmad, Lavin, Purdy, & Agha, 2017). In networks, due to the high diversity and volume of traffic, manual labeling is not viable. Thus, online clustering algorithms have been developed to tackle the challenges of detecting attacks without prior knowledge about the data and suitable predictive performance in an online fashion.

In this work, we propose a network-based IDS grounded on the usage of an unsupervised stream algorithm to detect attacks and protect SDN environments. The SDN architecture allows our IDS to be a standalone solution that uses the SDN programmability to increase network monitoring and security. Our approach acts online, updating itself, and profiles the network traffic according to the environment shifts. In this manner, the IDS is frequently adapted and can be ready to detect network anomalies while processing one traffic observation at a time.

We selected DenStream clustering as the unsupervised stream kernel using various datasets with different attack configurations. DenStream (Cao, Estert, Qian, & Zhou, 2006) is one of the most promising and successful algorithms applied in different stream applications (Li, Croitoru, & Yue, 2020; Putina & Rossi, 2020; Tajalizadeh & Boostani, 2019). Furthermore, it (Wankhade, Hasan, & Thool, 2013) described that DenStream requires less processing time and space and can also handle concept drifts. Concept Drift refers to a change in the whole distribution of the problem at a certain point in time, which poses a challenge for the traditional IDS solutions (Babüroğlu, Durmuşoğlu, & Dereli, 2021; Wang & Jones, 2017).

Our IDS was evaluated in terms of detection performance, delay in recognizing an attack, and insights about each infection behavior. To create a critical comparison, we consider the proposed IDS performance with the Half-Space-Trees (HS-Trees) (Tan, Ting, & Liu, 2011), a well-known one-class stream classification algorithm. The main contributions of this work are:

- The usage of unsupervised online anomaly detection to match real-life scenarios;
- Comprehensive support of anomalies based on massive experimentation based on multiple simultaneous attacks with several intensities, duration, and overlapping;
- Analysis of response delay to detect portscan and DDoS attacks under several scenarios.

The remainder of this paper is organized as follows. Section 2 discusses related work. In Section 3, we present the system design principles. Section 4 details the test scenarios and how the experimentation is carried out. Section 5 describes the results and performance of the system. Finally, Section 6 concludes the paper.

## 2. Related work

Many works already use stream mining techniques to perform anomaly detection. Mulinka and Casas (2018) demonstrated the feasibility of using stream mining techniques compared to their respective batch versions. The authors compared kNN, Hoeffding Adaptive Trees (HAT), Adaptive Random Forests (ARF), and Stochastic Gradient Descent (SGD). They also used Adaptive Windowing (ADWIN) to set the size of the windows used. Stream and batch algorithms achieved comparable performance. In some cases, the stream version overcame its batch competitors, particularly in Concept Drift scenarios using ARF and SGD.

Another work comparing batch and stream versions was presented by Shin, Yooun, Shin, and Shin (2018). The compared models were Hoeffding Tree, Naive Bayes (NB), kNN, Very Fast Decision Rules (VFDR), and SGD using the KDDCup 99 dataset. The authors concluded the batch method obtained the best results when evaluating each algorithm using both versions. However, the time and memory used by the stream mode were smaller. The batch mode becomes impractical in resource-limited scenarios due to the memory required, making the stream method a viable alternative in these cases. A deep discussion about the stream method and its capacity to reduce memory consumption can be found in da Costa et al. (2018).

Yin, Xia, Zhang, Sun, and Wang (2018) developed an approach grounded on clustering algorithm as an intrusion detection mechanism. The proposal uses a clustering algorithm to mine the data stream and detect patterns. An attenuating sliding window technique was used to reduce the importance as the data became obsolete. The intrusion detection system alerts the network administrator when a stream is considered abnormal.

Another approach using clustering was, Viegas, Santin, Abreu, and Oliveira (2017), which applied Micro Cluster Outlier Detection (MCOD) for detection. This approach is focused on the adversarial setting area, preventing the attacker from avoiding the intrusion detection mechanism by using advanced attack techniques, e.g., causative and exploratory strategies. According to the authors, the proposed approach dealt with DDoS attacks and provided a constantly updated detection system.

Chenaghlou, Moshtaghi, Leckie, and Salehi (2018) proposed OnCAD, an online clustering algorithm for anomaly detection. Their proposal uses Gaussian Clusters as the primary detection mechanism. The window concept is also used to split the data stream into time windows. The approach initially generates clusters of standard network behavior; as new samples arrive, the system assesses whether a compatible cluster exists. Otherwise, it is considered a possible anomaly or an emerging cluster. In the second stage, all samples from a window are used, and a DBScan is performed to locate emerging clusters and, consequently, the anomalies. The proposed algorithm was better than the existing compared algorithms, Online K-means and Adaptive Resonance Theory (ART-2). However, the time required to run this approach is longer. Synthetic and public datasets were tested, and in both, the proposed system obtained a better detection performance. The downside is the execution time, leading to a detection lag that can compromise the anomaly detection properly and open gaps in the security system.

The work presented by Zolotukhin and Hämäläinen (2018) proposed an approach that detects DDoS attacks on encrypted traffic. The proposal compared three algorithms: K-Means, K-Medoids, and Fuzzy C-Means. Each algorithm was applied in batch and online mode. The detection procedure is based on source IP and port, destination addresses, duration, packet numbers, and bytes sent per second. It also uses statistical data, such as the maximum, mean, and minimum values of the packet size. The online mode uses time windows to cluster the network behaviors. According to the authors, the online version of K-means and offline mode of k-medoids achieved the best results. These approaches were able to achieve a high detection performance and keep the number of false alarms low. This approach is only specialized in DDoS attacks, and they could be explored for other attack types.

Liu, Hu, and Shan (2021) proposed a detection method based on fast Fourier transform (FFT) and information entropy. The authors used the FFT coefficients and entropy values as features to train a neural network (NN) to detect DDoS attacks. The work defined each portion of network traffic data as a network behavior and proved the network traffic data conformed to the Riemann flow structure. A Riemann manifold generalizes Euclidean space's metric, differential, and topological concept to geometric objects that locally have the same structure as Euclidean space but globally can represent curved shapes. The method used the quantitative features of stream data like flow duration, packet amount, packet size, and source and destination IP to calculate frequency-domain information. According to the authors, the proposed method reached high accurate values. However, the false detection rate and the missed detection rate for detecting UDP and UDPlag type was high because the high degree of similarity between two types of the data packet.

Another work using HAT was presented by Corrêa, Enembreck, and Silla (2017). The proposal was an IDS to detect attacks in the Kyoto 2006 dataset. The authors compared the proposed algorithm with kNN, NB, and a Perceptron Neural Network as the algorithm in the classification tree node. The results revealed the advantages of HAT and kNN to detect anomalies. The authors reported that the HAT outperformed the other compared algorithms. However, the generated tree by HAT has many nodes, making it difficult for a network administrator to analyze the generated model.

Gore and Gupta (2014) developed an IDS composed of VFDT and ID3. The system adopted ID3 to generate static rules based on the anomaly signature from KDD Rule Base. According to these rules, the incoming network traffic is classified and forwarded to the VFDT to perform the incremental decision model. These incremental models classify new packets on the network and are continuously updated. The approach was capable of detecting anomalies with an initial accuracy under 70%. As time passes, the model learns the patterns and updates the dynamic rules delivering accuracy boosting.

Among the presented works, the works Mulinka and Casas (2018), Shin et al. (2018), Yin et al. (2018), require labeled samples for the training phase. In addition, there are works that need an initial training phase (offline) to learn the pattern, like as Dong and Japkowicz (2016), Viegas et al. (2017), Yin et al. (2018) or performs a detection through signatures such as Gore and Gupta (2014). Some proposals do not detect anomalies using the sample's positioning in relation to its search space, such as Chenaghlou et al. (2018), Corrêa et al. (2017), Zolotukhin and Hämäläinen (2018). Moreover, none of the cited proposals addressed the attack recognition delay.

Our work presents an unsupervised online approach designed to detect anomalies and get insights from the patterns based on the search space itself. In other words, the network administrator can comprehend the kind of attack, intensity, and evolution based on the cluster position on the projection of the feature space. Further information on multiple simultaneous attacks can also be addressed. Moreover, our analysis considers the lag between the beginning of the attack strike and its recognition.

## 3. Unsupervised online learning

When learning from data streams, algorithms need to deal with constraints, such as the (potentially) infinite size of the stream, and that their probability distribution may change over time (da Costa et al., 2018; Silva et al., 2013). A clustering problem consists of determining a finite set of features that describe the data in an unsupervised learning strategy. Features, also called as attributes, can be quantitative, ordinal, or qualitative when representing a property or attribute from the stream, particularly in network environments, most of them are quantitative, i.e., represented by continuous values. Several features can be used in the network's behavior characterization, such as IP addresses, ports, protocols, among others. Generally, features that describe traffic volume, such as bytes and packets transmitted, are commonly used in anomaly detection. However, these features can cause the wrong classification when few hosts use the network resources massively, generate significant peaks in these features, and distort the real behavior of the network (Carvalho et al., 2018).

Online clustering algorithms need to continuously create and update their clusters while also handling memory and time restrictions. The most common approaches are based on the distance between samples as similarity criteria, which can be organized as Gama (2010), such as Partitioning, Micro-clustering, Grid-based, Model-based, and Density-based algorithms. Partitioning algorithms follow the assumptions of traditional k-Means grounded on minimizing an objective function. Micro-clustering algorithms have deleveraged wide-used clustering methods on stream data, for example, BIRCH (Zhang, Ramakrishnan, & Livny, 1997) and CluStream (Aggarwal, Philip, Han, & Wang, 2003). These algorithms perform in two steps: the former summarize the online data to group the data into micro-clusters, and the latter is an offline step that delivers a general model of micro-clusters. Grid-based and Model-based algorithms use a multilevel granularity structure and model fitting, respectively. Both types are driven by solid a priori knowledge, which poses some challenges when dealing with anomalies.

Density-based algorithms take advantage of the connectivity between regions and their density, e.g., DenStream (Cao et al., 2006), to provide arbitrary numbers and forms of clusters. We consider this type of algorithm suitable for anomaly detection since the clustering procedure does not require hyperparameters related to an expected number of clusters (or behaviors) demanding little previous knowledge or modeling. Moreover, density metrics and cluster structure can help the administrator comprehend the attack intensity and further pattern analysis due to the compactness of representation, capacity to track cluster changes, and clear identification of outliers. Thus, we applied DenStream in our proposed IDS architecture because it matches the stream mining constraints and provides complete attack analysis.

The DenStream algorithm is an adaptation of DBSCAN for data streams grounded on some concepts of micro-clusters. It uses the concept of micro-clusters (MC) to create, delete, and update clusters dynamically using a small amount of memory. A MC is defined as $mc_i$ =

**Table 1**
DenStream hyperparameters.

| Hyperparameter | Description |
|---|---|
| $\lambda$ | Decay factor that limits the influence of past samples |
| $\beta$ | Outlier tolerance factor |
| $\mu$ | Core weight threshold |
| $\epsilon$ | Maximum radius of a micro-cluster |
| $\upsilon$ | Number of instances |
| $\kappa$ | Threshold to define whether P-MC is inside the PA |

$\{\overline{CF^1}_i, \overline{CF^2}_i, w_i\}$, where $\overline{CF^1}_i$ is the weighted linear sum of the feature values of all instances (network flows) in $mc_i$, $\overline{CF^2}_i$ is the weighted squared sum and $w_i$ is the weight of $mc_i$, i.e., the sum of its instances' weights (usually each instance has a weight of 1).

The center of $mc_i$ is defined as $c_{mc_i} = \frac{\overline{CF^1}_i}{w_i}$ and its radius is $r_{mc_i} = \sqrt{\frac{|\overline{CF^2}_i|}{w_i} - (\frac{\overline{CF^1}_i}{w_i})^2}$. MCs are used in three different ways in the DenStream algorithm. A "dense" MC is called a core MC (C-MC), where dense is defined if $w_i \geq \mu$, $\mu$ is a hyperparameter to control the minimum density of an MC to be considered a C-MC. A MC with $w \geq \beta\mu$, where $\beta$ is a hyperparameter, is called the potential MC (P-MC). Lastly, outlier MCs (O-MC) have $w < \beta\mu$ and instances belonging to them are considered outliers. Note that only P and O-MC are updated online with the arrival of new instances.

For every $\upsilon$ (hyperparameter) instances, the P-MCs and O-MCs, which were not updated by any of the $\upsilon$ instances, have an exponential decay applied to them, defined as $2^{-\lambda}$ ($\lambda$ is a hyperparameter). By performing this operation, P-MCs and O-MCs fade and eventually are deleted if not representing a new behavior.

The algorithm is divided into three main parts. First, there is an initialization step, where the DBSCAN algorithm is applied to the first $p$ (hyperparameter) instances. After creating the initial clusters, those with $w > \beta\mu$ are initialized as P-MCs. After that, new instances are presented in an online fashion.

The DenStream tries to incorporate each new instance $i$ to its nearest P-MC $pmc_n$. If, after incorporating $i$, $r_{pmc_n} < \epsilon$ would hold true, then the instance is added to that P-MC, where $\epsilon$ is a hyperparameter that limits the maximum value of a MC's radius. Otherwise, the algorithm tries to add $i$ to its nearest O-MC $omc_n$, employing the same test. If it cannot incorporate $i$ (i.e., because it is too far away), then a new O-MC containing only $i$ is created. Then, a procedure is employed to check if any O-MC has enough weight to be promoted to a P-MCs.

The last part consists of creating the final clusters (i.e., C-MCs) by applying the DBSCAN algorithm to the existing P-MCs. First, starting a cluster with an arbitrary P-MC $pmc_a$, it incorporates all the other P-MCs where $distance(pmc_a, pmc) \leq r_{pmc_a} + r_{pmc}$, where $pmc$ is a P-MC excluding $pmc_a$. Then, for each new P-MC in that cluster, the same scanning process is performed. This is performed until no new P-MC is added to that cluster. After that, if the total summed weight of P-MCs in that cluster is greater than $\mu$, it becomes a final cluster. This process repeats until all P-MCs are part of a final cluster or cannot be incorporated into any other. However, if a P-MC is in a potential area (PA), this P-MC will be excluded from the C-MC creation. This exclusion was necessary to prevent the C-MC from being affected by anomalous behaviors present in these P-MC. The potential area concentrates P-MCs that behave differently from C-MCs' expected behavior. DenStream pseudocode is shown in Algorithm 1.

As the paper's goal is to detect DDoS and portscan attacks, one of PA's best features is destination port entropy. PA is defined by the entropy of the destination port of C-MC and the hyperparameter $\kappa$. If the absolute distance between the entropy values of the destination port of the C-MC and the P-MC is greater than $\kappa$, this P-MC is inside the PA. In this manner, it is possible to detect a portscan attack when the P-MC destination port entropy value is higher than that of a C-MC. On the other hand, when this value is lower than a C-MC, it is considered a DDoS attack. Additionally, P-MCs within a PA have an extra exponential decay applied to them every new instance, defined as $1.1^{-\lambda}$. Using extra decay, it is possible to detect the end of the attack more accurately.

Table 1 summarizes the DenStream hyperparameters and presents their description.

In our proposal, each MC is related to a network pattern. O-MCs are the first MC created in the learning process and cannot provide any clear information about possible attacks in the network. When a new P-MC appears, it can be considered an anomaly when its position is inside the PA. Furthermore, its position in the clustering space reveals the nature of a given threat. The common behavior is represented by one or more C-MCs.

**Algorithm 1:** DenStream($Stream$, $\lambda$, $\beta$, $\mu$, $\epsilon$, $\upsilon$, $\kappa$)

```
//Get the next sample p at current time t from Stream;
//Try to merge p into its nearest P-MC;
if r_p (radius of P-MC) ≤ ε then
    Merge p into P-MC;
else
    //Try to merge p into its nearest O-MC;
    if r_o (radius of O-MC) ≤ ε then
        Merge p into O-MC;
        if w (the new weight of O-MC) > βμ then
            //Remove O-MC and create a new P-MC by O-MC;
        else
            //Create a new O-MC by p and insert it into the
              buffer;
        end
    end
end
//Checking each p on T_p (time periods) for applying
  exponential decay;
//T_p = ⌈(1/λ) log(βμ/(βμ−1))⌉;
if (υ mod T_p)=0 then
    for each P-MC do
        if w (the new weight of P-MC) ≤ βμ then
            // Remove P-MC;
        end
    end
    for each O-MC do
        if w (the new weight of O-MC) ≤ κ then
            // Remove O-MC;
        end
    end
end
```

## 4. Test scenario and evaluation

We propose to use the DenStream algorithm as an IDS kernel inside Floodlight,[1] a Java-based open source SDN controller used in these scenarios. The DenStream uses entropy features of source and destination IP addresses and ports for the detection process. If P-MC arises, several assumptions are evaluated, and an alarm is triggered in a portscan or DDoS attack. Our IDS and DenStream algorithm,[2] were implemented in Python 3.7. Fig. 1 presents an overview of our proposal.

The datasets were build using the Mininet[3] emulator and Open vSwitch[4] to control the network switches. Some factors influenced the

[1] https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/
[2] https://github.com/vturrisi/anomaly_detection_sdn
[3] http://mininet.org/overview
[4] https://www.openvswitch.org/

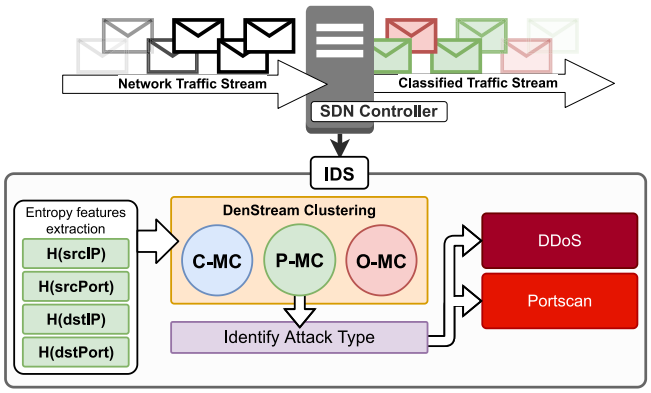



**Fig. 1.** Overview of proposed IDS architecture.

**Table 2**
Attributes used in DenStream algorithm.

| Attributes | Description | Std Dev | Mean value | Max value |
|---|---|---|---|---|
| $H(srcIP)$ | Entropy of Source IP Addresses | 0.341 | 5.430 | 6.599 |
| $H(srcPort)$ | Entropy of Source Ports | 0.315 | 3.998 | 7.098 |
| $H(dstIP)$ | Entropy of Destination IP Addresses | 0.684 | 4.705 | 5.481 |
| $H(dstPort)$ | Entropy of Destination Ports | 0.538 | 3.638 | 6.401 |

decision to choose the Mininet as the environment used to carry out the experiments: (i) Mininet is a resolute project that has an active community; (ii) since its inception, Mininet has received significant effort to support the SDN technologies, such as the OpenFlow protocol and several controllers; (iii) applications developed in Mininet can be deployed in a real environment with few or no changes. To do so, the emulator allows to configure and monitor various aspects of the network in real-time; (iv) Mininet is already considered a de facto standard for teaching, research, and SDN solutions prototyping. Moreover, Mininet is the most widely known and used tool for SDN simulation (Khorsandroo, Sánchez, Tosun, Arco, & Doriguzzi-Corin, 2021; Singh & Bhandari, 2020; Yurekten & Demirci, 2021) because it provides an accurate SDN environment for testing and validation.

To ensure a realistic scenario of an SDN environment with high traffic rates, we employed a tool called Scapy[5] to inject traffic into the emulated network. Scapy is a powerful tool to support a test environment, forging packets and sending them through the network interface. The flows that made up the network traffic were produced randomly and the volume of data changed throughout the day to simulate variations in legitimate network usage.

We collected the one-day stream generating a total of 86400 samples (network flows) per dataset. Four attributes were collected from the network flows: source IP address, destination IP address, source port, and destination port from the stream. Then, we computed the Shannon entropy to extracts information from the concentration and dispersion of these attributes. Table 2 summarizes all attributes used in the datasets.

We generated DDoS attacks using hping3[6] tool by flooding a single host with several requests from different sources. The intensity of DDoS attacks was set according to the number of malicious hosts. The attackers' requests were directed to a specific port of the target to overwhelm it and make the service associated with the port unavailable. Each attacker sent 500 UDP datagrams per second on average. Regarding portscan attacks, a host running Scapy crafted and sequentially sent packets with the SYN flag enabled to different ports of the destination host to probe active ports. We also varied the intensity of portscan attacks, changing the time interval between two consecutive malicious packets.

[5] http://www.secdev.org/projects/scapy
[6] https://github.com/antirez/hping

**Table 3**
Dataset terminology.

| Dataset names | DDoS Intensity | DDoS Duration | Portscan Intensity | Portscan Duration |
|---|---|---|---|---|
| *Stream_HS_HS* | High | Short | High | Short |
| *Stream_HS_HL* | High | Short | High | Long |
| *Stream_HS_LS* | High | Short | Low | Short |
| *Stream_HS_LL* | High | Short | Low | Long |
| *Stream_HL_HS* | High | Long | High | Short |
| *Stream_HL_HL* | High | Long | High | Long |
| *Stream_HL_LS* | High | Long | Low | Short |
| *Stream_HL_LL* | High | Long | Low | Long |
| *Stream_LS_HS* | Low | Short | High | Short |
| *Stream_LS_HL* | Low | Short | High | Long |
| *Stream_LS_LS* | Low | Short | Low | Short |
| *Stream_LS_LL* | Low | Short | Low | Long |
| *Stream_LL_HS* | Low | Long | High | Short |
| *Stream_LL_HL* | Low | Long | High | Long |
| *Stream_LL_LS* | Low | Long | Low | Short |
| *Stream_LL_LL* | Low | Long | Low | Long |

**Table 4**
Hyperparameters tuning values.

| Hyperparameters | Suggested values | Best value |
|---|---|---|
| $\lambda$ | 0.03, 0.06, 0.09, 0.12 | 0.06 |
| $\beta$ | 0.1, 0.2, 0.3, 0.4, 0.5 | 0.1 |
| $\mu$ | 250, 500, 1000, 1500, 2000 | 1000 |
| $\epsilon$ | 0.01, 0.02, 0.05, 0.10, 0.15 | 0.05 |
| $\upsilon$ | 250, 500, 750, 1000, 1500, 2000 | 1000 |
| $\kappa$ | 0.1, 0.25, 0.5 | 0.5 |

The first analysis is about the behavior of the proposed IDS. We generated 48 datasets, in which each of them has a DDoS and a portscan. These datasets are divided into three attack strategies: separated, partially overlapping, and totally overlapping. In each strategy, 16 datasets were generated, varying the intensity between high and low and the duration in short or long for both attacks.

We used the following terminology to describe each dataset. All datasets start with the acronym *Stream_*, followed by the configuration of the DDoS attack intensity, *H* for high or *L* for low intensity, and the attack duration, *L* for long attack or *S* for a short attack. The underline separates each attack's configuration, and the portscan configuration follows the same pattern. Lastly, the last letter represents the attack strategy used, *S* for separated, *P* for partially overlapping, and *O* for fully overlapping. For example, in the *Stream_HS_LL_S* dataset, whereas the DDoS attack has high intensity and short duration, the portscan attack has low intensity and long duration, and both attacks are separated. All possible combinations regarding the intensity and duration of DDoS and Portscan attacks are presented in Table 3.

As described in Section 3, some hyperparameters of the DenStream algorithm ($\lambda$, $\beta$, $\mu$, $\epsilon$, $\upsilon$ and $\kappa$) need to be tuned to obtain suitable results. Metrics were necessary to evaluate the tuning process. The metrics used in this process were accuracy, precision, recall, false-positive rate, and f-measure.

Accuracy (Acc) assesses the proportion of correctly classified intervals among all samples. Precision (Prec) emphasizes the detection of abnormal intervals and penalizes the misclassified normal intervals. Acc and Prec complement each other, showing suitable results when the classes are unbalanced. Recall (Rec) indicates the proportion of correctly classified among all anomalous samples. F-measure (Fm) consists of a global score given to the classifier, and the score is the harmonic mean between precision and recall. All presented metrics range from 0 to 1, in which the former is the worst-case scenario, and the latter represents the optimal value. Finally, the false-positive rate

**Table 5**
Attack detection and delay of the first analysis in terms of number of instances.

| Dataset | Separated | | Partially overlapping | | Fully overlapping | |
|---|---|---|---|---|---|---|
| | DDoS | PortScan | DDoS | PortScan | DDoS | PortScan |
| *Stream_HS_HS* | 139 | 277 | 156 | 427 | 170 | ✗ |
| *Stream_HS_HL* | 155 | 259 | 103 | 679 | 99 | 403 |
| *Stream_HS_LS* | 190 | ✗ | 165 | ✗ | 150 | ✗ |
| *Stream_HS_LL* | 151 | 2277 | 148 | 699 | 137 | 2222 |
| *Stream_HL_HS* | 157 | 159 | 160 | 261 | 184 | ✗ |
| *Stream_HL_HL* | 160 | 246 | 393 | 341 | 147 | ✗ |
| *Stream_HL_LS* | 166 | ✗ | 162 | ✗ | 151 | ✗ |
| *Stream_HL_LL* | 166 | 1772 | 158 | 458 | 147 | ✗ |
| *Stream_LS_HS* | 184 | 258 | 104 | 332 | 208 | ✗ |
| *Stream_LS_HL* | 186 | 231 | 101 | 334 | 100 | 291 |
| *Stream_LS_LS* | 170 | ✗ | 230 | ✗ | 227 | ✗ |
| *Stream_LS_LL* | 173 | 1154 | 178 | 844 | 195 | 2518 |
| *Stream_LL_HS* | 120 | 339 | 224 | 261 | 221 | ✗ |
| *Stream_LL_HL* | 182 | 160 | 102 | 346 | 243 | ✗ |
| *Stream_LL_LS* | 203 | ✗ | 163 | ✗ | 167 | ✗ |
| *Stream_LL_LL* | 141 | 2662 | 188 | 606 | 233 | ✗ |

(FP) indicates the balance of misclassified among all normal samples, and better results are achieved when the value tends to zero.

The hyperparameter tuning process was done through the grid search technique.[7] (Veloso, Gama, Malheiro, & Vinagre, 2021) The dataset used in this process were random subsets of 48 generated datasets, which represent 10 percent of each dataset. During the tuning process, no sample labels are used because Denstream is an unsupervised algorithm. These labels are only used to verify the performance of the algorithm. For every test in a grid search, the prequential method (Lopes et al., 2020) was performed sample by sample.

The prequential learning method is a methodology used in streaming scenarios for evaluating the algorithms over time. Table 4 summarizes all hyperparameters tested values and the best values achieved. We selected values that focused on providing comprehensive and accurate results by balancing the trade-off between true and False Positives.

We divided the evaluation of our proposal into three analyses. To evaluate our IDS in the first analysis, we consider two aspects: (a) Detection: the capacity for recognizing a given attack; (b) Latency: the delay to trigger the detection alert according to the number of samples elapsed between the attack start and P-MC creation. The second analysis explores the capacity of our IDS to identify specific attacks by observing the cluster (P-MC) spatial position following the premises of concentration and dispersion of entropy. The third analysis uses six datasets to compare the overall result of our approach's detection to another algorithm.

## 5. Results and discussion

In the first analysis, we used the presented approach to evaluate the 48 datasets' results regarding the attack's detection and the delay in detecting them. The delay is considered the number of analyzed samples between the beginning of the attack and its detection. Additionally, the attack recognition's capability and another analysis of the proposed algorithm were discussed.

### 5.1. Attack's detection and delay in detection

Based on DenStream's foundation, we interpret C-MCs as a common behavior, a P-MC in a PA as a cluster of attack samples, and O-MC as addressing minor adjustments during the learning procedure and noise samples. In this way, Table 5 presents the results of the 48 datasets, divided into three groups: separated, partially overlapping, and fully overlapping. The table also shows whether attacks were detected and the delay in catching them.

The proposed approach was able to detect DDoS attacks in all datasets. It also expressed a detection delay average of 169.93 instances that may be considered a low deferral in recognizing DDoS attacks. The detection of portscan attacks was more challenging than the DDoS ones. Our approach detected the attack in 28 datasets, and the average delay was higher, achieving 743.42 instances.

The most difficult attacks to be detected are those with low intensity (LL) and short duration (LS), as they do not affect the indicators to identify the attack as much. Still, DDoS attacks with this behavior were detected. However, the portscan attack detection was no longer able to obtain good results; only one of the 12 portscan attacks with this behavior was detected. Another scenario that deserves to be highlighted is the portscan attacks that are totally overlapped are scarcely detected: Only 5 of these 16 attacks were correctly detected.

In-depth analysis revealed that when different attacks have the same overlapped duration, the portscan attack is not detected. Whereas a DDoS attack tends to decrease the destination port's entropy, a portscan increases this value. Thus, the DDoS attack negatively influences the detection of the portscan attack. Even a low-intensity DDoS attack is still being detected.

Six datasets were selected for a deeper analysis of our approach. Fig. 2 depicts the entropy value calculated for each dataset during the 24 h of traffic collection. Each row represents a dataset, and the columns are, respectively: source IP (SrcIP) entropy, destination IP (DstIP) entropy, source port (SrcPort) entropy, and destination port (DstPort) entropy. Each graph's green area represents the entropy values at one-second intervals when attacks were not generated. The red area represents the values calculated at the time the attacks were taking place. As can be observed, the entropy values are fairly stable. In contrast, they are sensitive to changes in network usage behavior, for example, in portscan attacks, where the destination port entropy tends to increase, as can be observed in *Stream_HL_LH_S* and *Stream_LL_HL_P* datasets. In the DDoS attacks, the entropy tends to fall, as observed in *Stream_LL_HL_P* and *Stream_LS_LL_S* datasets.

Fig. 3 shows the number of MCs of each type of cluster (C-MC, P-MC, and O-MC) detected in each dataset. The pink vertical marks represent the DDoS attacks, and the yellow vertical mark represents the portscan attacks.

Fig. 3(a) represents the *Stream_HL_HL_S* and as the stream began, a P-MC was created, but the cluster position was outside the PA, so it was not considered an attack. Soon after, the P-MC became C-MC, proving that it was not an attack, but a stage in the approach's learning process. The C-MC remained equal (one C-MC was detected) throughout the analysis period, showing that it represented the core behavior of the network. Both attacks were also correctly detected, being created and deleted in the P-MC near the beginning and end of the attacks, respectively. This dataset is the best possible scenario, as it

[7] https://scikit-learn.org/stable/modules/grid_search.html

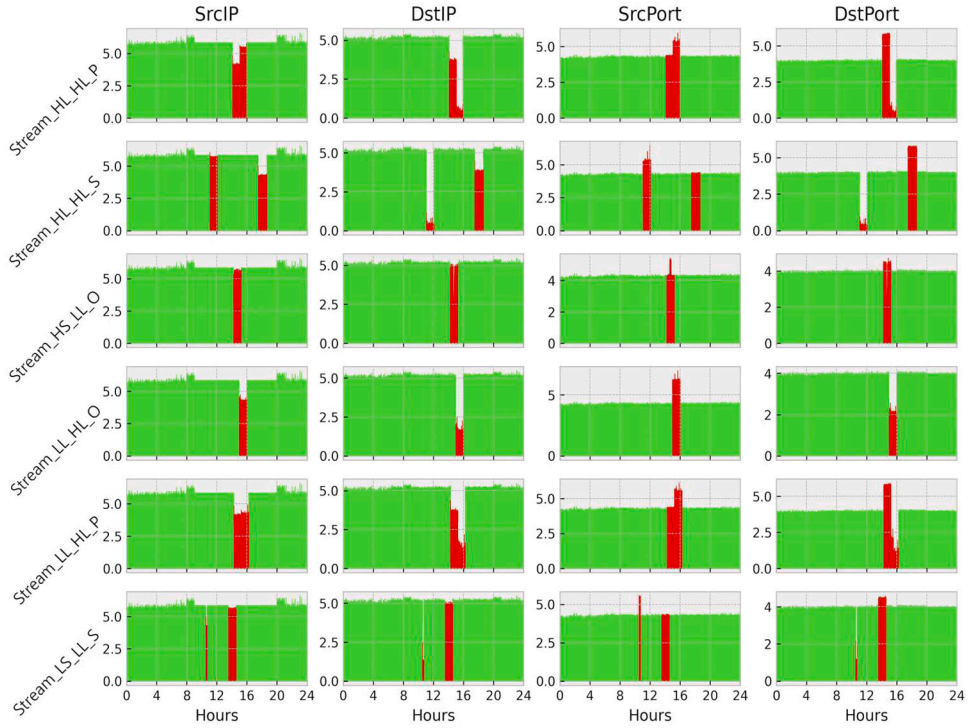



**Fig. 2.** Entropies values of the six datasets used on the more in-depth analysis.

has two high intensity, prolonged, and separated attacks, being possible to detect without significant difficulty.

A more complex scenario is shown in Fig. 3(b). At the beginning of the stream, we observed the approach's learning process in its initial stage. The P-MC portscan attack (marked as a yellow region) had a significant delay in detection (P-MC creation), and this probably occurred because the attack had low intensity. Even with low intensity and short duration, the P-MC DDoS attack (thin pink region) was detected correctly.

Fig. 3(c) represents the *Stream_LL_HL_P*. At the beginning of the portscan attack, a P-MC was created without significant delay. During the portscan attack, the DDoS attack started, and a new P-MC cluster was created, representing the new attack behavior. At the end of the portscan attack, a new P-MC cluster was created, representing only the DDoS attack and excluding the cluster that represented the portscan attack. Thereby, the approach was able to detect attacks individually even when occurring in an overlapping scenario.

However, in some cases, it was not possible to detect overlapped attacks. Fig. 3(d) represents the *Stream_HL_HL_P*. The dataset represents a very similar scenario to the previous one but changes in DDoS intensity. Portscan attack was correctly detected, however, at the beginning of the DDoS attack, a new P-MC was created and excluded the portscan's P-MC. When the portscan ended, no new clusters were created, and it remained active until the end of DDoS, demonstrating that in some cases with low intensity, the detection is tricky. In this scenario, a P-MC representing both attacks was not created; this was because DDoS causes a high distortion of entropy compared to portscan. Thus, even with both attacks, only the DDoS was detected.

Considering a totally overlapping scenario, as in Fig. 3(e) is represented (*Stream_HS_LL_O*), as the streams began, the P-MC was created and soon transformed into C-MC, representing the approach's learning process. A long portscan attack with low intensity has started and was not detected. A DDoS was started too, and this attack was correctly recognized, and only at the end of the DDoS attack, the previous portscan was detected. Finally, Fig. 3(f) represents the *Stream_LL_HL_O*. In this dataset, the anomalous moment was detected correctly; however, it was detected as a DDoS attack and not as a joint attack.

All scenarios exposed a regular pattern for C-MC, P-MC, and O-MC during their processing. C-MC represents the core behavior arising from the beginning of the stream ingestion throughout stream processing. P-MCs outside the PA are not considered alarms, but rather the learning process. They can be related to a concept drift, in which a novel common behavior has been injected synthetically, changing the current pattern. P-MCs inside the PA are considered attacks, and their location in the PA may indicate this attack's nature to be properly mitigated. Therefore, our IDS is susceptible to creating a new P-MC outside the PA when the actual behavior deviates from the previous common behavior. The created P-MC was often transformed into a new C-MC.

### 5.2. Identification of attacks type

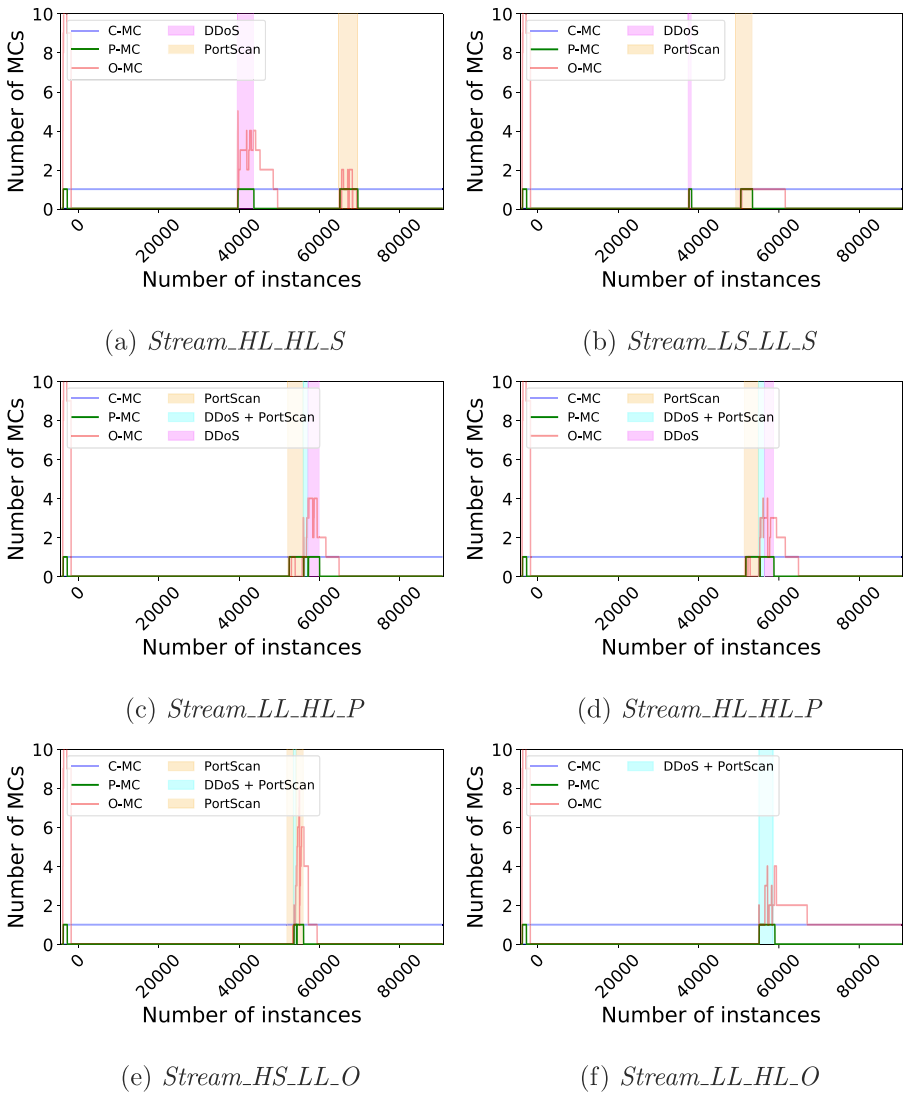
The proposed IDS uses features based on entropy from IP addresses and ports to cluster the network flows and detect the common and anomalous behavior. The identification of attack types are grounded on some premises:

- DDoS: P-MC with lower entropy of destination port than C-MC (common behavior).
- Portscan: P-MC with higher entropy of destination port and lower entropy of source IP than C-MC (common behavior).

We can get some insights when representing the clustering space projected on the entropy of the destination port and the entropy of the source IP, as in Fig. 4. The scenarios from the previous analysis were used. At the moment that a P-MC was created, it was plotted and it has included the identification of which attack was taking place.

Fig. 4(a) shows the cluster space in *Stream_HL_HL_S* dataset. The P-MC with PS label represents the cluster created during the portscan attack. Its position reinforces the premises mentioned, as the entropy of destination port is higher and the entropy of the source IP address is lower than the C-MC cluster. The P-MC with the DDoS label, representing the DDoS attack, has a lower destination port entropy than the C-MC, confirming the premises.

Fig. 4(b) shows the cluster space in *Stream_LS_LL_S* dataset. Again, the P-MC reinforces the approach premises. However, both P-MC were




(a) *Stream_HL_HL_S* (b) *Stream_LS_LL_S*

(c) *Stream_LL_HL_P* (d) *Stream_HL_HL_P*

(e) *Stream_HS_LL_O* (f) *Stream_LL_HL_O*

**Fig. 3.** Different Stream.

closer to C-MC compared to the previous scenario. It occurred because, in this scenario, the attacks have low intensity. Fig. 4(c) shows the cluster space in *Stream_LL_HL_P* scenario. In this dataset, in addition to the 2 P-MC of each attack, an extra P-MC refers to both attacks overlapped. This cluster has a position closer to the DDoS attack, therefore taking into account the premises presented, this behavior represents a DDoS, despite containing two types of attacks simultaneously. Fig. 4(d) shows the cluster space in *Stream_HL_HL_P* dataset. The P-MC positions were similar to the *Stream_HL_HL_S* dataset. Thus, it was found that even if the attacks are partially overlapping, the clusters' position was slightly changed.

Fig. 4(e) shows the cluster space in *Stream_HS_LL_O* dataset. The P-MC labeled DDoS refers to two different attacks running together. The P-MC labeled PS is not so far from the C-MC, as it is a low-intensity portscan attack. Finally, Fig. 4(f) shows the cluster space in *Stream_LL_HL_O* dataset. Only one P-MC was created (DDoS attack) with a very similar position to the P-MC regarding the joint attacks of the *Stream_LL_HL_P* dataset. Due to the intensity and duration of the attack, even having both overlapped attacks, just DDoS was detected in this scenario.

In general, the P-MC's positioning on the projected feature space has attended the premises specified in the attack definition. However, in some cases, when multiple attacks overlap, P-MC's cluster can only reveal the most striking anomalous behavior.

### 5.3. Proposed approach comparison

The previous subsections demonstrated the performance of our approach in detecting DDoS and portscan attacks. However, for additional validation, we compared our approach with HS-Trees (Tan et al., 2011), a one-class anomaly detector for stream data. We selected this algorithm due to its high predictive capacity and successful application in anomaly detection on computer networks (Bhaya & Alasadi, 2016; Pevnỳ, 2016; Tan et al., 2011).

The HS-Trees application for online anomaly detection uses an ensemble of HS-Trees. Each HS-Tree is a binary tree, and its nodes have conditions related to the most important features for a specific decision. All leaves have a mass of elements with the same condition as nodes. The method uses the uniform mass distribution to calculate a score and make anomaly detection. When a new element arrives, each HS-Tree calculates a score for this new element, and the final score is the sum of HS-Trees from the ensemble.

HS-Trees algorithm has three hyperparameters. The first one is $\psi$, which is the size of the window used to create the trees, $t$ is the number of trees, and $h$ is the maximum depth of each tree generated. As HS-Trees is one-class, the first $\psi$ samples need to be free of anomalies for the algorithm's initial training and correct functioning. A grid search was used to define the values of each hyperparameter. F-measure, a harmonic mean between precision and recall, was used for comparing

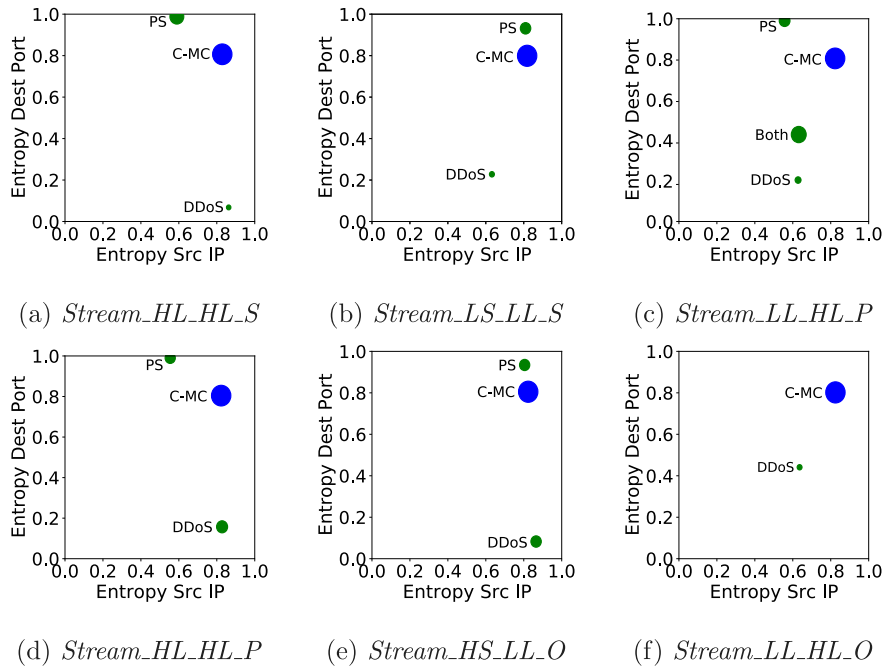



**Fig. 4.** Clustering space of infections after the creation of a P-MC containing the new infection behavior.

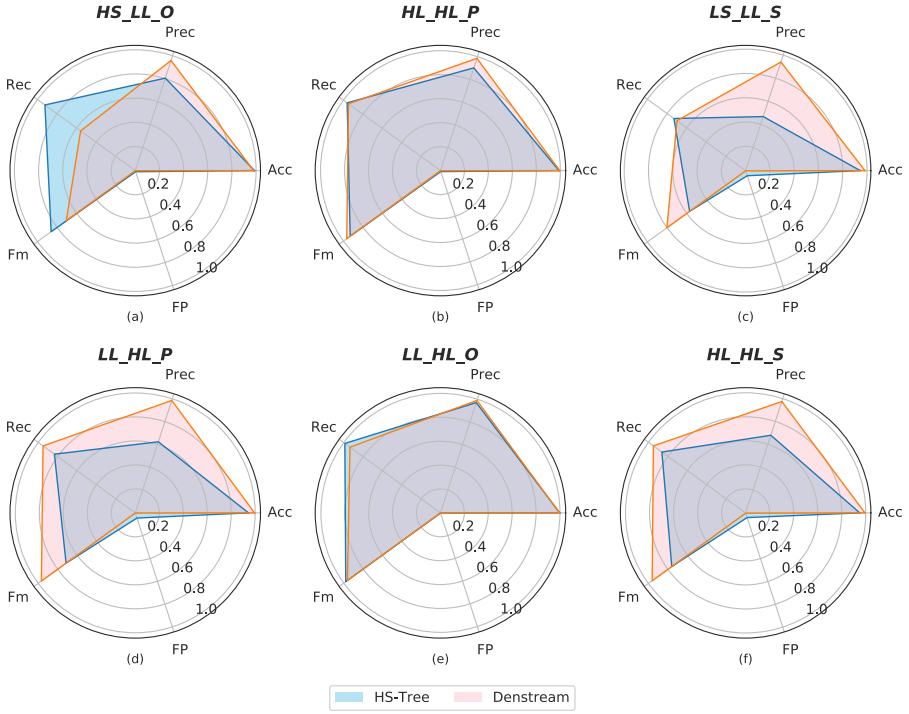
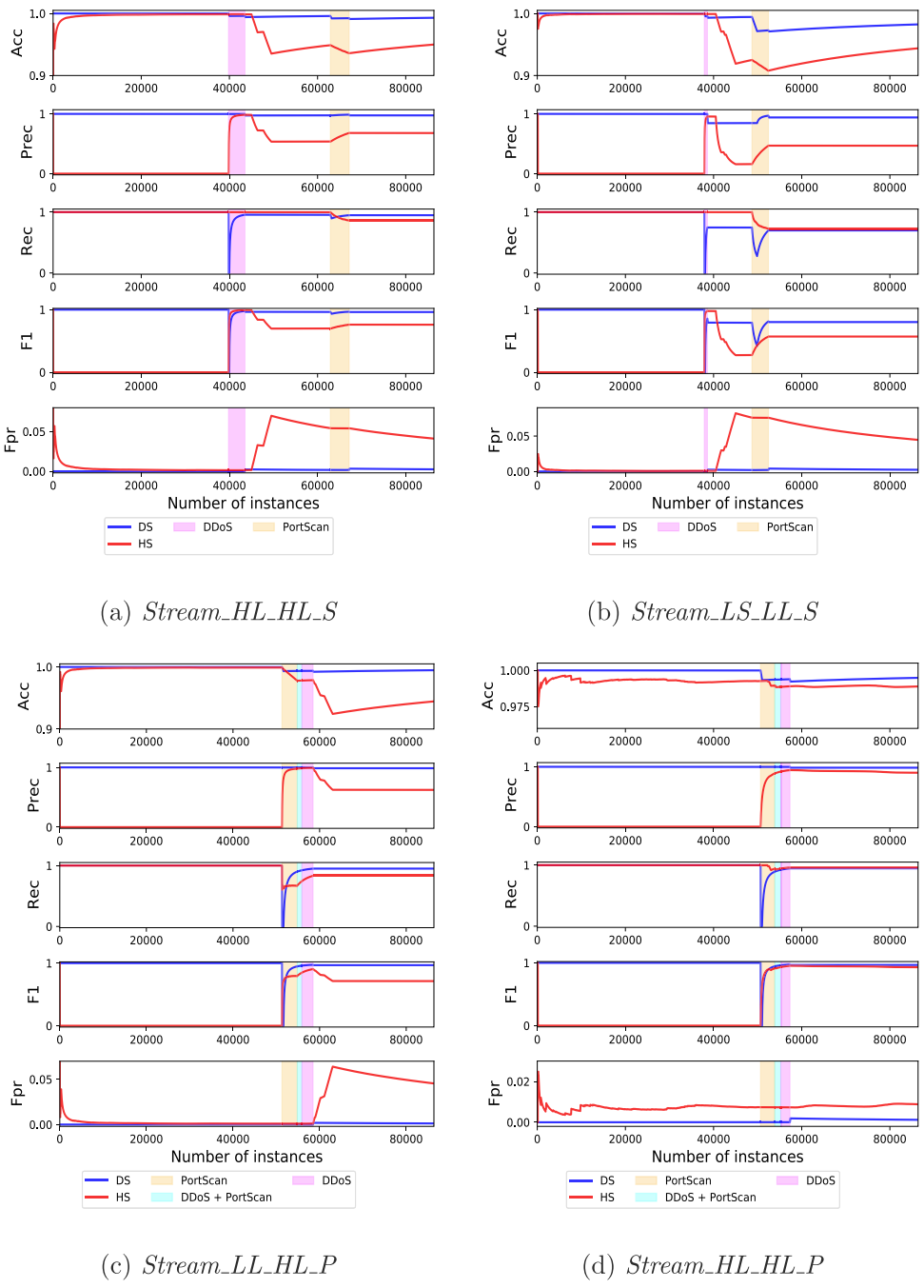



**Fig. 5.** Comparative radar chart between DenStream and HS-Trees.

the overall performance. In the tests, the best f-measure value was achieved when $\psi = 4500$, $t = 3$ and $h = 4$.

Fig. 5 shows the comparison between DenStream and HS-Trees in the scenarios evaluated for identifying the type of attack. As can be seen, DenStream achieved accuracy (Acc) and precision (Pre) rates higher than HS-Trees in all scenarios. The results indicate that DenStream recognized more attack intervals and was less susceptible to false-positives (FP).

In contrast, HS-Trees yielded a higher recall rate (Rec) in the four evaluated datasets and DenStream achieved a higher rate of false-negatives due to the delay in recognizing the beginning of attacks. Whereas DenStream acts completely online, adjusting to constant changes as traffic measurements arrive, HS-Trees require prior training with non-anomalous samples to operate correctly. Although the training samples can decrease the detection time, they can skew the identification of the attack, requiring constant training of the algorithm as the normal network traffic pattern changes. Besides, ensuring that the training samples are attack-free is time-consuming and requires expert evaluation, which is not always feasible.




(a) *Stream_HL_HL_S* (b) *Stream_LS_LL_S*

(c) *Stream_LL_HL_P* (d) *Stream_HL_HL_P*

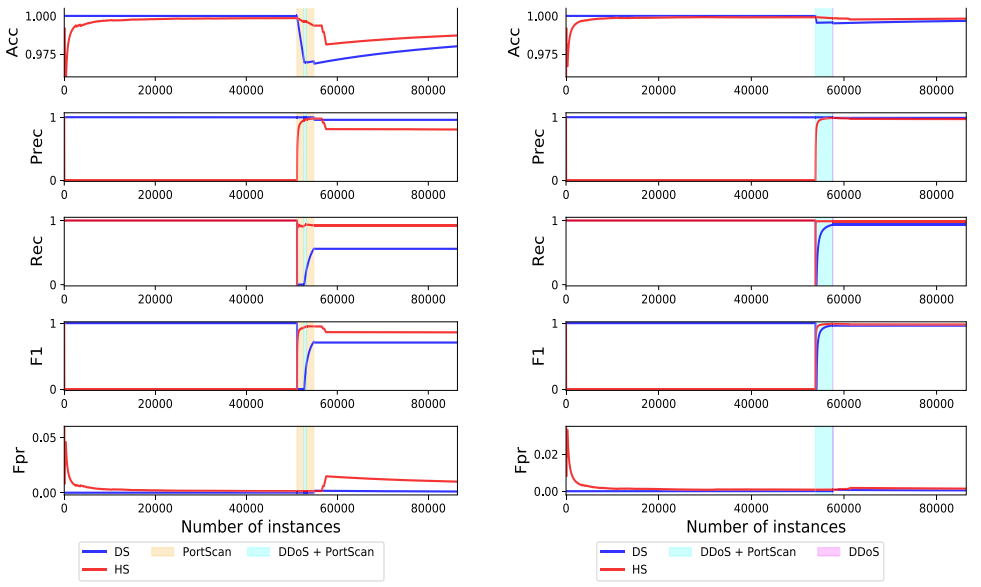
**Fig. 6.** Prequential metric analysis of DS and HS.

As depicted in Fig. 5(a) and Fig. 5(e), HS-Trees slightly outperformed DenStream regarding the overlapped attacks. HS-Trees achieved a satisfactory combination of precision and recall, resulting in a higher f-measure (Fm) in these scenarios. In general, both methods recognized DDoS attacks but were less precise to detect portscan, especially the low-intensity one in the *HS_LL_O* dataset. DenStream exceeded the HS-Trees' results in all other scenarios. The difference is more prominent in scenarios where attacks are separated as in *LS_LL_S* and *HL_HL_S* datasets.

Fig. 6 shows the metrics' results throughout the experiment's execution in each of the datasets. Each chart presents the adaptation of the algorithms with concept drift generated by the attacks. Fig. 6(a) depicts that the values of all metrics were affected right after the start of the DDoS attack, however, DS was able to recognize the anomalous behavior faster than HS and promptly reestablished the metric values to levels reached before the attack. Similar results are also seen in Figs. 6(b)–(d).

Fig. 6(e) and Fig. 6(f) show results for scenarios where attacks are overlapping. The result is similar to the radar charts presented in Fig. 5(a) and Fig. 5(e), in which HS presents results slightly superior to DS. In spite of FPR, HS was able to better adapt itself to the concept drift generated by executing the DDoS and Portscan attacks simultaneously.

For further analysis, Fig. 7, which corresponds to the second row of Fig. 2, shows the entropy values for source and destination IP and ports, calculated during the 24 h of *Stream_HL_HL_S* collection. The green area represents the values calculated at intervals when the attacks have not been generated. The red area corresponds to the entropy values calculated when an attack occurred. The figure also highlights in blue the intervals that DenStream and HS-Trees classified as having attacks. Both compared algorithms were able to classify most of the




(e) *Stream_HS_LL_O* (f) *Stream_LL_HL_O*

**Fig. 6.** (*continued*).

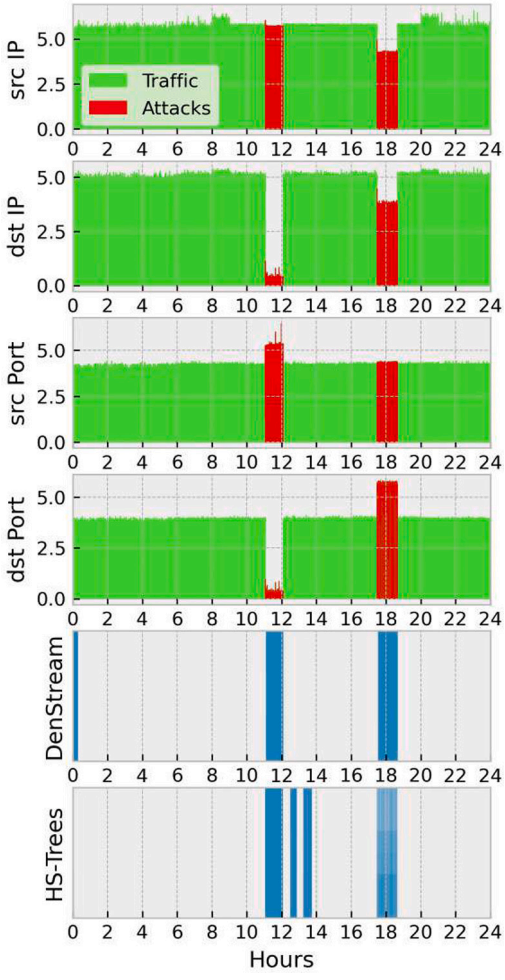



**Fig. 7.** Comparative detection between DenStream and HS-Trees on *Stream_HL_HL_S*.

analyzed one-second intervals correctly. However, DenStream captured more efficiently the disturbances generated by anomalies in IP and port entropies. Besides, fewer false-positives were raised during its execution. This analysis corroborates the results expressed in Fig. 5(f). Furthermore, it is important to mention the visual analysis supported by DenStream when comparing the HS-Trees. Using the clustering projection, it is possible to have insights and a more comprehensive analysis about the attack, its amplitude, and likely multiple attacks.

The DenStream complexity has a linear growth as the stream proceeds, even with more features. The memory usage does not significantly enlarge if the amount of flows increases because the algorithm stores the information of clusters rather than each traffic sample individually (Cao et al., 2006). HS-Trees also has linear growth as the stream or data dimensionality increases. Since HS-trees stores binary trees instead of the entire stream, it can run with reduced memory (Tan et al., 2011).

Thus, when comparing our proposal of unsupervised online anomaly detection with a semi-supervised high accurate algorithm, our proposal achieved superior results in detecting and identifying attacks in human-friendly monitoring. This comprehensive monitoring is able to support insights about the intensity, duration, and overlapping of attacks.

## 6. Conclusion

In this work, we proposed an IDS for online detection of DDoS and portscan in the SDN context. We explored the DenStream algorithm to handle the trade-off between predictive performance, low-latency detection, insightful analysis, and complexity of detector generation.

Different attack scenarios were evaluated by simulating 48 datasets. In these scenarios, the attack settings such as intensity, duration, and overlap have been modified. In addition, the tests were divided into three stages. The first one evaluates the detection capacity and delay of our proposal. The second stage explores the identification of the attack type based on premises that use the distortion pattern suffered by entropy during an attack. The last step of the evaluation compared DenStream with another anomaly detection algorithm. In this manner, the experiments carried out made it possible to assess the detection capacity of our IDS in each scenario.

Our IDS detected all DDoS attacks in the most varied scenarios with a short time response. The portscan attacks were more tricky to be

detected because this attack does not distort entropy as much as DDoS. Thus, it took longer to create the P-MC, especially when the intensity of the attack is low and its duration is short. When comparing DenStream and HS-Trees, both approaches showed close results; however, our approach achieved better results, mainly concerning low false-positive rate and suitable portscan detection.

A DenStream's limitation occurs in attacks that do not alter the used features. They may not be detected correctly by the IDS. Furthermore, if the network presents more than one common behavior (C-MC), the algorithm can assume this behavior as an attack. In future works, these issues and the integration of mitigation solutions will be considered.

## CRediT authorship contribution statement

**Gustavo Frigo Scaranti:** Data curation, Formal analysis, Investigation, Methodology, Software, Visualization, Writing – original draft, Writing – review & editing. **Luiz Fernando Carvalho:** Formal analysis, Methodology, Visualization, Writing – original draft, Writing – review & editing. **Sylvio Barbon Junior:** Conceptualization, Formal analysis, Investigation, Methodology, Visualization, Writing – original draft, Writing – review & editing. **Jaime Lloret:** Funding acquisition, Methodology, supervision, Writing – review & editing. **Mario Lemes Proença Jr.:** Conceptualization, Funding acquisition, Methodology, Project administration, Supervision, Writing – review & editing.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

This work was supported by the National Council for Scientific and Technological Development (CNPq) of Brazil under Grant of Projects 420562/2018-4, 310668/2019-0, and 309863/2020-1, and Fundação Araucária (Paraná, Brazil); by the "Ministerio de Economía *y* Competitividad" in the "Programa Estatal de Fomento de la Investigación Científica *y* Técnica de Excelencia, Sub-programa Estatal de Generación de Conocimiento" within the project under Grant TIN2017-84802-C2-1-P.

## References

Abdulqadder, I. H., Zhou, S., Zou, D., Aziz, I. T., & Akber, S. M. A. (2020). Multi-layered intrusion detection and prevention in the SDN/NFV enabled cloud of 5G networks using AI-based defense mechanisms. *Computer Networks*, *179*, Article 107364. http://dx.doi.org/10.1016/j.comnet.2020.107364.

Aggarwal, C. C., Philip, S. Y., Han, J., & Wang, J. (2003). A framework for clustering evolving data streams. In *Proceedings 2003 VLDB conference* (pp. 81–92). Elsevier, http://dx.doi.org/10.1016/B978-012722442-8/50016-1.

Ahmad, S., Lavin, A., Purdy, S., & Agha, Z. (2017). Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, *262*, 134–147. http://dx.doi.org/10.1016/j.neucom.2017.04.070.

Aldribi, A., Traoré, I., Moa, B., & Nwamuo, O. (2020). Hypervisor-based cloud intrusion detection through online multivariate statistical change tracking. *Computers & Security*, *88*, Article 101646. http://dx.doi.org/10.1016/j.cose.2019.101646.

Aldweesh, A., Derhab, A., & Emam, A. Z. (2020). Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. *Knowledge-Based Systems*, *189*, Article 105124.

Babüroğlu, E. S., Durmuşoğlu, A., & Dereli, T. (2021). Novel hybrid pair recommendations based on a large-scale comparative study of concept drift detection. *Expert Systems with Applications*, *163*, Article 113786. http://dx.doi.org/10.1016/j.eswa.2020.113786.

Barakabitze, A. A., Ahmad, A., Mijumbi, R., & Hines, A. (2020). 5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges. *Computer Networks*, *167*, Article 106984. http://dx.doi.org/10.1016/j.comnet.2019.106984.

Bhaya, W. S., & Alasadi, S. A. (2016). Anomaly detection in network traffic using stream data mining. *Research Journal of Applied Sciences*, *11*(10), 1076–1082.

Calikus, E., Nowaczyk, S., Sant'Anna, A., & Dikmen, O. (2020). No free lunch but a cheaper supper: A general framework for streaming anomaly detection. *Expert Systems with Applications*, Article 113453. http://dx.doi.org/10.1016/j.eswa.2020.113453.

Cao, F., Estert, M., Qian, W., & Zhou, A. (2006). Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM international conference on data mining* (pp. 328–339). SIAM.

Carvalho, L. F., Abrão, T., de Souza Mendes, L., & Proença, M. L., Jr. (2018). An ecosystem for anomaly detection and mitigation in software-defined networking. *Expert Systems with Applications*, *104*, 121–133. http://dx.doi.org/10.1016/j.eswa.2018.03.027.

Chenaghlou, M., Moshtaghi, M., Leckie, C., & Salehi, M. (2018). Online clustering for evolving data streams with online anomaly detection. In *Pacific-Asia conference on knowledge discovery and data mining* (pp. 508–521). Springer.

Corrêa, D. G., Enembreck, F., & Silla, C. N. (2017). An investigation of the hoeffding adaptive tree for the problem of network intrusion detection. In *2017 international joint conference on neural networks* (pp. 4065–4072). IEEE.

Correa Chica, J. C., Imbachi, J. C., & Botero Vega, J. F. (2020). Security in SDN: A comprehensive survey. *Journal of Network and Computer Applications*, *159*, Article 102595. http://dx.doi.org/10.1016/j.jnca.2020.102595.

da Costa, V. G. T., de Leon Ferreira, A. C. P., Junior, S. B., et al. (2018). Strict very fast decision tree: a memory conservative algorithm for data stream mining. *Pattern Recognition Letters*, *116*, 22–28.

De Assis, M. V., Novaes, M. P., Zerbini, C. B., Carvalho, L. F., Abrãao, T., & Proença, M. L. (2018). Fast defense system against attacks in software defined networks. *IEEE Access*, *6*, 69620–69639. http://dx.doi.org/10.1109/ACCESS.2018.2878576.

Dong, Y., & Japkowicz, N. (2016). Threaded ensembles of supervised and unsupervised neural networks for stream learning. In *Canadian conference on artificial intelligence* (pp. 304–315). Springer.

Gama, J. (2010). *Knowledge discovery from data streams*. CRC Press.

Gamage, S., & Samarabandu, J. (2020). Deep learning methods in network intrusion detection: A survey and an objective comparison. *Journal of Network and Computer Applications*, Article 102767. http://dx.doi.org/10.1016/j.jnca.2020.102767.

Gore, S., & Gupta, P. (2014). Online network intrusion detection system using VFDT. *International Journal of Emerging Technology and Advanced Engineering*, *4*(6), 536–542.

Jin, D., Lu, Y., Qin, J., Cheng, Z., & Mao, Z. (2020). SwiftIDS: Real-time intrusion detection system based on lightGBM and parallel intrusion detection mechanism. *Computers & Security*, Article 101984. http://dx.doi.org/10.1016/j.cose.2020.101984.

Khorsandroo, S., Sánchez, A. G., Tosun, A. S., Arco, J., & Doriguzzi-Corin, R. (2021). Hybrid SDN evolution: A comprehensive survey of the state-of-the-art. *Computer Networks*, *192*, Article 107981. http://dx.doi.org/10.1016/j.comnet.2021.107981.

Kim, T., & Park, C. H. (2020). Anomaly pattern detection for streaming data. *Expert Systems with Applications*, *149*, Article 113252. http://dx.doi.org/10.1016/j.eswa.2020.113252.

Kopp, M., Pevnỳ, T., & Holeňa, M. (2020). Anomaly explanation with random forests. *Expert Systems with Applications*, *149*, Article 113187. http://dx.doi.org/10.1016/j.eswa.2020.113187.

Li, M., Croitoru, A., & Yue, S. (2020). GeoDenStream: An improved DenStream clustering method for managing entity data within geographical data streams. *Computers & Geosciences*, *144*, Article 104563. http://dx.doi.org/10.1016/j.cageo.2020.104563.

Liu, Z., Hu, C., & Shan, C. (2021). Riemannian manifold on stream data: Fourier transform and entropy-based ddos attacks detection method. *Computers & Security*, *109*, Article 102392.

Liu, G., Quan, W., Cheng, N., Zhang, H., & Yu, S. (2019). Efficient DDoS attacks mitigation for stateful forwarding in internet of things. *Journal of Network and Computer Applications*, http://dx.doi.org/10.1016/j.jnca.2019.01.006.

Lopes, J. F., Santana, E. J., da Costa, V. G. T., Zarpelão, B. B., & Barbon, S. (2020). Evaluating the four-way performance trade-off for data stream classification in edge computing. *IEEE Transactions on Network and Service Management*, 1013–1025. http://dx.doi.org/10.1109/TNSM.2020.2983921.

Masdari, M., & Khezri, H. (2020). A survey and taxonomy of the fuzzy signature-based intrusion detection systems. *Applied Soft Computing*, Article 106301. http://dx.doi.org/10.1016/j.asoc.2020.106301.

Mulinka, P., & Casas, P. (2018). Stream-based machine learning for network security and anomaly detection. In *Proceedings of the 2018 workshop on big data analytics and machine learning for data communication networks* (pp. 1–7). ACM.

Novaes, M. P., Carvalho, L. F., Lloret, J., & Proença, M. L. (2020). Long short-term memory and fuzzy logic for anomaly detection and mitigation in software-defined network environment. *IEEE Access*, *8*, 83765–83781. http://dx.doi.org/10.1109/ACCESS.2020.2992044.

Pena, E. H. M., Barbon, S., Rodrigues, J. J. P. C., & Proença, M. L. (2014). Anomaly detection using digital signature of network segment with adaptive ARIMA model and paraconsistent logic. In *2014 IEEE symposium on computers and communications* (pp. 1–6). http://dx.doi.org/10.1109/ISCC.2014.6912503.

Pevnỳ, T. (2016). Loda: Lightweight on-line detector of anomalies. *Machine Learning*, *102*(2), 275–304. http://dx.doi.org/10.1007/s10994-015-5521-0.

Proença, M. L., Zarpelão, B. B., & Mendes, L. S. (2005). Anomaly detection for network servers using digital signature of network segment. In *Advanced industrial conference on telecommunications/service assurance with partial and intermittent resources conference/e-learning on telecommunications workshop, vol. 05* (AICT/SAPIR/ELETE'05), (pp. 290–295). http://dx.doi.org/10.1109/AICT.2005.26.

Putina, A., & Rossi, D. (2020). Online anomaly detection leveraging stream-based clustering and real-time telemetry. *IEEE Transactions on Network and Service Management*, 1. http://dx.doi.org/10.1109/TNSM.2020.3037019.

Sahay, R., Meng, W., & Jensen, C. D. (2019). The application of software defined networking on securing computer networks: A survey. *Journal of Network and Computer Applications*, *131*, 89–108. http://dx.doi.org/10.1016/j.jnca.2019.01.019.

Sharma, A., Pilli, E. S., Mazumdar, A. P., & Gera, P. (2020). Towards trustworthy internet of things: A survey on trust management applications and schemes. *Computer Communications*, *160*, 475–493. http://dx.doi.org/10.1016/j.comcom.2020.06.030.

Shin, G., Yooun, H., Shin, D., & Shin, D. (2018). Incremental learning method for cyber intelligence, surveillance, and reconnaissance in closed military network using converged IT techniques. *Soft Computing*, *22*(20), 6835–6844. http://dx.doi.org/10.1007/s00500-018-3433-1.

Silva, J. A., Faria, E. R., Barros, R. C., Hruschka, E. R., Carvalho, A. C. P. L. F. d., & Gama, J. a. (2013). Data stream clustering: A survey. *ACM Computing Surveys*, *46*(1), http://dx.doi.org/10.1145/2522968.2522981.

Singh, J., & Behal, S. (2020). Detection and mitigation of ddos attacks in SDN: A comprehensive review, research challenges and future directions. *Computer Science Review*, *37*, Article 100279. http://dx.doi.org/10.1016/j.cosrev.2020.100279.

Singh, M. P., & Bhandari, A. (2020). New-flow based ddos attacks in SDN: Taxonomy, rationales, and research challenges. *Computer Communications*, *154*, 509–527. http://dx.doi.org/10.1016/j.comcom.2020.02.085.

Sovilj, D., Budnarain, P., Sanner, S., Salmon, G., & Rao, M. (2020). A comparative evaluation of unsupervised deep architectures for intrusion detection in sequential data streams. *Expert Systems with Applications*, *159*, Article 113577. http://dx.doi.org/10.1016/j.eswa.2020.113577.

Tajalizadeh, H., & Boostani, R. (2019). A novel stream clustering framework for spam detection in Twitter. *IEEE Transactions on Computational Social Systems*, *6*(3), 525–534. http://dx.doi.org/10.1109/TCSS.2019.2910818.

Tan, S. C., Ting, K. M., & Liu, F. T. (2011). Fast Anomaly Detection for Streaming Data. In *IJCAI*.

Thakkar, A., & Lohiya, R. (2020). Role of swarm and evolutionary algorithms for intrusion detection system: A survey. *Swarm and Evolutionary Computation*, *53*, Article 100631. http://dx.doi.org/10.1016/j.swevo.2019.100631.

Ujjan, R. M. A., Pervez, Z., Dahal, K., Bashir, A. K., Mumtaz, R., & González, J. (2020). Towards sflow and adaptive polling sampling for deep learning based ddos detection in SDN. *Future Generation Computer Systems*, *111*, 763–779. http://dx.doi.org/10.1016/j.future.2019.10.015.

Veloso, B., Gama, J. a., Malheiro, B., & Vinagre, J. a. (2021). Hyperparameter self-tuning for data streams. *Information Fusion*, *76*, 75–86.

Viegas, E., Santin, A., Abreu, V., & Oliveira, L. S. (2017). Stream learning and anomaly-based intrusion detection in the adversarial settings. In *2017 IEEE symposium on computers and communications* (pp. 773–778). IEEE.

Wang, L., & Jones, R. (2017). Big data analytics for network intrusion detection: A survey. *International Journal of Networks and Communications*, *7*(1), 24–31.

Wankhade, K., Hasan, T., & Thool, R. (2013). A survey: Approaches for handling evolving data streams. In *Communication systems and network technologies (CSNT), 2013 international conference on* (pp. 621–625). IEEE.

Yamansavascilar, B., Baktir, A. C., Ozgovde, A., & Ersoy, C. (2020). Fault tolerance in SDN data plane considering network and application based metrics. *Journal of Network and Computer Applications*, *170*, Article 102780. http://dx.doi.org/10.1016/j.jnca.2020.102780.

Yi, B., Wang, X., Huang, M., & Zhao, Y. (2020). Novel resource allocation mechanism for SDN-based data center networks. *Journal of Network and Computer Applications*, *155*, Article 102554. http://dx.doi.org/10.1016/j.jnca.2020.102554.

Yin, C., Xia, L., Zhang, S., Sun, R., & Wang, J. (2018). Improved clustering algorithm based on high-speed network data stream. *Soft Computing*, *22*(13), 4185–4195. http://dx.doi.org/10.1007/s00500-017-2708-2.

Yurekten, O., & Demirci, M. (2021). Sdn-based cyber defense: A survey. *Future Generation Computer Systems*, *115*, 126–149. http://dx.doi.org/10.1016/j.future.2020.09.006.

Zhang, T., Ramakrishnan, R., & Livny, M. (1997). BIRCH: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery*, *1*(2), 141–182. http://dx.doi.org/10.1023/A:1009783824328.

Zolotukhin, M., & Hämäläinen, T. (2018). Data stream clustering for application-layer ddos detection in encrypted traffic. In M. Lehto, & P. Neittaanmäki (Eds.), *Cyber security: Power and technology* (pp. 111–131). Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-319-75307-2_8.