IoT DDoS Detection Based on Stream Learning

Gustavo Vitral Arbex*, **Kétly Gonçalves Machado***, **Michele Nogueira**†**,**
Daniel M. Batista*, **Roberto Hirata Jr.**,*
*Department of Computer Science, University of São Paulo, Brazil
† Department of Computer Science, Federal University of Minas Gerais, Brazil
Emails: gustavo.arbex@usp.br, {ketly, batista, hirata}@ime.usp.br, michele@dcc.ufmg.br

2021 12th International Conference on Network of the Future (NoF) | 978-1-6654-2434-9/21/$31.00 ©2021 IEEE | DOI: 10.1109/NoF52522.2021.9609940

***Abstract*—The Internet of Things (IoT) represents a new reality, as smart devices spread quickly and a higher number of applications arises. This attracts the attention of not only legitimate users but also attackers aiming to jeopardize the entire IoT infrastructure. Intrusion detection mechanisms are paramount in this networking environment as its first line of defense. Hence, this work proposes a Network Intrusion Detection System (NIDS) that deals with the Distributed Denial of Service (DDoS) attack, one of the most critical attacks that occur through IoT. The proposed NIDS uses stream learning to detect DDoS attacks in the IoT network and is designed to be deployed in a fog infrastructure. The detection model, built on Hoeffding Anytime Tree (HATT) algorithm, achieved a 99% accuracy and a 99% recall.**

***Index Terms*—Internet of Things, Security, Intrusion Detection System, Distributed Denial of Service, Stream Learning.**

I. Introduction

The Internet of Things comprises smart devices that can communicate with each other, perform tasks, and even actively participate in decision making without direct human interference. IoT keeps expanding as smart devices (e.g., smartphones, smart appliances, wearable devices) get popular. Part of this expansion results from the importance of these devices to our lives. Due to the growing opportunities in the IoT market, hardware manufacturers and software developers keep promoting the integration of IoT in different environments, such as Smart Homes and Smart Cities.

There is a significant number of IoT applications involved in various and sensitive tasks, generating a considerable volume of network traffic and connecting heterogeneous devices. Hence, there is a high and urgent demand to protect these networks and applications. Several papers [1]–[7] point to the lack of security in IoT and highlight the imminent danger that this problem represents. An example of this risk is one of the most famous attacks that ever occurred through IoT, the Mirai botnet [8]. Mirai is a malicious software that attacks and infects vulnerable IoT devices that become controlled by Command and Control (C&C) servers, composing a botnet. This botnet is responsible for different kinds of DDoS attacks against targets determined by the C&C servers.

DDoS attacks can be detected in several ways. The sooner an attack is detected, the better a target can be protected. Some mechanisms employ not-usual sources of information to early detect DDoS attacks [9], but, in general, the main source used in detection tools is the network traffic flowing to the target. In this case, the main tool deployed is a NIDS.

Considering the emergent need for the development of NIDS capable of dealing with significant challenges that exist in IoT networks [10], such as resource limitations and volume of data, this paper proposes a NIDS with three modules: Traffic Collection, Anomaly Detection, and Mitigation Actions. We aimed to develop a specific NIDS to detect and mitigate DDoS attacks since this is a critical security problem to IoT and we understand that a specialized system to detect a specific attack may achieve better performance than a general approach.

We aimed to build a NIDS that is specifically suitable for IoT environments, specially thinking in therms of resource limitation. To do so, we (i) developed our NIDS architecture having the environment characteristics in mind, designing it to run on the fog infrastructure in the IoT architecture so it can keep its efficiency, even with resource limitation; (ii) employed the stream learning algorithm HATT [11] that is an online learning approach proper to deal with data stream networks such as IoT, it does not need to revisit data, requiring low memory resources, and it is also capable of adapting itself to deal with data distribution variations; and (iii) performed our experiments on the detection model using BoT-IoT Dataset [12], which contains realistic IoT traffic, testing our NIDS against realistic DDoS attack scenarios.

The main contributions of this paper are: (i) The proposal of a detection model based on stream learning; (ii) The results of experiments performed using the BoT-IoT Dataset to evaluate the performance and effectiveness of the detection model proposed; (iii) The proposal of a NIDS architecture that aims to detect and implement actions to help mitigate DDoS attacks in IoT; (iv) Codes produced as open-source software and experiments results shared as open data, both available in [13].

This paper proceeds as follows. Section II presents the related works. Section III reviews the background employed as basis to develop the work. Section IV describes the proposed NIDS. Section V details the experiments performed and the results obtained. Finally, Section VI concludes the paper and highlights future directions.

II. Related Works

Several papers that use learning approaches to implement intrusion detection systems are found in the literature. In [14]

is presented an intrusion detection system architecture that uses Support Vector Machine (SVM), Self Organizing Map (SOM), and Stacked Autoencoder (SAE) for threat detection in IoT networks based on cloud and software-defined networking. An intrusion detection system that uses a Deep Neural Network (DNN) and network virtualization to detect anomalies in IoT is shown in [15]. In [16] the authors present an intrusion detection system that identifies each IoT device's normal behavior, detects infected packets in real-time using a supervised learning approach, and points the type of attack that occurred.

Intrusion detection systems, based on learning techniques that address specifically DDoS attacks in IoT, are proposed in different papers. The work in [17] presents a lightweight intrusion detection system that aims to detect DDoS attacks in IoT, considering the rate of packet arrival to classify the traffic using an SVM classifier. Also, for detecting DDoS attacks in IoT, the work presented in [18] introduces an intrusion detection system that uses multi-objective optimization to select features and classifies the attack with a hybrid detection model based on Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM). A detection system based on Artificial Neural Network (ANN) used to detect DDoS attacks in IoT is presented in [19]. Similar to our work, this last work also uses the BoT-IoT Dataset to train and test its model.

In the sense of data stream learning approaches to construct intrusion detection systems for IoT security, the work presented in [20] introduces a framework that performs the online monitoring of IoT traffic to process it and extract statistical features to determine the presence of attacks. This framework conducts an incremental learning, same as stream learning, using an adaptation of Self-Organizing and Incremental Neural Networks (SOINN) together with a multi-class approach of SVM.

Table I shows a summary of related works, questioning three main aspects of each paper: Does the work propose an intrusion detection system? Does the work apply the stream learning technique in its detection model? Is the work proposal specifically focused on DDoS attacks in IoT applications?

During the literature review, we notice a vast amount of proposals that address intrusion detection systems for IoT security, mainly works based on machine learning and deep learning techniques. However, to the best of our knowledge, no work uses a stream learning algorithm, in a fog infrastructure, to implement an intrusion detection system to detect DDoS attacks in IoT applications. Therefore, the present work aims to fill this gap.

TABLE I
SUMMARY OF RELATED WORK

Work	Year	Intrusion Detection System?	Stream Learning Approach?	Addresses Specifically DDoS Attacks?
Nguyen et al. [14]	2019	Yes	No	No
Thamilarasu and Chawla [15]	2019	Yes	No	No
Anthi, Williams, Słowińska, Theodorakopoulos and Burnap [16]	2019	Yes	No	No
Jan, Ahmed, Shakhov and Koo [17]	2019	Yes	No	Yes
Roopak, Tian and Chambers [18]	2020	Yes	No	Yes
Soe, Santosa and Hartanto [19]	2019	No	No	Yes
Constantinides, Shiaeles, Ghita and Kolokotronis [20]	2019	Yes	Yes	No
Current Work	**2021**	**Yes**	**Yes**	**Yes**

III. PRELIMINARIES

A. DDoS Attacks in IoT

DDoS attacks are performed by multiple devices that are distributed in a network. These malicious devices send forged requisitions to the target aiming to flood the system with data. The target system then becomes inaccessible and denies genuine requisitions that are discarded before its conclusion, since the system is flooded and the data from the forged requisitions consume some resource of the target, such as bandwidth or memory. In IoT, DDoS attacks are performed the same, but they are improved and diversified due to network devices' heterogeneity. Fig. 1 illustrates a DDoS Attack. In this paper, we understand that the botnet can possibly be made of IoT devices and that the target server is part of the infrastructure of an IoT application.

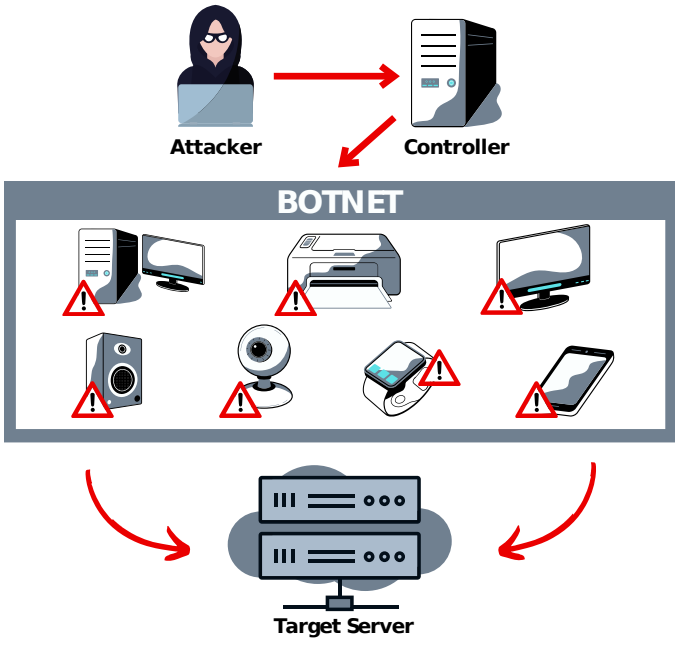

Fig. 1. DDoS Attack

The work of Vishwakarma and Jain [21] presents a taxonomy of DDoS attacks in IoT based in an IoT layered

architecture, where attacks are divided into Application and Infrastructure layers. In the proposed taxonomy, attacks of the Application layer aim to flood the system using HTTP requests and other requests in the IoT architecture application layer. This type of attack includes HTTP flood, DNS service-based attacks, and others. Attacks of the Infrastructure layer explore the transport and network layers of the IoT architecture and are divided in protocol-based attacks, that consume the resources of the target and work in the middle of communication equipment, and volume-based attacks, that generate a massive volume of traffic to flood the target system. Protocol-based attacks include ACK/SYN floods, Ping of Death, Smurf DDoS, and others. Volume-based attacks include UDP/TCP floods, ICMP floods, and others. Furthermore, the authors include Zero-day DDoS attacks, which are unknown or brand new DDoS attacks.

In this paper, we address DDoS UDP, TCP, and HTTP flood attacks since those attacks are present in the BoT-IoT Dataset.

B. Stream Learning Algorithms

Stream learning algorithms are suitable for addressing the security problem in IoT applications since this type of online learning technique is appropriate for dealing with data stream networks. In Fig. 2 we present the stream learning process and highlight where each of our proposed NIDS models will take part.

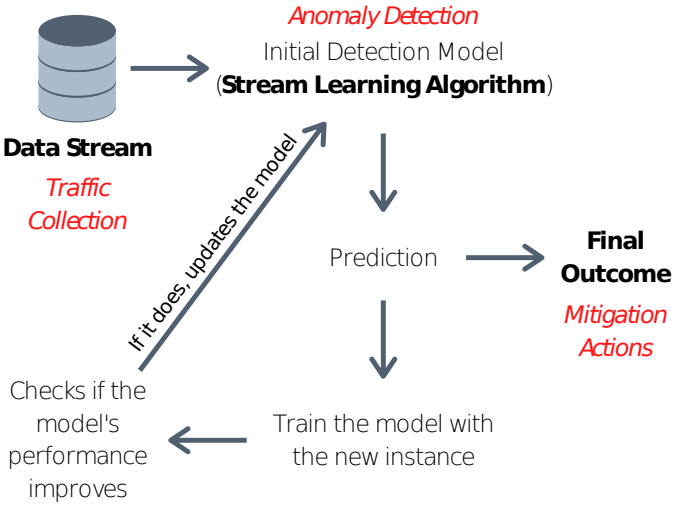

Fig. 2. Stream Learning Process

The initial detection model is previously trained with a small portion of labeled data so it can be used in the stream learning process. As it is possible to notice from Fig. 2, the stream learning algorithm does not need to revisit the data, as one of its fundamentals is to look at one data stream at time and use it only once to train the model, what makes it require a limited amount of memory, which is fundamental for resource constrained environments as IoT. For each new data stream collected from the network, the process presented in Fig. 2 is executed.

In addition, a stream learning algorithm is capable of adapting to the data, meaning that as data distribution changes, the model can adjust itself to manage the variations. Also, it is ready to predict at any moment.

In this paper, our main goal is to investigate the impact of using a stream learning algorithm to build a detection model to detect DDoS attacks in IoT applications.

1) Hoeffding Anytime Tree (HATT): To construct the detection model for the Anomaly Detection module of our NIDS, we used the Extremely Fast Decision Tree (EFDT), the implementation of the HATT stream learning algorithm, presented in Algorithm 1 and Functions 2 and 3 as proposed by Manapragada, Webb, and Salehi [11]. The HATT is an improved version of the Hoeffding Tree stream learning algorithm [22] that uses the Hoeffding Bound to perform the splits of the decision tree. The Hoeffding Tree uses the Hoeffding Bound to decide whether the statistical evidence is sufficient to determine the best attribute to split the tree.

Algorithm 1 Hoeffding Anytime Tree [11]

Input: S, a sequence of examples. At time t, the observed sequence is $S^t = ((\vec{x}_1, y_1), (\vec{x}_2, y_2), ..., (\vec{x}_t, y_t))$
Input: $X = \{X_1, X_2, ..., X_m\}$, a set of m attributes
Input: δ, the acceptable probability of choosing the wrong split attribute at a given node
Input: G(.), a split evaluation function
Output: $HATT^t$, the model at time t constructed from having observed sequence S^t

$HATT \leftarrow$ tree with single leaf, the $root$
$X_1 \leftarrow X \cup X_\emptyset$
$G_1(X_\emptyset) \leftarrow G$ obtained by predicting the most frequent class in S
for class y_k **do**
  for value x_{ij} of each attribute $X_i \in X$ **do**
    $n_{ijk}(root) \leftarrow 0$
  end for
end for
for example $(\vec{x}, y)$ in S **do**
  Sort $(\vec{x}, y)$ into a leaf l using $HATT$
  for node in path $(root...l)$ **do**
    for x_{ij} in $\vec{x}$ such that $X_i \in X_{node}$ **do**
      Increment $n_{ijk}(node)$
      if $(node = l)$ **then**
        $AttemptToSplit(l)$
      else
        $ReEvaluateBestSplit(node)$
      end if
    end for
  end for
end for

The HATT algorithm is an extension of the Hoeffding Tree, in the sense that it improves the splitting decision-making process with the idea that a minor split is better than no split at all. Instead of splitting the tree only when it calculates the best split, the HATT performs the split if the best candidate

provides an information gain larger than zero. After some iterations, the splits are revisited, and if a better split is detected, the HATT updates it.

Our implementation of the HATT algorithm is an adaptation of a code available at [https://github.com/doubleplusplus/incremental_decision_tree-CART-Random_Forest_python/blob/master/efdt.py], which is based in the original work by Manapragada, Webb, and Salehi [11].

Function 2 AttemptToSplit(leafNode l) [11]

Label l with the majority class at l
if all examples at l are not of the same class **then**
  Compute $\overline{G_l}(X_i)$ for each attribute $X_l - X_\emptyset$ using the counts $n_{ijk}(l)$
  $X_a \leftarrow$ the attribute with the highest $\overline{G_l}$
  $X_b \leftarrow X_\emptyset$
  Compute the Hoeffding Bound ϵ
  if $\overline{G_l}(X_a) - \overline{G_l}(X_b) > \epsilon$ and $X_a \neq X_\emptyset$ **then**
    Replace l by an internal node that splits on X_a
    for each branch of the split **do**
      Add a new leaf l_m
      $X_m \leftarrow X - X_a$
      $\overline{G_m}(X_\emptyset) \leftarrow$ the G obtained by predicting the most frequent class at l_m
      for each class y_k and each value x_{ij} of each attribute $X_i \in X_m - \{X_\emptyset\}$ **do**
        $n_{ijk}(l_m) \leftarrow 0$
      end for
    end for
  end if
end if

C. BoT-IoT Dataset

To train and test our detection model, we used the BoT-IoT Dataset [12], a public dataset that was released in 2018 and contains realistic IoT traffic that incorporates normal and attack packets. The entire dataset has more than 72 million records, of which 9543 represent normal traffic, and more than 38 million consists of DDoS attack instances. Of these 38 million instances, more than 19 million consists of TCP flooding, more than 18 million consists of UDP flooding, and about 19 thousand consists of HTTP flooding.

The dataset is available in different formats, including CSV files, which we chose to use in this work. In the files are present 35 distinct features that describe the traffic data. To work with the dataset, we performed some preprocessing in the data. First, we generated a CSV file containing all DDoS attack instances and a CSV file containing all normal traffic. After that, we joined both types of instances in a single CSV file selecting a sample of the attack instances accordingly to the experiment we wanted to perform. All preprocessing codes are available at the previously mentioned repository, and the dataset files can be downloaded at [https://cloudstor.aarnet.edu.au/plus/s/umT99TnxvbpkkoE].

Function 3 ReEvaluateBestSplit(internalNode int) [11]

Compute $\overline{G_i nt}(X_i)$ for each attribute $X_i nt - X_\emptyset$ using the counts $n_{ijk}(int)$
$X_a \leftarrow$ the attribute with the highest $\overline{G_i nt}$
$X_{current} \leftarrow$ the $current$ split attribute
Compute the Hoeffding Bound ϵ
if $\overline{G_l}(X_a) - \overline{G_l}(X_{current}) > \epsilon$ **then**
  if $X_a = X_\emptyset$ **then**
    Replace internal node int with a leaf (kills subtree)
  else if $X_a \neq X_{current}$ **then**
    Replace int with an internal node that splits on X_a
    for each branch of the split **do**
      Add a new leaf l_m
      $X_m \leftarrow X - X_a$
      $\overline{G_m}(X_\emptyset) \leftarrow$ the G obtained by predicting the most frequent class at l_m
      for each class y_k and each value x_{ij} of each attribute $X_i \in X_m - \{X_\emptyset\}$ **do**
        $n_{ijk}(l_m) \leftarrow 0$
      end for
    end for
  end if
end if

As the BoT-IoT Dataset is unbalanced in terms of normal and attack packets, we used the Adaptive Synthetic (ADASYN) [23] sampling approach to resampling the minority class. Algorithm 4 presents the structure of the ADASYN sampling approach. At first, the algorithm calculates the degree of imbalance di. If this degree is lower than a threshold t, the algorithm knows it needs to balance the classes.

IV. The Proposed Network Intrusion Detection System (NIDS)

We idealized our NIDS to follow the structure presented in Fig. 3. As a result, the proposed system is intended to be executed at the fog computing level of the IoT architecture. Thereby, we want to allow the system to access all the computational resources the fog can offer, which are more powerful than the resources provided to a system executed in the gateway or in the device itself. Meanwhile, we also want our detection system to be closer to the device than the systems running in the cloud, allowing faster communication during the execution of mitigation actions.

To effectively improve our NIDS performance, we divided the system into three different modules with distinct attributions. These modules perform independently, and as the result of the whole system, we have an alert scheme that provides information about a possible DDoS attack to the network operators.

A. Traffic Collection

The first module of the NIDS is responsible for collecting the traffic that comes through the fog. To do so, it uses a sniffer software to capture the packets. These packets are then

Algorithm 4 ADASYN [23]

Input: Dataset (Unbalanced)
Output: Dataset (Balanced)

$X \leftarrow$ minority class instances of Dataset
$M \leftarrow$ majority class instances of Dataset
$mi \leftarrow |X|$
$ma \leftarrow |M|$
$di \leftarrow mi/ma$
if $(di < t)$ **then**
  $syn \leftarrow (ma - mi) \times bl$
  $sum \leftarrow 0$
  for $x_i \in X$ **do**
    $K_i \leftarrow k$ nearest neighbors of x_i (Euclidean Distance)
    $km_i \leftarrow |K_i \cap M|$
    $r_i \leftarrow km_i/k$
    $sum \leftarrow sum + r_i$
  end for
  for $i \leftarrow 1$ **to** mi **do**
    $nr_i \leftarrow r_i/sum$
    $s_i \leftarrow nr_i \times syn$
  end for
  for $x_i \in X$ **do**
    for $j \leftarrow 1$ **to** s_i **do**
      $k_{ij} \leftarrow$ random k_i where $k_i \in K_i$
      $sx_{ij} \leftarrow x_i + (k_{ij} - x_i) \times \xi$
      Dataset $\leftarrow$ Dataset $\cup \{sx_{ij}\}$
    end for
  end for
end if

treated and transformed in data structures that are preprocessed to comply with the Anomaly Detection module requirements.

B. Anomaly Detection

This module is the core of the NIDS and receives data from the Traffic Collection module. The Anomaly Detection module is responsible for identifying whether the traffic is malicious or not. It consists of a detection model that uses the EFDT implementation to online analyze the data streams and detect possible DDoS attacks occurring through the IoT application network.

C. Mitigation Actions

The execution of actions that aim to help the network operators prevent DDoS attacks in IoT applications is performed at the Mitigation Actions module. This module is in charge of implementing mitigation actions regarding the DDoS Attacks, raising alerts, and providing detailed information about the problem when such type of attack occurs to assist network operators in the networking monitoring task.

V. Experiments and Results

To evaluate the detection model's effectiveness and performance, we run different experiments using the BoT-IoT Dataset. We used the adapted EFDT implementation to conduct training and testing with different settings and, given the performance evaluation results, changed the hyper-parameters to obtain a low training error and a good generalization.

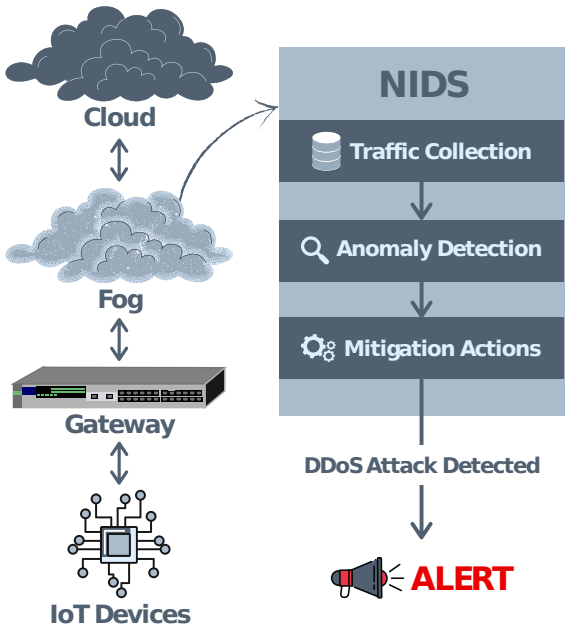

Fig. 3. NIDS Structure

A. Performance Evaluation

We focused mainly on standard metrics to analyze the model's performance because this is a 2-class classification problem. Based on the confusion matrix presented in Table II, we can calculate the accuracy, precision, recall, and F1-score metrics.

TABLE II
Confusion Matrix

		Actual: Positive	Actual: Negative
Predicted	Positive	TP	FP
	Negative	FN	TN

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} \quad (1)$$

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

$$F\text{-}Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (4)$$

Accuracy (1) shows how accurate the model is in making predictions. Precision (2) tells us how much of the positively

predicted instances are, in fact, positive. On the other hand, Recall (3) represents the correctly predicted instances from all the actual positive instances.

Given that the proposed NIDS goal is to stop harmful traffic while allowing normal one simultaneously, we cannot focus solely on recall or precision. We need both metrics to provide a useful information, especially since our classes are unbalanced. A better measure for this scenario is the F-score (4), the harmonic average.

B. Experiments

We ran a series of experiments considering different settings regarding the dataset. Since the dataset authors had already created a CSV file containing the ten best learning features, we decided to start the algorithm using this subset, which contains about 3.5 million records.

As of now, the EFDT adapted implementation did not support categorical variables. We used one-hot encoding in the categories with less than ten unique values to mitigate this problem, dropping the others. Based on this preprocessed dataset, we performed the first two experiments with a different number of samples. We were firstly using about 1% of the dataset and then the complete dataset.

Since our focus is on DDoS attacks, the next step was to consider only the packets originated from this category (plus the normal ones). We then obtained a dataset with about two million records.

Given that more than 99% of the samples corresponded to the attack traffic category, we used ADASYN to balance the distribution. This strategy doubled the size to provide similar distributions. The experiments ran on a computer with Ubuntu 18.04, Intel i7, 8GB RAM, and on Google Colab notebooks, free tier.

After preprocessing the BoT-IoT Dataset as described in Section III-C, we performed a second set of experiments using the treated CSV files. However, we could not finish all these experiments since the execution time was too high.

Results presented in the next subsection are from a partial execution of the implementation. These results are based in a dataset that uses 1% of the DDoS attack instances and all the normal instances plus the ADASYN technique to make it balanced, totaling 770639 instances. We also set 75% of the data for training and 25% for testing.

The stream's size, or the size of training sets for the algorithm, was set to 10000 instances each, except for the last one that would have 17979 instances. The results show up to 250000 instances of training, which reflects the information resulting from 9 hours running the EFDT for this setup. The second set of experiments were performed in a computer using Ubuntu 20.04, Intel i5, and 8GB of RAM.

C. Results

Due to the class imbalance existing on the dataset, the first experiment ran in less than one minute, with all metrics equal to one. We understood that this result was biased, even with the dataset shuffling, and ran again with the full dataset this time. This second training, which took about 7 hours, produced acceptable results, above 99% score in all metrics.

This was also seen in the DDoS dataset. The DDoS/Normal ADASYN dataset training took about one day to complete. It also produced very high scores, which may be due to overfitting. In any case, we can say that the model learns. Table III presents the detailed results of the described experiments.

TABLE III
PERFORMANCES OF THE ALGORITHMS/DATASETS

	Small Sample	Full Dataset	DDoS Dataset	ADASYN
Accuracy	1.0000	0.9999	0.9997	1.0000
Precision	1.0000	0.9997	0.9997	1.0000
Recall	1.0000	0.9999	1.0000	1.0000
F-Score	1.0000	0.9998	0.9999	1.0000

Another impressive result was obtained from the experiments where the metrics' printing after a percentage of the examples show us that the metrics improved continuously, with the tree adjusting its splits if it finds a better setting, as we can see in Fig. 4. For all the metrics, the initial value was approximately 0.94. As the detection goes by, the values increase until they reach 0.99.

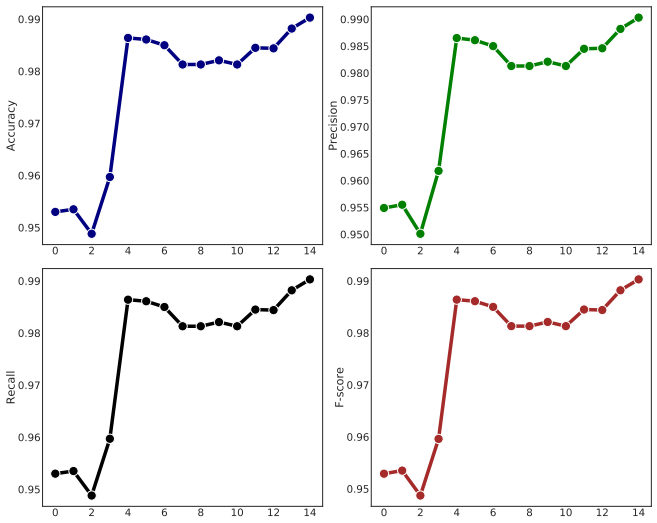

Fig. 4. Accuracy, Precision, Recall and F-score (First set of experiments)

The setup from the second set of experiments presented in this paper, even though it did not finish up, shows high-performance metrics, achieving more than 99% in all metrics. The partial execution also shows an evolution and improvement of metrics as the tree learn, same as the previous experiments, reflecting the algorithm's learning capabilities. Results achieved with this partial execution are presented in Fig. 5.

VI. Conclusions and Future Work

In this paper we propose a robust, adaptable technique to decide whether the IoT incoming traffic is from a DDoS attack

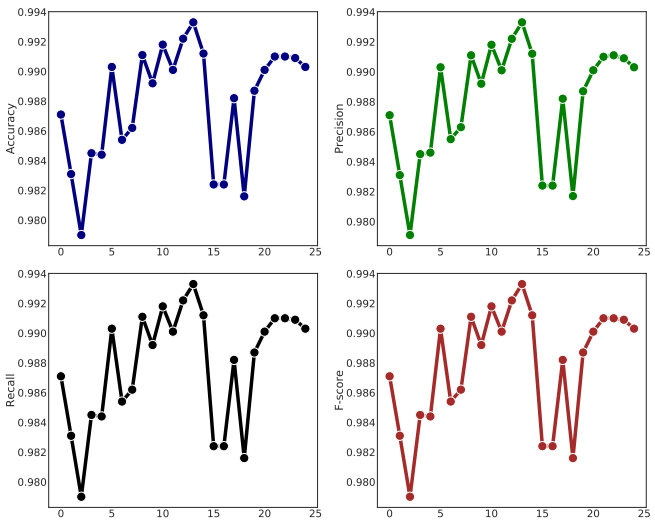

Fig. 5. Accuracy, Precision, Recall and F-score (Second set of experiments)

or not. The technique is based on the HATT algorithm, a stream learning algorithm. Also, the technique is proposed to be deployed in the fog infrastructure of the IoT. As the experiments shown, the detection improves its metrics as the stream advances, meaning that its continuous learning is successful. In this sense, the idea of building a NIDS based on stream learning techniques to detect DDoS attacks in IoT shows itself as promising.

Future works include the following ideas:

- Improve the EFDT implementation code to reduce the execution time and memory consumption;
- Execute the EFDT implementation using all the DDoS attack instances, while using the ADASYN sampling technique to balance the dataset;
- Implement the HATT algorithm inside the Massive Online Analysis (MOA) software [24] to execute a realistic scenario of stream/online learning. MOA is a software for data stream mining with concept drift;
- Elaborate more sophisticated approaches to address the Traffic Collection and Mitigation Actions modules and implement them using diverse mechanisms;
- Implement the proposed NIDS in a real IoT application and perform attack simulations to evaluate the detection model performance in a realistic situation and its feasibility to IoT networks.

The results of this research, including all codes as open-source software and all experiments data as open data, are available in [13].

VII. Acknowledgments

This research is part of the INCT of the Future Internet for Smart Cities funded by CNPq proc. 465446/2014-0, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001, FAPESP proc. 14/50937-1, and FAPESP proc. 15/24485-9. It is also part of the FAPESP proc. 18/22979-2 and FAPESP proc. 18/23098-0.

References

[1] B. Chifor, S. Arseni, I. Matei, and I. Bica, "Security-oriented framework for internet of things smart-home applications," in *2019 22nd International Conference on Control Systems and Computer Science (CSCS)*, 2019, pp. 146–153.

[2] N. Sengupta, "Designing security system for IoT," in *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*, 2019, pp. 195–199.

[3] A. Wang, A. Mohaisen, and S. Chen, "Xlf: A cross-layer framework to secure the internet of things (IoT)," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, 2019, pp. 1830–1839.

[4] J. Bugeja, B. Vogel, A. Jacobsson, and R. Varshney, "IoTsm: An end-to-end security model for IoT ecosystems," in *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, 2019, pp. 267–272.

[5] S. Rathore, B. Wook Kwon, and J. Hyuk Park, "BlocksecIoTnet: Blockchain-based decentralized security architecture for IoT network," *Journal of Network and Computer Applications*, vol. 143, pp. 167 – 177, 2019.

[6] J. Jung, J. Cho, and B. Lee, "A secure platform for IoT devices based on arm platform security architecture," in *2020 14th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, 2020, pp. 1–4.

[7] B. M. Rahal, A. Santos, and M. Nogueira, "A Distributed Architecture for DDoS Prediction and Bot Detection," *IEEE Access*, vol. 8, pp. 159 756–159 772, 2020.

[8] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[9] R. Campiolo, L. A. F. Santos, D. M. Batista, and M. A. Gerosa, "Evaluating the Utilization of Twitter Messages as a Source of Security Alerts," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 2013, p. 942–943.

[10] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for IoT security based on learning techniques," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2671–2701, 2019.

[11] C. Manapragada, G. Webb, and M. Salehi, "Extremely fast decision tree," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1953–1962.

[12] N. KoronIoTis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset," *Future Generation Computer Systems*, vol. 100, pp. 779 – 796, 2019.

[13] G. V. Arbex, K. G. Machado, D. M. Batista, and R. Hirata Junior. (2020) Iot-ids-efdt-ddos. [Online]. Available: https://github.com/ketlymachado/IoT-IDS-EFDT-DDoS

[14] T. G. Nguyen, T. V. Phan, B. T. Nguyen, C. So-In, Z. A. Baig, and S. Sanguanpong, "Search: A collaborative and intelligent nids architecture for sdn-based cloud IoT networks," *IEEE Access*, vol. 7, pp. 107 678–107 694, 2019.

[15] G. Thamilarasu and S. Chawla, "Towards deep-learning-driven intrusion detection for the internet of things," *Sensors*, vol. 19, no. 9, p. 1977, 2019.

[16] E. Anthi, L. Williams, M. Słowińska, G. Theodorakopoulos, and P. Burnap, "A supervised intrusion detection system for smart home IoT devices," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 9042–9053, 2019.

[17] S. U. Jan, S. Ahmed, V. Shakhov, and I. Koo, "Toward a lightweight intrusion detection system for the internet of things," *IEEE Access*, vol. 7, pp. 42 450–42 471, 2019.

[18] M. Roopak, G. Y. Tian, and J. Chambers, "An intrusion detection system against DDoS attacks in IoT networks," in *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, 2020, pp. 0562–0567.

[19] Y. N. Soe, P. I. Santosa, and R. Hartanto, "DDoS attack detection based on simple ann with smote for IoT environment," in *2019 Fourth International Conference on Informatics and Computing (ICIC)*, 2019, pp. 1–5.
[20] C. Constantinides, S. Shiaeles, B. Ghita, and N. Kolokotronis, "A novel online incremental learning intrusion prevention system," in *2019 10th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2019, pp. 1–6.
[21] R. Vishwakarma and A. K. Jain, "A survey of DDoS attacking techniques and defence mechanisms in the IoT network," *Telecommunication Systems*, vol. 73, pp. 3–25, 2020.
[22] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2000, p. 71–80.
[23] Haibo He, Yang Bai, E. A. Garcia, and Shutao Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, 2008, pp. 1322–1328.
[24] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive Online Analysis," *J. Mach. Learn. Res.*, vol. 11, p. 1601–1604, Aug. 2010.