Evaluation of the Latency of Machine Learning Random Access DDoS Detection in Open RAN

Paulo R. B. da Silva, João P. S. H. Lima, Erika C. Alves, Daniel L. Feferman, William S. Farfan, Thomas W. do P. Paiva, Victor A. Coutinho, Francisco Hugo Costa Neto
{prsilva,jsales,erikaa,dlazkani,wfarfan,tpaiva,vaguiarc,fhugo}@cpqd.com.br
Centro de Pesquisa e Desenvolvimento em Telecomunicações (CPQD)
Campinas, Brazil

ABSTRACT

A testbed with open-source tools and commercial equipment is designed to investigate the timing of ML-based xApp operations in DDoS attack detection during the Random Access procedure. Results show high-precision and high-recall ML classifiers operating faster than the Medium Access Control (MAC) layer's Contention Resolution timer. For classifiers with over 97% precision and recall, XGBoost has the fastest execution time and the narrowest estimated run time probability density function.

CCS CONCEPTS

• **Networks → Network performance analysis**; • **Security and privacy → Denial-of-service attacks**.

KEYWORDS

Open RAN, DDoS, Signaling Storm, Machine Learning, RIC

ACM Reference Format:

Paulo R. B. da Silva, João P. S. H. Lima, Erika C. Alves, Daniel L. Feferman, William S. Farfan, Thomas W. do P. Paiva, Victor A. Coutinho, Francisco Hugo Costa Neto. 2024. Evaluation of the Latency of Machine Learning Random Access DDoS Detection in Open RAN. In *The 30th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '24), November 18–22, 2024, Washington D.C., DC, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3636534.3701546

1 INTRODUCTION

The Open Radio Access Network (Open RAN) leverages virtualization and disaggregation to divide the RAN into the Central Unit (CU), Distributed Unit (DU), and Radio Unit (RU), enabling the collection of control data at the interfaces between them. This data is then processed by the RAN Intelligent Controller (RIC), which generates network control signals. To achieve this, the RIC hosts specialized applications, known as xApps, that may employ Artificial Intelligence (AI) to optimize network performance [8].

The experimental evaluation of Open RAN is a topic undergoing continuous research. In [12], the authors assess AI-based RAN management algorithms and propose a new low-latency architecture interface. In [13], equipment from various vendors is integrated, and a digital twin framework for radio frequency planning is tested with iPerf on different Commercial off-the-shelf (COTS) smartphones. In [10], a new Open RAN experimentation platform is presented, along with datasets that test the impact of virtualized base stations in terms of computing and energy. In [7], Deep Reinforcement Learning is explored for managing slicing and scheduling in an Open RAN architecture that uses software-defined radios and the Colosseum wireless network emulator.

New functional splits, interfaces, and virtualization create vulnerabilities to attacks [8]. A threat widely studied in the literature is Distributed Denial of Service (DDoS), in which excessive invalid requests overload a network, denying service to real users. In [14], Machine Learning (ML) is used to analyze traffic patterns and detect Denial of Service attacks in the DU. In [6], ML-based mitigation is proposed for low-rate DDoS attacks in Software Defined Networks (SDNs). In [9], an evolutionary Support Vector Machine (SVM) approach aided by genetic algorithms is applied for detecting attacks in SDNs. In [4], a closed-loop control is developed for massive Machine Type Communications in Internet of Things (IoT) that uses ML to block attackers. These studies focus on Core network attacks, like Transmission Control Protocol (TCP) SYN and HTTP GET flooding. In contrast, we examine the exploitation of the Random Access (RA) procedure (RA DDoS) during the initial User Equipments (UEs) connection request to the RAN stack.

The RA procedure includes a Contention Resolution Timer (CRT) that stops when contentions are resolved. If the CRT expires without resolution, the RA procedure fails. The ML DDoS mitigation operation must be completed before the CRT expires so as to respect the deadline of the RA procedure. It is important to note that ML control loops offer flexibility and adaptability as compared to timers, allowing for nuanced decisions. A possibility is to create an ML DDoS detection system, with the CRT timer serving as its backup in case it fails to identify an ongoing attack.

As it can be seen, latency is an important consideration in virtualized system operations. In [5], a mixed-integer linear programmer is proposed to optimize energy efficiency while guaranteeing short delays for latency-sensitive traffic flows. ScalO-RAN [3] allocates and scales AI-based xApps deployed as microservices to meet specific latency requirements. In [14], results are presented for the time xApp-based ML classifiers spend on the detection of DDoS attacks targeting the core network. These classifiers are trained on tabular feature data rather than on time series, as it is done in this paper. The findings of [14] show that detection methods based on decision trees have lower delays. Similarly, [9] compares the time taken by genetic-algorithm-optimized SVM to detect attacks such as SMURF,

ACM MobiCom '24, November 18–22, 2024, Washington D.C., DC, USA

ACM ISBN 979-8-4007-0489-5/24/11
https://doi.org/10.1145/3636534.3701546

User Datagram Protocol (UDP) flood, and HTTP flood with that of Random Forest (RF) and K-Nearest Neighbors (KNN) detection.

This work examines the inference and control-signaling latency of an ML-based xApp designed to mitigate RA DDoS attacks in a real-world Open RAN setup. Our contributions include: (1) setting up a testbed with software-defined radio, open-source tools, and real/simulated UEs to analyze RA DDoS handling; (2) evaluating ML models within a RIC framework for detecting RA DDoS attacks using time series data; and (3) measuring latency of key steps in an xApp using ML inference to ensure compliance with CRT deadlines.

The rest of this paper is organized as follows. Section 2 reviews the FlexRIC and service models for ML deployment. Section 3 defines the RA DDoS problem, details ML attack detection, and briefly describes control signaling. Section 4 describes the experimental setup. Section 5 features test protocols, ML classifier time profiling for DDoS detection, and a discussion of our findings. Section 6 concludes the paper.

2 RAN INTELLIGENT CONTROLLER

The RIC consists of two modules: the non-Real-Time RIC (non-RT RIC) and the near-Real-Time RIC (near-RT RIC). The non-RT RIC, with response times above 1 second, handles tasks such as model training and policy creation for the near-RT RIC [8]. However, the non-RT RIC is not a primary focus of this paper, as detailed in § 4.

The near-RT RIC is a software-defined controller designed for fast response times, positioned at the edge of regional telecom clouds to meet strict latency requirements. It runs Radio Resource Management (RRM) control loops with response times between 10 ms and 1 s [8] that interact with the DU and the CU [8]. The near-RT RIC uses services and xApps that rely on the E2 interface between the RIC and the CU/DU and on Service Models (SMs) to receive metrics from the RAN for effective RRM. xApps process these Key Performance Metrics (KPMs), generating control signals that are sent to the RAN.

Open Air Interface (OAI)'s FlexRIC is the near-RT RIC used in this work. It extends RAN functionality communicating over the E2 interface services to the near-RT RIC which can be monitored and used by xApps. SMs control the use of these services, enabling their utilization through read and write functions between the near-RT RIC and the gNodeB [11].

3 DDOS DETECTION AND CONTROL

3.1 Problem Definition

The RA procedure uses the Radio Resource Control (RRC) protocol to allow UEs to request a connection to the radio medium. This occurs in several scenarios: establishing a radio link during the transition from RRC_IDLE to the RRC_CONNECTED state, reestablishing links after a failure, performing handovers that require uplink synchronization with a new cell, and achieving uplink synchronization when data arrives with an unsynchronized uplink [1].

Figure 1 shows how the RRC protocol exchanges messages between UEs and the gNodeB to establish a connection. Message (MSG) 3 initiates RRC Connection Request or Reestablishment, while MSG 4 responds with RRC Connection Setup, Reject, or Reestablishment Complete [1]. The RA procedure includes a CRT at the MAC layer, which is started by a UE after it transmits MSG 3.

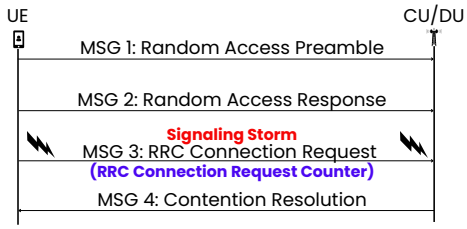

Figure 1: RA with MSG 3 overloading indicated as SS.

Configured using the `ra-ContentionResolutionTimer` parameter, the CRT ranges from 8 to 64 subframes (8 to 64 ms). The timer stops when contention is resolved, i.e., upon receiving MSG 4. If the CRT expires without resolution, the RA procedure fails, discarding temporary values assigned to UE and informing the upper layers of the failure [1].

In a DDoS event, attackers can exploit MSG 3. If too many devices compete for the medium simultaneously, the system may fail to provide all UEs with radio resources, as it is unable to resolve numerous medium access conflicts. More critically, if devices repeatedly send MSG 3 to the RAN in rapid bursts with little delay, known as Signaling Storm (SS), the RAN can exhaust its computing resources, potentially causing a system crash.

To mitigate this vulnerability, an SM and corresponding xApp are developed to monitor, detect, and handle SS. Communication between the gNodeB and the Near-RT RIC platform is established via the E2 interface. Metrics available during the initial access request, such as RA-RNTI, Timing Advance, and RRC Connection Request counts, can be used by ML-based xApps to classify requests as normal or attack and recommend either acceptance or rejection to the RAN, which then sends the appropriate MSG 4 to the UE. ML inference and control signaling must be completed within 8 ms to ensure timely detection and handling of DDoS attacks before the RA failure caused by CRT expiration.

3.2 Machine Learning Handling of RA DDoS Detection

The attack behavior has no specific pattern. The attacker may decide to operate with varying connection request frequencies. The lack of a definite pattern renders impossible the use of pre-established thresholds on RAN parameters to detect attacks. This makes the use of ML highly advantageous as it is flexible enough to address various forms of attack behavior.

The xApp obtains metrics to detect attacks. After the UE's attachment request to the gNodeB in the form of an RRC Setup Request, the gNodeB collects initial access metrics. These metrics are sent to the xApp as an `INDICATION` message from the SM and presented in the form of a message log. The metrics are transferred from the E2 node to the xApp via SM report callback functions. In addition, this information is stored in data banks for ML model training.

DDoS detection poses two main challenges: data is imbalanced, as DDoS events are infrequent compared to normal UE activity, and the binary labeling of individual data samples is difficult due to the high volume of network RA events.

Imbalanced datasets contain underrepresented classes. Managing these datasets involves strategies like random under-sampling, to

reduce instances of the majority class, and random over-sampling and synthetic data creation to augment the minority class [2].

Metrics such as precision, recall, F1-score, and Area Under the Curve (AUC) provide ways to assess imbalanced classification. Let the positive class be the anomalous event. Then, in a highly unbalanced dataset, recall is crucial to avoid false negatives, as undetected attacks compromise the system and disrupt service. High precision is needed to prevent inadvertent detections.

Let N_{P} be the number of positives, N_{PP} be the number of predicted positives, and N_{TP} be the number of true positives. Precision P, recall R, and F1-score F are defined as follows: $P = N_{\mathrm{TP}}/N_{\mathrm{PP}}$, $R = N_{\mathrm{TP}}/N_{\mathrm{P}}$, $F = 2PR/(P+R)$. Additionally, let AUC be the area under the precision-recall curve, rather than the area under the Receiver Operating Characteristic (ROC) curve. Performance is considered to improve as the values of all the aforementioned metrics increase.

In this paper, we focus on ML-based RA DDoS detection using the RRC Connection Request Counter (RCRC) time series, with lengths matching the length of the First In First Out (FIFO) queues described in § 5. This metric is chosen for its ease of implementation in function calls that manage the RRC protocol. More importantly, the RCRC is also chosen because it directly corresponds to the stochastic process that governs network event arrivals, including anomalous RA DDoS attacks. In fact, the arrival process is equivalent to its associated counter process, further justifying the use of RCRC.

After the ML analysis of temporal data, the xApp issues recommendations regarding the access request. In case of acceptance, the xApp sends CONTROL bit 0 to the gNodeB, and in case of rejection, the xApp sends bit 1. The gNodeB then sends a corresponding RRC Setup response to the UE: RRC Connection Setup (acceptance) or RRC Connection Reject.

4 EXPERIMENTAL SETUP

To validate the ML-based DDoS detection xApp and profile its run time, we set up an experimental testbed (Fig. 2) with radio transceivers, open-source Open RAN units, and simulated/real UEs. This setup provides realistic RA DDoS data to train ML models for the problem described in §3.

The testbed includes a laptop with Open RAN software, a Universal Software Radio Peripheral (USRP) functioning as the RU, a UE simulator, and a mobile phone. The laptop integrates RAN and RIC and connects to the USRP for radio frequency transmission/reception. A virtual machine hosts the Core network, providing Internet access to the UEs. Fig. 2 illustrates the proposed setup.

4.1 RU, RAN, and Core

The RU functionality is performed by a USRP B210, which is configured with the following radio frequency specifications: (1) band n41, (2) frequency 2574.27 MHz, (3) Single-Sideband Modulation (SSB) Frequency 2593.35 MHz, (4) bandwidth 40 MHz, (5) subcarrier spacing 30 KHz, and (6) Single Input Single Output (SISO) mode. We observe in Fig. 2 that the RU connects to the RAN via a USB 3.0 port.

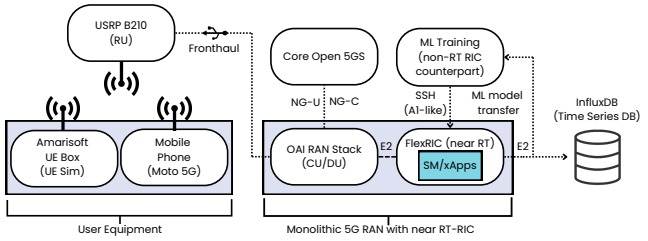

Figure 2: Testbed Setup used in DDoS detection timing.

We use a 3GPP-compliant, 5G-enabled OAI RAN stack, chosen for being open source and for easily integrating with the FlexRIC. The RAN implementation is monolithic, i.e., it integrates the CU and DU in the same virtual function. The interaction with the FlexRIC software is realized via localhost using the E2 interface (see Fig. 2) and its associated E2AP protocol. The hardware hosting the RAN and the near-RT RIC is a laptop with an Intel Core i7-1165G7 CPU (4 cores, 2.5 GHz) and 32 GB of RAM.

The core network used with the OAI RAN stack is the Open5GS, an open-source implementation of the Evolved Packet Core (EPC) that complies with 3GPP's Release 17 for NR/LTE networks. This core network is hosted on a separate virtual machine. As shown in Fig. 2, it connects to the RAN via the NG-U (user plane) and NG-C (control plane) interfaces. In our setup, the core network is linked to the gNodeB through an optical fiber connection.

4.2 User Equipment

In Fig. 2, we see there are two types of UEs connected to the RU via radio frequency: a mobile phone Moto 5G and the UE simulator Amarisoft UE Box. The Amarisoft UE software-hardware solution is able to simulate UE activity. It includes the physical, data link, and network layers, as well as traffic and channel simulators. This solution allows connecting simulated UEs to the RAN via radio frequency. The Amarisoft UE software runs on a virtual machine in close proximity to the laptop hosting the RAN. This proximity ensures a stable radio frequency connection, preventing experimental limitations due to connection issues. Amarisoft is the primary tool used for simulating attacks and also plays a role in increasing the number of normal samples. Details of Amarisoft's role in the experimental protocol and its use in data collection are provided in § 5.

4.3 Machine Learning Training Loop

Data for ML models is collected from the RAN by the near-RT RIC through a Key Performance Indicator (KPI) monitoring SM and stored in the InfluxDB time series database. As shown in Fig. 2, the setup does not include a non-RT RIC; instead, ML training is performed externally in the Cloud using the data stored in InfluxDB. Once trained, the models are deployed in the near-RT RIC, utilizing embedded ML tools such as TensorFlow Lite and micromlgen to convert them from Python to C and reduce their memory footprint. The connection between the cloud and FlexRIC is established via SSH. This setup serves as a proxy for the operation that would

typically be performed by the non-RT RIC using its A1 interface to connect to the near-RT RIC.

5 EXPERIMENTAL TESTING

The tests in this papers are study cases meant to represent scenarios identified during the trials with OAI RAN and Amarisoft UE simulation. However, especially in terms of attack behavior, we intend to study different attack patterns in future work.

Two types of tests are conducted in the experimental setup described in § 4. The first test simulates normal traffic, with $n_{\text{UE}} \in \{1, 2, 3, 4\}$ UEs being randomly turned on and off. The on periods are randomly selected from a uniform distribution ranging from 30 seconds to 1 minute, while the off periods are uniformly chosen from [5s, 10s]. These on and off periods are interleaved according to a Bernoulli(p) Random Variable with $p = 0.5$. This process is repeated until samples of approximately 6 minutes are obtained for all UEs. During the on periods, Amarisoft transmits UDP and TCP packets. Normal traffic is replicated with a single mobile phone following the same testing protocol for Amarisoft, with the difference that the mobile generates higher-layer traffic using HTTP.

The second test simulates DDoS attacks using Amarisoft by simultaneously attempting connection with 20, 32, or 64 UEs. The OAI NR stack overflows and crashes almost immediately in most experiments when 20 or more UEs attempt to connect. To guarantee a crash, additional connection requests from 20 UEs are made at intervals uniformly chosen between 1 and 5 seconds.

5.1 Machine Learning Model Definitions

The ML classifiers XGBoost (XGB), RF, and Multilayer Perceptron (MLP) [2] are tested in order to analyze their inference times within the near-RT RIC. Let n be the input size of the time series vector used for detection and n_{est} the number of estimators in ensemble methods.

Figure 3 illustrates examples of normal and attack behaviors for the experimental protocol chosen for this paper. Each point in the graph receives a label based on its current counter value and the values of the preceding $n - 1$ points. Therefore, the figure only provides enough information to detect the behavior at the last point in the series. A counter increment appears as a step, while a constant counter value is shown as a flat line. Attack behavior typically exhibits a step-wise ramp pattern, whereas normal behavior is characterized by long plateaus. Also, attacks have a high count difference between the end and the beginning of the series, while for normal behavior this difference is small.

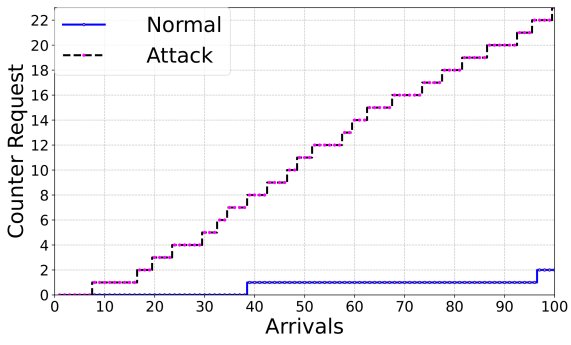

Figure 3: Normal and attack behavior (n=100).

For the MLP, $n = 64$ provides precision and recall results above 97%. RF and XGB only reach similar levels of performance for $n \geq 100$. Therefore, to preserve performances above the 97% mark and to carry out a fair comparison of inference delays, which are directly dependent on n, n is chosen as 100 for all classifiers. Other model hyperparameters are found by means of Bayesian optimization. In what follows, we list these parameters.

- *RF:* $n_{\text{est}} = 100$, Shannon Entropy as the tree node splitting criterion, maximum depth of 12, minimum of 2 samples required to split a node, minimum of 1 sample required at a leaf node, and random subsets of $\sqrt{n}$ (i.e., 10) features considered when splitting a node.
- *XGB:* $n_{\text{est}} = 50$, use of decision trees as the base estimators, maximum tree depth of 14, and learning rate $\eta = 0.1$ for step size shrinkage to prevent overfitting.
- *MLP:* 2 layers, ReLU activation function, sigmoid classification function, learning rate $\alpha = 0.001$ for the Adam optimization algorithm, 5101 total parameters.

Table 1 contains performance metrics of these algorithms, including precision, recall, F1-Score, and AUC. Results show that, with $n = 100$, the algorithms consistently exceed the target performance of 97% for both precision and recall.

Table 1: ML Model performances

CLF	Precision	Recall	F1-Score	AUC
XGB	0.99	0.99	0.99	0.99
RF	0.98	0.99	0.98	0.99
MLP	0.99	0.99	0.99	0.99

5.2 Temporal Analysis

As discussed in § 3.1, the xApp must operate within 8 ms to meet the CRT, which is faster than the O-RAN Alliance's near-RT RIC control loop delay of 10 ms to 1 s. It is therefore essential to measure the time of key procedures in an ML-based xApp: (1) T_{IB} for *input buffering* (IB), where new elements are added to a size n input FIFO queue; (2) T_{PP} for *pre-processing* (PP), where the input tuple is transformed by a function $\mathcal{F}$ (here, the difference function); (3) T_{MLI} for *ML inference* (MLI), the main focus of profiling, where model classification occurs and time depends on model architecture; and (4) T_{CS} for *control signaling* (CS), where the classification result is sent back to the RAN, completing the control loop. In our case, this consists in sending a recommendation to transmit MSG 4 based on the classification of UE activity.

Times are reported using the sample mean and standard deviation (SD). For a given procedure time T, the sample mean $\overline{T}$ is calculated from N samples, with variance σ^2/N from the true mean. The sample size N is chosen to keep the SD of $\overline{T}$ below 0.5 µs. For some measurements, like CS, this requires over 13,000 samples. However, practical constraints limit data collection to fewer than 10,000. As a result, in such cases, we aim for SDs below 1 µs instead of 0.5 µs.

In the table 2 we indicate the metrics used to measure the times of each of the stages of the xApp pipeline, for example, the average execution time and the standard deviation for each of the algorithm implementations.

The measurements in Table 2 are from sequential operations (assumed statistically independent). Hence, samples from different classifiers can be aggregated for IB, PP, and CS, resulting in a higher number of samples for these operations in Table 2. This table shows that all T_{IB} and T_{PP} are under 1 µs, with SDs too imprecise for meaningful interpretation. The CS time, estimated at 284 ± 58 µs from 8590 samples, accounts for over 70% of the control loop time for all classifiers.

We model the estimate $\hat{T}_{\text{Tot}}$ of the total duration of the entire ML xApp operation as the sum of the expected values of all the procedures in Table 2:

$$\hat{T}_{\text{Tot}} \triangleq \mathbb{E}\left[T_{\text{IB}}\right] + \mathbb{E}\left[T_{\text{PP}}\right] + \mathbb{E}\left[T_{\text{MLI}}\right] + \mathbb{E}\left[T_{\text{CS}}\right].$$

This underestimates the true T_{Tot}, as it does not take into account the component ζ, caused by CPU parallel processing, background functions that handle the connection between the xApp and the RAN, amongst other unmodeled effects. To address this, we develop the unbiased estimator $\hat{T}'_{\text{Tot}} \triangleq \hat{T}_{\text{Tot}} + \mathbb{E}\left[\zeta\right]$. The term $\mathbb{E}[\zeta]$ cannot be obtained directly. Instead, $\hat{T}'_{\text{Tot}}$ is given by the measurement of the average duration of the entire xApp operation, from which we then subtract $\hat{T}_{\text{Tot}}$ to obtain $\mathbb{E}[\zeta]$, which is important in its own right, as it indicates how important the measurements in Table 2 are to the final estimate of the xApp's total operation time.

Without loss of generality, we obtain the unbiased estimate for RF, as it has the most samples collected for all operations (5016), including the total run time which averages to approximately 371 µs (well below the 8 ms CRT threshold). This results in an estimate for $\mathbb{E}[\zeta]$ of about 40 µs, indicating that roughly 10% of the total time is not accounted for by the operations listed in Table 2. This proportion of unaccounted time is consistent across the other two classifiers as well.

Figure 4 presents inference time histograms for the XGB, RF, and MLP models, with sample sizes of 1272, 5016, and 2886, respectively. Histogram bins are determined using Kernel Density Estimation (KDE). Since Gaussian, Tophat, and Epanechnikov kernels yield nearly identical results, we arbitrarily select Gaussian kernels. Kernels with bandwidths 3.1 for XGB, 3.2 for RF, and 4.5 for MLP are empirically chosen to balance variance and bias in Probability Density Function (PDF) estimation. The number of histogram bins is also empirically chosen and is selected for best fit to the KDEs, resulting in 86 bins for XGB, 240 for RF, and 150 for MLP.

Table 2: Breakdown of times of xApp Component Tasks

Tasks	No. Samples	Mean (μs)	SD (μs)
IB	8590	< 1	–
PP	8590	< 1	–
MLI:			
XGB	1272	28	17
RF	5016	44	20
MLP	2302	77	24
CS	8590	288	60

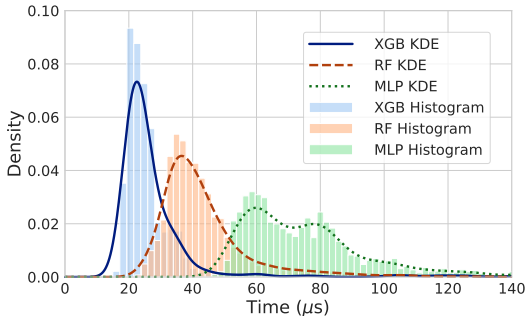

Figure 4: Histograms and KDES: XGB, RF, and MLP.

Figure 4 and Table 2 show that xApp operation with all classifiers tested is realized within the CRT deadline. XGB has the lowest mean execution time (28 ± 17 µs), with the thinnest PDF tail, indicating less variability and fewer outliers. RF has a higher mean (44 ± 20 µs), while MLP has the highest mean (77 ± 24 µs), a bimodal distribution, and the thickest PDF tail.

Results indicate that XGB is advantageous due to its low latency and deterministic behavior, which reduce the window of opportunity for attackers to exploit RA MSG 3 and enable the timely planning of DDoS countermeasures. Furthermore, XGB presents a lower computational complexity compared to RF, and it is easier to deploy than MLP due to the build of TensorFlow lite library in C.

6 CONCLUSION

A testbed using the Open RAN paradigm has been developed to profile the time of a ML-based xApp for RA DDoS detection. Our results show that all tested algorithms operate within CRT requirements, with XGB providing the fastest and most deterministic inference.

As future work, we could add dimensions to the time series, including variables like Timing Advance, and embed unsupervised anomaly detection algorithms in the near-real-time RIC. We could also explore the RA DDoS SM and the complexity of ML algorithms used in RA DDoS detection.

Other research directions which we intend to study include the use of different attacker behaviors with more variation in frequency and time and the study of detection algorithm time complexity. In particular, this analysis must observe how complexity compares with CRT as block lengths grow for more complex attack patterns, which require more history from the time series to identify attack events.

ACKNOWLEDGMENTS

This work has been partially funded by the AITORAN project supported by the EMBRAPII/CPQD EXCellence Centre in Open Networks, with financial resources from the PPI IoT/ Manufatura 4.0/PPI HardwareBR of the MCTI grant number 51/2023, signed with EMBRAPII.

REFERENCES

[1] 3GPP. 2024. *NR; Medium Access Control (MAC) protocol specification.* Technical Specification (TS) TS 38.321. 3rd Generation Partnership Project (3GPP). Version 18.2.0.

[2] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. 2009. *The elements of statistical learning: data mining, inference, and prediction.* Vol. 2. Springer.

[3] Stefano Maxenti, Salvatore D'Oro, Leonardo Bonati, Michele Polese, Antonio Capone, and Tommaso Melodia. 2024. ScalO-RAN: Energy-aware Network Intelligence Scaling in Open RAN. In *Proc. of IEEE Intl. Conf. on Computer Communications (INFOCOM).*

[4] Redouane Niboucha, Sabra Ben Saad, Adlen Ksentini, and Yacine Challal. 2022. Zero-touch security management for mMTC network slices: DDoS attack detection and mitigation. *IEEE Internet of Things Journal* 10, 9 (2022), 7800–7812.

[5] Turgay Pamuklu, Shahram Mollahasani, and Melike Erol-Kantarci. 2021. Energy-efficient and delay-guaranteed joint resource allocation and du selection in o-ran. In *2021 IEEE 4th 5G World Forum (5GWF).* IEEE, 99–104.

[6] Jesus Arturo Perez-Diaz, Ismael Amezcua Valdovinos, Kim-Kwang Raymond Choo, and Dakai Zhu. 2020. A flexible SDN-based architecture for identifying and mitigating low-rate DDoS attacks using machine learning. *IEEE Access* 8 (2020), 155859–155872.

[7] Michele Polese, Leonardo Bonati, Salvatore D'Oro, Stefano Basagni, and Tommaso Melodia. 2022. ColO-RAN: Developing machine learning-based xApps for open RAN closed-loop control on programmable experimental platforms. *IEEE Transactions on Mobile Computing* 22, 10 (2022), 5787–5800.

[8] Michele Polese, Leonardo Bonati, Salvatore D'oro, Stefano Basagni, and Tommaso Melodia. 2023. Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges. *IEEE Communications Surveys & Tutorials* 25, 2 (2023), 1376–1411.

[9] Kshira Sagar Sahoo, Bata Krishna Tripathy, Kshirasagar Naik, Somula Ramasubbareddy, Balamurugan Balusamy, Manju Khari, and Daniel Burgos. 2020. An evolutionary SVM model for DDOS attack detection in software defined networks. *IEEE access* 8 (2020), 132502–132513.

[10] J Xavier Salvat, Jose A Ayala-Romero, Lanfranco Zanzi, Andres Garcia-Saavedra, and Xavier Costa-Perez. 2023. Open radio access networks (O-RAN) experimentation platform: Design and datasets. *IEEE Communications Magazine* 61, 9 (2023), 138–144.

[11] Robert Schmidt, Mikel Irazabal, and Navid Nikaein. 2021. FlexRIC: An SDK for next-Generation SD-RANs. In *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies* (Virtual Event, Germany) *(CoNEXT '21).* Association for Computing Machinery, New York, NY, USA, 411–425. https://doi.org/10.1145/3485983.3494870

[12] Pratheek S Upadhyaya, Nishith Tripathi, Joseph Gaeddert, and Jeffrey H Reed. 2023. Open AI Cellular (OAIC): An Open Source 5G O-RAN Testbed for Design and Testing of AI-Based RAN Management Algorithms. *IEEE Network* (2023).

[13] D. Villa, I. Khan, F. Kaltenberger, N. Hedberg, R. Soares da Silva, A. Kelkar, C. Dick, S. Basagni, J. M. Jornet, T. Melodia, M. Polese, and D. Koutsonikolas. 2024. An Open, Programmable, Multi-vendor 5G O-RAN Testbed with NVIDIA ARC and OpenAirInterface. In *Proc. of the 2nd IEEE Workshop on Next-generation Open and Programmable Radio Access Networks (NG-OPERA)* (Vancouver, Canada).

[14] Bruno Missi Xavier, Merim Dzaferagic, Diarmuid Collins, Giovanni Comarela, Magnos Martinello, and Marco Ruffini. 2023. Machine learning-based early attack detection using open RAN intelligent controller. In *ICC 2023-IEEE International Conference on Communications.* IEEE, 1856–1861.